Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Program na podporu výuky prstové abecedy

2025

Bc. Filip Martiník

Vedoucí práce:
doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní program: Aplikovaná informatika,
Specializace: Vývoj software

Bibliografické údaje

Autor:	Bc. Filip Martiník
Název práce:	Program na podporu výuky prstové abecedy
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2025
Studijní program:	Aplikovaná informatika, Specializace: Vývoj software
Vedoucí práce:	doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	50
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Bc. Filip Martiník
Title:	Program to support the teaching of the finger alphabet
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2025
Study program:	Applied Computer Science, Specialization: Software Development
Supervisor:	doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	50
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Práce pojednává o představení prstové abecedy a jejím významu. Zabývá se problémem převodu textu na 3D animaci prstové abecedy a rozpoznáním prstové abecedy z videa. Tyto funkcionality jsou následně spojeny v jednu webovou aplikaci, která umožňuje podporu výuky české jednoruční prstové abecedy.

Synopsis

The thesis discusses the introduction of the sign alphabet, the problem of converting text into a 3D animation of the finger alphabet, and the challenge of recognizing the finger alphabet from video. These methods are then combined into a single web application that supports the learning of the Czech one-handed finger alphabet.

Klíčová slova: Prstová abeceda, animace, rozpoznání gest ruky

Keywords: Finger alphabet, animations, hand gesture recognition

Děkuji panu doc. RNDr. Miroslavovi Kolaříkovi, Ph.D. za veškeré rady, připomínky a odborné vedení diplomové práce.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Potřeba naučit se znakový jazyk a prstovou abecedu může potkat každého. Pokud se člověk zaměří na nabízené možnosti učení, tak zjistí, že jich na internetu není mnoho. Tato práce se snaží vytvořit výchozí bod pro seznámení s prstovou abecedou a také možnosti jejího procvičování.

1.1 Zadání

Diplomant vytvoří program, který bude umět převádět zadaný text na odpovídající 3D animaci s prstovou abecedou, přičemž si uživatel bude moci nastavit vhodnou rychlost přehrávání. Diplomant se také pokusí o opačný proces, tedy že program na základě videa (záznamu webkamery) s prstovou abecedou napíše odpovídající zprávu v textové podobě. Cílem práce je vytvoření aplikace, která spojí tyto funkcionality a vytvoří tím prostředí pro podporu výuky jednoruční prstové abecedy.

1.2 Existující řešení

V této části jsou popsána některá existující řešení, nejen pro českou prstovou abecedu, ze kterých vytvořená aplikace čerpá inspiraci pro vhodné vlastnosti.

- *Znakujte s Tamtamem*

  Povedená mobilní aplikace, která obsahuje základní znaky slov českého znakového jazyka. Znaky třídí do různých kategorií a zobrazuje je pomocí videozáznamu, na kterém je člověk znakující daný znak. Aplikace umožňuje označovat znaky za již naučené a pro procvičování nabízí kvízy, které jsou založeny na tom, že uživatel musí zvolit jednu ze tří odpovědí na základě zobrazeného znaku pomocí videa. [1]

- *Webové stránky*

  Volně dostupné webové stránky s obsahem o jednoruční české prstové abecedě nebo i o ČZJ (český znakový jazyk) obsahují většinou pouze několik odstavců textu se statickým obrázkem obsahujícím jednotlivé znaky. Tyto stránky jsou většinou doprovázeny odkazy na placené kurzy.

- *Hand Talk Translator*

  Mobilní aplikace, která umí převádět text do 3D animace amerického a brazilského znakového jazyka. Pokud aplikace pro dané slovo nemá animovaný znak, zobrazí jej pomocí prstové abecedy. Vzhled aplikace je na obrázku 1. [2]

- *Sign Language ASL Pocket Sign*

Obrázek 1: Aplikace Hand Talk Translator

Mobilní aplikace pro výuku amerického znakového jazyka. Také využívá videozáznamy pro zobrazování znaků. Navíc tyto videozáznamy dobře integruje do vzhledu aplikace, protože využívá záznamy s vklíčovaným pozadím. Neumožňuje však volbu rychlosti přehrávání či ovládání úhlu pohledu. [3]

- *www.spreadthesign.com*

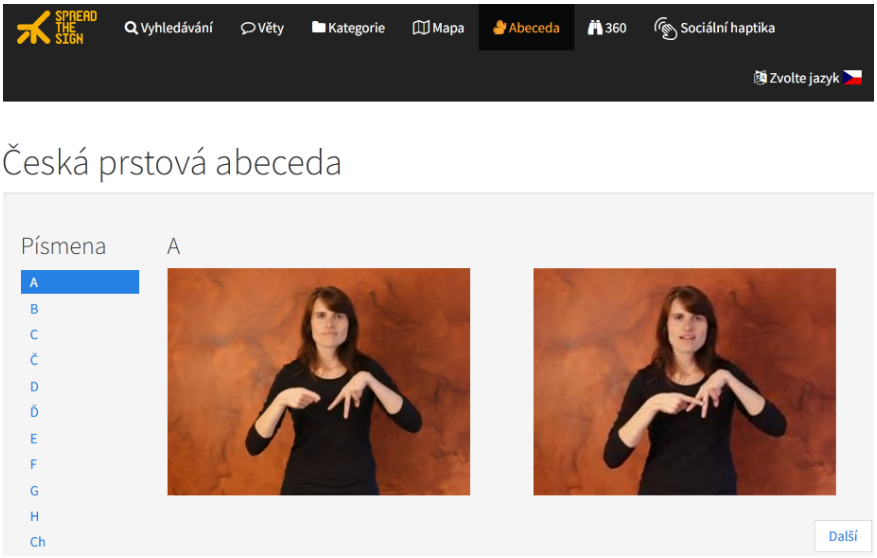
  Webová stránka s obsahem znaků slov a písmen různých znakových jazyků, včetně českého. Obsahuje slovník s videozáznamy. Vzhled stránky zobrazuje obrázek 2. [4]

- *Mnoho dílčích projektů*

  Na internetu lze najít řadu malých projektů, které například implementují rozpoznání prstové abecedy z webkamery, nicméně jsem nedohledal žádnou funkční aplikaci, která by podobnou funkcionalitu nabízela obzvlášť za účelem výuky či v českém jazyce.

Většina existujících řešení využívá videozáznamy pro ukázky znaků. Velkou výhodou při použití animací místo videozáznamů je flexibilita aplikace a příjemnější vzhled. Animace například nabízí možnost změnit úhel pohledu nebo změnu rychlosti přehrávání bez ztráty plynulosti.

Žádné z prozkoumaných řešení nenabízí funkci ověření správnosti znakování uživatelem pomocí rozpoznávání. Vyvinutá aplikace tedy přichází s inovativním přístupem pomocí rozpoznání znaků z webkamery a tím nabízí nový přístup, jak se učit prstovou abecedu.



Obrázek 2: Webová stránka www.spreadthesign.com

1.3 Funkcionalita aplikace

Tato podkapitola uvádí seznam všech požadavků na funkcionalitu a chování vyvinuté aplikace. Tento seznam vychází zejména ze zadání práce, ale obsahuje i aspekty, kterými byla aplikace rozšířena nad rámec zadání. Tento seznam slouží pro lepší představu o vyvíjené aplikaci a také k sjednocení všech funkcionalit aplikace.

- Převod textu na plynulou sekvenci 3D animací interpretující daný text v české jednoruční prstové abecedě.
  - Možnost měnit rychlost přehrávání animace a jejího zastavení.
  - Možnost ovládání úhlu pohledu.
- Rozpoznání zobrazeného znaku na snímku z webkamery.
  - Za účelem ověření, zda je zobrazen správný znak.
  - Za účelem pokusu o přepsání souvislé sekvence snímků do souvislého textu.
- Různé režimy sloužící k podpoře výuky a procvičování české jednoruční prstové abecedy.
- Responzivní webová aplikace, kterou lze používat jak na velkých, tak i malých obrazovkách.
- Možnost používání aplikace bez přístupu k internetu.
- Možnost volby dvouruční české prstové abecedy a českého znakového jazyka v režimech nevyžadujících rozpoznávání znaků.

2 Znakový jazyk

Znakové jazyky představují plnohodnotné systémy, které slouží jako primární komunikační prostředek pro komunity neslyšících a nedoslýchavých po celém světě. Na rozdíl od jazyků mluvených, které využívají zvuk, jsou znakové jazyky založeny na vizuální podobě. To znamená, že informace jsou předávány prostřednictvím kombinace tvarů rukou, jejich umístění a pohybu v prostoru, mimiky, pohybů hlavy a horní části trupu. Je klíčové zdůraznit, že znakové jazyky nejsou pouhou pantomimou nebo manuální reprezentací mluvených jazyků; mají vlastní komplexní gramatiku, syntaxi a bohatou slovní zásobu.

Podobně jako mluvené jazyky, i znakové jazyky vykazují značnou geografickou a kulturní diverzitu. Neexistuje jeden univerzální znakový jazyk. Prakticky každý stát má svůj vlastní unikátní znakový jazyk (např. Český znakový jazyk – ČZJ, Americký znakový jazyk – ASL atd.), který se vyvíjel nezávisle. Tato rozmanitost se týká nejen samotných znaků pro slova a pojmy, ale také systémů pro reprezentaci písmen – prstových abeced. [5]

2.1 Prstová abeceda

Nedílnou součástí většiny znakových jazyků je prstová abeceda. Jedná se o systém manuálních znaků (pozic ruky a prstů), které vizuálně reprezentují jednotlivá písmena psané abecedy daného jazyka. Její primární funkcí není běžná komunikace, pro kterou slouží komplexní znaky reprezentující celé pojmy, ale spíše komunikace v situacích, kdy pro daný pojem neexistuje ustálený znak, nebo je potřeba zdůraznit psanou podobu slova. Typické využití zahrnuje vlastní jména nebo odborné termíny.

Prstové abecedy se celosvětově liší nejen tvary jednotlivých znaků pro písmena, ale také základním principem jejich provedení. Hlavní dělení je podle počtu rukou zapojených do znakování:

- *Jednoruční prstové abecedy*

  K vytvoření znaku pro písmeno se používá pouze jedna ruka (obvykle dominantní). Tvary ruky a prstů se snaží stylizovaně napodobit tvar písmene.

- *Dvouruční prstové abecedy*

  K vytvoření znaku pro písmeno se používají obě ruce, které vzájemně spolupracují. Jedna ruka často tvoří základ tvaru a druhá jej doplňuje, nebo ruce společně formují tvar písmene.

I v rámci jednoho typu mohou existovat drobné regionální či historické varianty jednotlivých znaků.

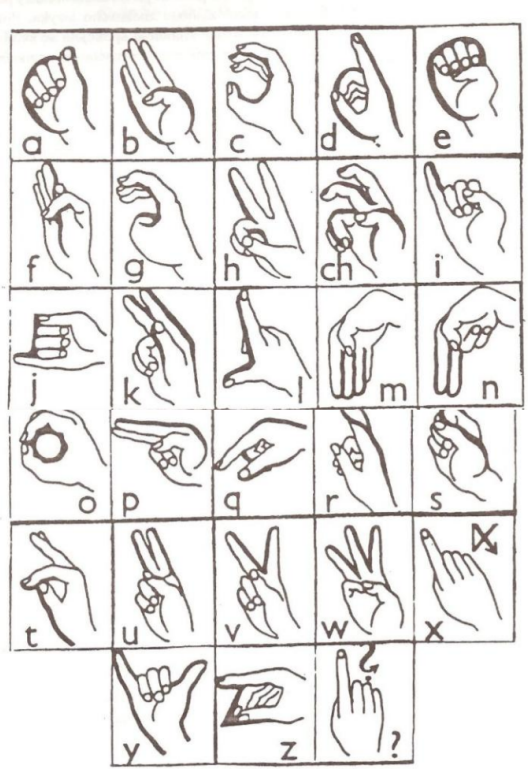
2.2 Prstové abecedy v České republice

V českém prostředí se setkáváme s oběma hlavními typy prstových abeced – jednoruční i dvouruční.

2.2.1 Česká jednoruční prstová abeceda

Tato diplomová práce a výsledná aplikace se zaměřují specificky na podporu výuky české jednoruční prstové abecedy. Důvodem pro toto zaměření je, že pro slyšící začátečníky může představovat určitou výzvu. Ačkoliv se tvary ruky snaží vizuálně přiblížit podobě písmen, vyžadují poměrně vysokou míru přesnosti v provedení a pozici prstů. Některé znaky si mohou být pro nezkušené oko velmi podobné, což může zpočátku činit potíže při jejich rozlišování a učení.

Vizualizaci všech znaků používaných v české jednoruční prstové abecedě poskytuje obrázek 3.



Obrázek 3: Česká jednoruční prstová abeceda [6]

2.2.2 Reprezentace diakritiky

Česká jednoruční prstová abeceda, stejně jako český jazyk, pracuje s diakritickými znaménky (háčky a čárky). Jejich reprezentace není řešena samostatným znakem, ale dynamickým pohybem ruky, který následuje po ukázání základního znaku písmene.

- Pro *čárku* se po zobrazení znaku písmene provede krátký pohyb ruky shora dolů, vizuálně napodobující tvar čárky.
- Pro *háček* se obdobně provede pohyb ruky naznačující tvar háčku (obvykle krátký pohyb šikmo dolů a šikmo vzhůru).

3 Převod textu na animaci

Cílem této části je implementace převodu textu na 3D animaci, ve které bude postava ukazovat sekvenci znaků odpovídající danému textu s možností volby rychlosti přehrávání animace.

K vytvoření takového systému je potřeba vytvořit 3D model postavy, vytvořit pro tento model animace konkrétních znaků a následně tento model vykreslit a řídit pomocí algoritmu posloupnost zobrazených animací znaků na vykresleném modelu. Tento komplexní úkol je rozdělen do vhodných menších částí, které jsou popsány v následujících podkapitolách.

3.1 Použité technologie

Jedinou použitou technologií při tvorbě animací na již vytvořeném 3D modelu postavy je Blender.

3.1.1 Blender

Blender je open-source software pro tvorbu 3D grafiky, který je zdarma dostupný pro všechny platformy. Tento program nabízí širokou škálu nástrojů pro modelování, texturování, animaci a rendering 3D objektů. Blender je velmi populární mezi profesionály i amatéry díky své flexibilitě a rozsáhlé komunitě, která neustále přispívá k jeho rozvoji. Při vývoji aplikace sloužil Blender pro vytvoření animací znaků na již vytvořeném modelu člověka. [7]

3.2 Animování modelu

Prvním krokem je vytvoření 3D modelu postavy, který zobrazuje jednotlivé znaky v podobě animací. Každý znak je reprezentován jednou animací, která je následně přehrávána v sekvenci podle zadaného textu.

3.2.1 3D model člověka

Nejprve je potřeba vytvořit 3D model člověka. Takový model lze vytvořit například v programu Blender. Vytvoření vizuálně zdařilého modelu postavy není triviální záležitost, z tohoto důvodu byl použit již vytvořený model.

Mixamo [8] je online platforma, kterou vlastní společnost Adobe. Poskytuje širokou škálu 3D modelů a animací zdarma. Uživatelé mohou na Mixamo najít předem vytvořené 3D modely postav a animace, které lze snadno integrovat do různých 3D projektů. Mixamo také umožňuje automatické rigování, což znamená, že uživatelé mohou nahrát své vlastní 3D modely a Mixamo automaticky přidá kostru postavy, umožňující pohyb daného modelu. Tato platforma je velmi užitečná pro vývojáře her, animátory a další profesionály pracující s 3D grafikou. Model postavy byl použit právě z této platformy.

3.2.2 Animace

Animování modelu je náročný proces. Zjednodušeně řečeno, vytvoření animace probíhá nejdříve pojmenováním animace, poté pomocí časové osy se v konkrétních časech nastavují pozice a rotace kostry modelu. Nastavení kostry ale není potřeba dělat pro každý snímek dané animace, chybějící snímky jsou automaticky dopočítávány.

Manuální vytváření animací je velmi časově náročné, proto byla využita *Pose Library*, která uchovává jedno konkrétní nastavení kostry. Jedno nastavení zastupuje výchozí stav, ze kterého všechny ostatní animace vycházejí, a dále je pro každý znak vytvořeno jedno nastavení kostry reprezentující dané písmeno.

Výsledná animace je vytvořena tak, že se první snímek animace nastaví na výchozí pozici a poslední snímek na pozici znaku. Animace je i přesto velmi plynulá, jelikož jsou chybějící snímky dopočítávány.

Nicméně, při tomto jednoduchém přístupu mohou nastávat nežádoucí situace, kdy například prst projde jiným prstem. Tento problém by bylo možné vyřešit použitím více nastavení kostry v průběhu animace. Ve výsledné aplikaci je tento problém řešen prolnutím animací do sebe (viz 5.3.3), přičemž zmíněné situace nijak nenarušují dojem z plynulosti a správnosti animace.

3.2.3 Vytvoření animací pomocí aplikace Blender

V této sekci je stručný popis funkcionalit, který popisuje proces tvorby animací. Prostředí aplikace při tvorbě animací je zobrazeno na obrázku 4.

- Model

  Nejprve je potřeba vytvořit nebo nahrát již vytvořený model s kostrou. Blender podporuje mnoho formátů 3D modelů, včetně formátu *FBX*, který je podporován platformou Mixamo.

- Pozice

  V režimu *Pose* je možné nastavovat pozice a rotace jednotlivých kostí v kostře modelu.

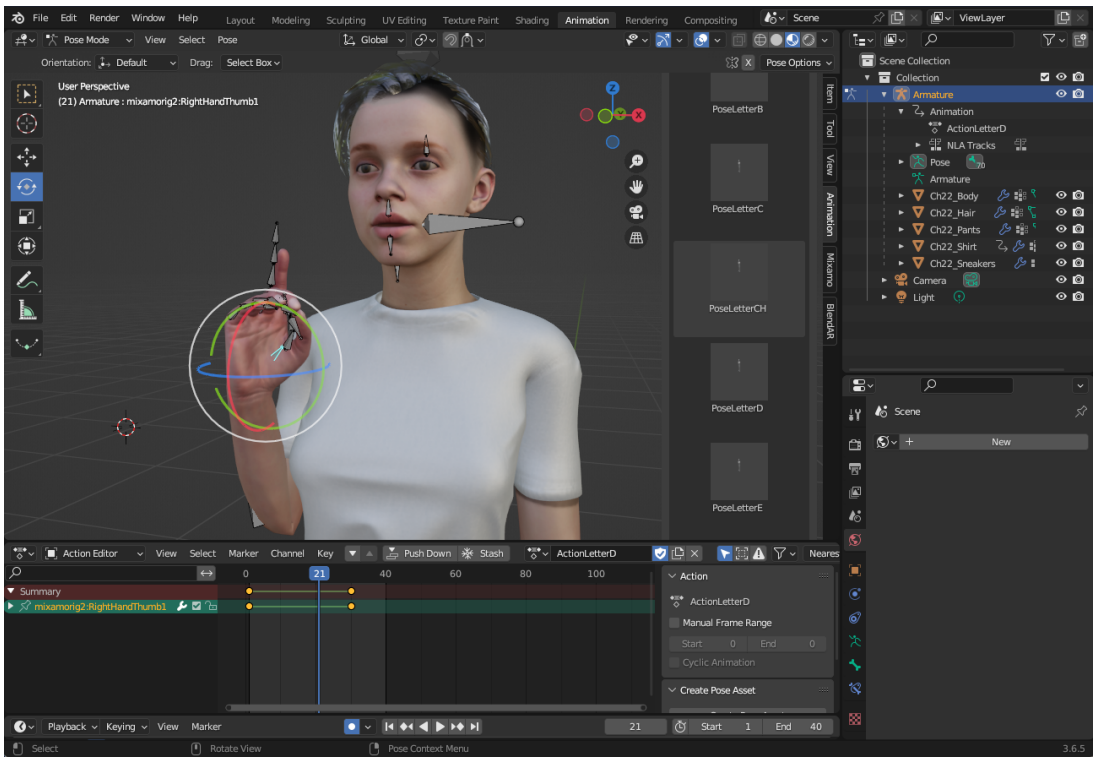
- Animace

  Animace se vytváří pomocí *Action Editor*, který obsahuje již zmíněnou časovou osu a další nástroje pro vytváření animací. Průběh tvorby si lze představit tak, že se nastaví konkrétní čas na časové ose, zapne se tlačítko záznamu a následně se nastaví pozice a rotace kostí. Tím se dané nastavení kostry aplikuje v daném čase animace. V použitém případě se na konkrétní časové místo aplikuje kostra z *Pose Library*.

- Výraz obličeje

  Ve znakovém jazyku je důležitý pohyb úst. Tento pohyb ale není možné zobrazit pomocí kostry, jelikož ústa a obličej celkově nejsou součástí výchozí kostry. Tento problém je možné vyřešit například pomocí *Shape keys*.

Pomocí této techniky lze vytvářet jednoduché deformace modelu, které lze ovládat například pomocí pohybu nové kosti, kde se pozice kosti prováže s hodnotou *Shape key*. Poté lze upravovat výraz obličeje pomocí pozice konkrétních kostí jako u výchozích kostí kostry.



Obrázek 4: Ukázka animování v aplikaci Blender

3.2.4 Exportování modelu

Pro použití modelu na webové stránce je klíčová jeho velikost. Použitý model při výchozím uložení obsahuje zhruba 100 MB dat ve formátu glb. Taková velikost je pro použití na webové stránce příliš velká, proto je potřeba model specificky exportovat, s cílem minimalizovat jeho velikost. Model lze komprimovat dvěma hlavními způsoby. Komprimovat lze samotný 3D model pomocí Draco komprese, a textury modelu lze komprimovat například převodem do .jpg formátu nebo zmenšením jejich rozlišení. Kombinací těchto dvou přístupů vznikl model s velikostí kolem 10 MB.

3.3 Řízení přehrávání animací

Vytvořený model je potřeba nějakým způsobem zobrazit a spouštět jeho animace ve správném pořadí a načasování. V této sekci je daný problém popsán obecně.

Konkrétní vykreslení modelu a řízení animací je popsáno v kapitole Webová aplikace (kapitola 5), která využívá knihovnu Threlte spolu s vytvořeným modelem z této kapitoly.

3.3.1 Řídící algoritmus

Animace je potřeba přehrávat postupně za sebou. Tedy je nutné vstupní text rozložit na jednotlivé animace znaků. Pro každý jeden znak je potom spuštěna animace, která odpovídá danému znaku a zvolenému jazyku. Je potřeba detekovat konec této animace a na jejím konci spustit další animaci pro následující znak. Popsaný postup je opakován, dokud nejsou všechny znaky přehrány.

K přechodům jsou poté dodány prodlevy. Na konci každé animace je zobrazený znak na chvíli ponechán. Na konci celých slov je také vytvořena prodleva na signalizaci mezery mezi slovy.

3.3.2 Plynulý přechod mezi animacemi

Jelikož nejsou animace v tomto případě příliš sofistikovaně vytvořeny a nenavazují na sebe, tak je potřeba řídit jejich přechody tak, aby vypadaly přirozeně. Tento problém lze vyřešit splynutím konce aktuální animace do začátku následující animace.

3.3.3 Ovládací prvky

Důležitým aspektem a výhodou přehrávání animací je volba její rychlosti. Tato funkcionalita je pro cílovou aplikaci velice vhodná. Začátečníci mohou používat aplikaci v pomalejším tempu a pokročilým uživatelům umožní využívat různé režimy ve vyšší rychlosti a tím výrazně zvýšit náročnost procvičování.

Volba rychlosti také ovlivňuje délky prodlev, kde s pomalejší rychlostí přehrávání časové rozmezí prodlev stoupá a díky tomu se nenaruší plynulost celého přehrávání animace.

Animaci lze v průběhu zastavit a znovu spustit. V průběhu animace umožní tento ovládací prvek zastavení animace v konkrétním bodě a uživatel si poté může pohodlně prohlédnout zobrazený znak.

Zmíněné zastavení se váže i s dalším ovládacím prvkem, který je umožněn díky vykreslení modelu ve 3D prostoru. To umožňuje pohybovat kamerou ve všech směrech a tím si prohlédnout dané znaky z jakéhokoliv úhlu a pozice.

4 Rozpoznání prstové abecedy z videa

Následující problém je řádově složitější a zajímavější než ten předchozí. Pokud bychom výše popsaný převod z textu na animaci otočili a vyměnili animaci za video z reálného světa, tak se dostaneme k následujícímu problému – převod české jednoruční prstové abecedy z videozáznamu na text.

Tento problém lze také rozdělit do několika podproblémů, konkrétně na rozpoznání souřadnic prstů ze snímku videozáznamu, zpracování těchto souřadnic, rozpoznání zobrazovaného znaku ze zpracovaných souřadnic a převod všech rozpoznání na souvislý text. Při vytváření rozpoznávání narazíme ještě na jeden podproblém a tím je dataset s trénovacími daty.

4.1 Použité technologie

V této podkapitole jsou popsány klíčové technologie, kterými je problém rozpoznání prstové abecedy z videa řešen.

4.1.1 MediaPipe Solutions

Souhrn knihoven a nástrojů od společností Google, které slouží k různým rychlým nasazením metod umělé inteligence do aplikace. Nabízí podporu pro Android, iOS, JavaScript a Python. MediaPipe již obsahuje několik hotových, v některých případech rozšiřitelných, řešení například pro rozpoznání obličeje, objektovou klasifikaci, text embedding a zejména rozpoznání orientačních bodů (souřadnic) ruky. Poslední zmíněné řešení je klíčovou součástí vyvinuté aplikace a její použití značně zjednodušilo vývoj a umožnilo vysokou přesnost tohoto systému. [9]

4.1.2 TensorFlow

TensorFlow je open-source knihovna pro strojové učení vyvinutá společností Google. Je velmi populární mezi vývojáři díky své flexibilitě a široké škále nástrojů pro vytváření a trénování modelů strojového učení. TensorFlow je dostupný pro všechny hlavní platformy a podporuje mnoho programovacích jazyků, včetně Pythonu a JavaScriptu. TensorFlow je použit pro vytvoření modelu pro rozpoznání prstové abecedy. [10]

4.1.3 TensorFlow Decision Forests

TensorFlow Decision Forests (zkráceně TFDF) je knihovna pro TensorFlow, která poskytuje nástroje pro vytváření a trénování rozhodovacích lesů. Rozhodovací lesy jsou populární metodou strojového učení, která se používá pro klasifikaci a regresi dat. TFDF je použit pro vytvoření modelu pro rozpoznání prstové abecedy. [11]

4.1.4 Gradient Boosted Trees

Gradient Boosted Trees (GBT) je výkonná metoda strojového učení založená na vícero rozhodovacích stromech. Pracuje iterativně, přičemž se každý nový strom učí opravovat chyby předchozích stromů minimalizací gradientu ztrátové funkce. GBT je široce používán pro klasifikaci i regresi díky své schopnosti efektivně zachytit složité vzory v datech. Populární knihovny implementující GBT zahrnují XGBoost, LightGBM a TensorFlow Decision Forests. [12]

4.1.5 Python

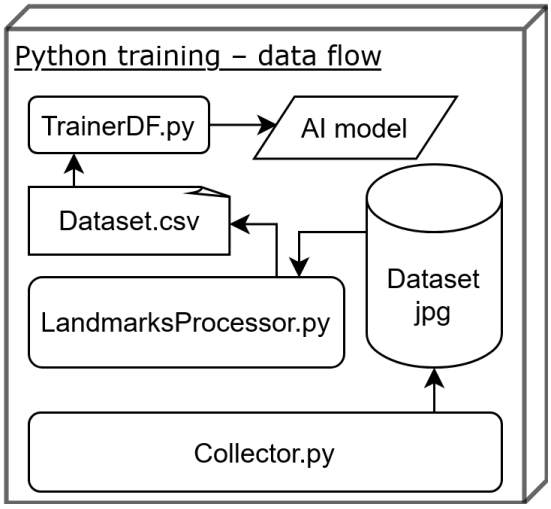
Python je všeobecný programovací jazyk, který je velmi populární mezi vývojáři díky své jednoduché syntaxi a široké škále knihoven a nástrojů pro různé účely. Python je použit pro tvorbu skriptů, které vedou k vytvoření modelu pro rozpoznání prstové abecedy.

4.2 Proces rozpoznání gest

Rozpoznání zobrazeného znaku ze snímku kamery probíhá následujícími kroky.

1. Rozpoznání pozice ruky a prstů
2. Zpracování rozpoznaných dat
3. Klasifikace zpracovaných dat

Všechny zdrojové kódy sloužící k tvorbě modelu pro rozpoznávání gest se nacházejí ve složce *python_recognition*. Diagram zobrazující tok dat mezi jednotlivými částmi při tvorbě klasifikátoru lze vidět na obrázku 5.



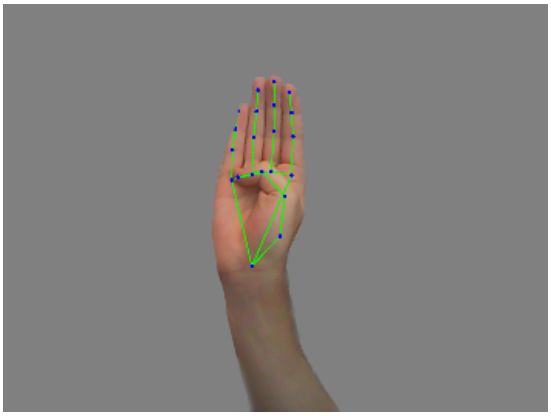
Obrázek 5: Diagram toku dat při vytváření klasifikátoru

4.2.1 Extrakce souřadnic bodů ruky

Pomocí modelu Hand Landmark Detection z frameworku MediaPipe lze získat z obrazu ruky 21 souřadnic, které reprezentují orientační body ruky. Tyto souřadnice jsou vyjádřeny trojicí (x, y, z), kde x a y jsou souřadnice bodů relativně k celému snímku a z je vzdálenost od kamery.

Velkou výhodou využití MediaPipe Hand Landmark Detection je vysoká přesnost určení souřadnic bodů ruky, a to i za špatných světelných podmínek, při různých rotacích ruky nebo při jakémkoliv pozadí, před kterým uživatel svou ruku zobrazuje. Názorné rozpoznání souřadnic lze vidět na obrázku 6.

Tento fakt umožňuje se více zaměřit na klasifikaci rozpoznaných souřadnic a tím se abstrahovat od problému správné detekce bodů ruky a prstů.



Obrázek 6: Rozpoznání 21 souřadnic ruky pomocí MediaPipe Solutions

4.2.2 Zpracování souřadnic

Pro efektivní zpracování souřadnic metodami strojového učení je potřeba souřadnice převést do vhodného formátu. Aplikace souřadnice nejprve převede tak, aby byly reprezentovány relativně k souřadnici pozice zápěstí (souřadnice s indexem 0) a tím zanikne závislost na pozici ruky v určité části snímku kamery. Následně jsou souřadnice normalizovány do rozsahu $[-1, 1]$ a tím zaniká závislost na vzdálenosti ruky od kamery.

Algoritmus zpracovává souřadnice x a y, hodnota z je ignorována. Tím je výsledný klasifikátor značně zjednodušen. Model s menší dimenzionalitou vyžaduje méně dat při učení a snižuje se riziko přeučení modelu při použití malého datasetu. Hodnota z je také méně spolehlivá/přesná a z těchto důvodů je ignorována, ačkoliv by poskytovala další informaci o zobrazeném gestu ruky.

Tento postup vychází z již existujícího řešení tvorby klasifikátoru gest ruky [13].

4.2.3 Klasifikace

Klasifikace je metoda, která na vstupu přijímá zpracované souřadnice prstů ruky a na základě těchto souřadnic vyhodnocuje, s jakou pravděpodobností je jaký znak vyobrazen.

4.3 Tvorba klasifikátoru

Při vytváření klasifikátoru je nejdůležitější mít dataset obsahující data, na kterých bude model trénován. Nad takovými daty je poté spuštěn trénovací algoritmus, jehož výstupem je model, který lze použít pro klasifikaci souřadnic ruky.

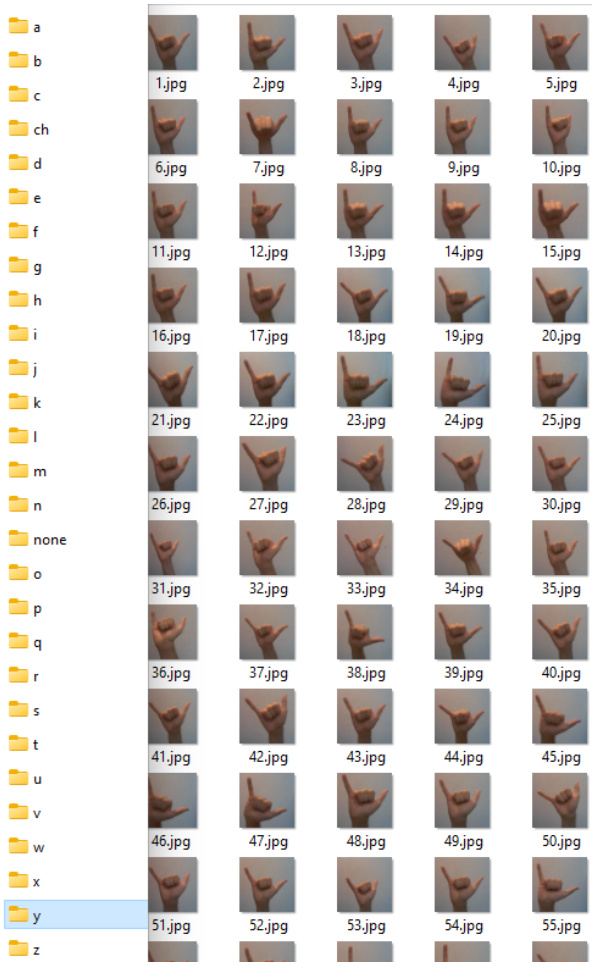
4.3.1 Dataset

Datasetem jsou v tomto případě fotky, na kterých je viditelná ruka člověka zobrazující nějaký konkrétní znak. Tyto fotky jsou v jednotlivých složkách, kde každá složka obsahuje fotky s jedním konkrétním znakem. Velikost datasetu je zhruba 100 fotek jedné osoby a 25 fotek druhé osoby na každý znak české jednoruční prstové abecedy pro levou i pravou ruku. K těmto znakům je ještě přidán znak *None*, reprezentující žádný znak za účelem zpřesnění klasifikace. Dohromady dataset obsahuje 7169 snímků v rozlišení 250x250 pixelů. Strukturu a ukázku obrázků datasetu zobrazuje obrázek 7.

Každý obrázek je převeden do formátu zpracovaných souřadnic a uložen do .csv souboru, který je použit jako vstup pro trénovací algoritmus.

K tvorbě datasetu existují dva skripty v jazyce Python ulehčující tento proces. Prvním je *Collector.py*, který v okně zobrazí záznam webkamery a odposlouchává stisky kláves klávesnice. Pokud uživatel stiskne písmeno na klávesnici, tak se algoritmus pokusí získat souřadnice prstů ruky z bufferu několika posledních snímků kamery. Nejnovější ořezaný snímek, ze kterého lze rozeznat souřadnice prstů ruky, se uloží do složky s názvem odpovídajícím stisknuté klávesy.

Druhým skriptem je *LandmarksProcessor.py*, který zpracuje všechny snímky v dané složce a uloží je do .csv souboru.



Obrázek 7: Ukázka datasetu písmene Y

4.3.2 Použití frameworku MediaPipe v Pythonu

Ukázka použití MediaPipe Solutions frameworku, konkrétně Hand Landmark Detection v jazyce Python viz zdrojový kód 1.

4.3.3 Trénování sítě

Pro žádanou klasifikaci souřadnic existuje spousta metod, v rámci této práce byly vyzkoušeny následující metody:

- MediaPipe Gesture recognition
- Neuronové sítě
- XGBoost
- TensorFlow Decision Forests – Gradient Boosted Trees.

```python
import mediapipe as mp
import cv2

# Načtení obrázku
image = cv2.imread(file_path)

hands = mp.solutions.hands.Hands(
    static_image_mode=STATIC_IMAGE_MODE,
    max_num_hands=MAX_NUM_HANDS,
    min_detection_confidence=MIN_DETECTION_CONFIDENCE,
    min_tracking_confidence=MIN_TRACKING_CONFIDENCE,
)

result = hands.process(image)
```

Zdrojový kód 1: Použití MediaPipe Solutions v Pythonu

MediaPipe Gesture recognition nabízí možnost úpravy a rozšíření jejich modelu pro rozpoznávání gest, který pracuje mírně odlišným způsobem při zpracování souřadnic oproti ostatním vyzkoušeným přístupům. Nejprve z extrahovaných souřadnic vytvoří 128 prvkový vektor pomocí neuronové sítě a následně daný výsledek embeddingu klasifikuje další neuronovou sítí. Tato metoda měla mírně vyšší přesnost než vlastní implementace neuronových sítí se zmíněným zpracováním souřadnic z podkapitoly 4.2.2.

Ve finální aplikaci je použita metoda *Gradient Boosted Trees* z knihovny TensorFlow Decision Forests. Oproti neuronovým sítím měla tato metoda výrazně větší přesnost. Metoda XGBoost byla mírně přesnější než TFDF Gradient Boosted Trees, ale ve finální aplikaci nebyla použita z důvodu náročnosti integrace do webové aplikace.

K trénování modelu slouží skript *TrainerDF.py*, který rozdělí vstupní data z .csv souboru do trénovací a testovací množiny. Následně nad trénovací množinou spustí trénovací algoritmus poskytnutý knihovnou TensorFlow Decision Forests a následně natrénovaný model vyhodnotí a uloží do formátu, který je využíván knihovnou TensorFlow v JavaScriptu.

4.3.4 Použití TensorFlow v Pythonu

Ukázka použití TensorFlow knihovny v jazyce Python. Následující zdrojový kód 2 ukazuje příklad trénování *Gradient Boosted Trees* modelu.

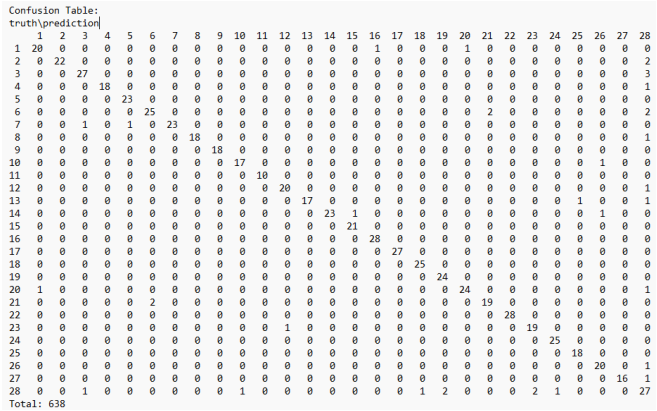
4.3.5 Přesnost modelu

Natrénovaný model má přesnost ∼94 % na testovací množině dat. Při testování neuronových sítí se přesnost pohybovala zhruba o ∼10 % níže. Důležitým faktorem pro reálné použití modelu je přesnost pro každý znak zvlášť. Pokud by model vyhodnocoval všechny znaky kromě jednoho správně a ten jeden zcela špatně,

```python
import tensorflow_decision_forests as tfdf

# Příprava rozdělených dat
train_ds = tfdf.keras.pd_dataframe_to_tf_dataset(train_set, label="label")
test_ds = tfdf.keras.pd_dataframe_to_tf_dataset(test_set, label="label")

# Vytvoření klasifikátoru
model = tfdf.keras.GradientBoostedTreesModel(verbose=2)
model.fit(train_ds)

# Evaluace modelu
model.compile(metrics=["accuracy"])
evaluation = model.evaluate(test_ds, return_dict=True)

# Uložení modelu
model.save(OUTPUT_TFDF_MODEL_PATH)
tfjs.converters.tf_saved_model_conversion_v2.convert_tf_saved_model(OUTPUT_TFDF_MODEL_PATH, OUTPUT_TFJS_MODEL_PATH)
```

Zdrojový kód 2: Použití TensorFlow Decision Forests v Pythonu

tak by byl takový model v aplikaci nepoužitelný. Pro takové vyhodnocení lze využít například matici záměn. Tabulku reprezentující takovou matici lze vidět na obrázku 8, která obsahuje hodnoty použitého modelu.



Confusion Table:
truth\prediction

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
2	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
4	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	2
7	0	0	1	0	1	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	1	0	0	1	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	23	1	0	0	0	0	0	0	0	0	0	1	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	
20	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	1	
21	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	
23	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	1
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	1
28	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	2	0	0	0	2	1	0	0	0	27

Total: 638

Obrázek 8: Matice záměn natrénovaného modelu

Výsledná přesnost je mírně nižší (zhruba o ∼3 %) oproti článku [14], který prezentuje GBT a MediaPipe Solutions pro rozpoznání prstové abecedy ASL na zhruba 100× větším datasetu. Menší přesnost je očekávaná na základě jednodušší práce se zpracováním souřadnic. Pro účely aplikace je natrénovaný model dostačující, ale nabízí se zde určitá vylepšení. Například ve zmíněné sofistikovanější práci se souřadnicemi ruky nebo využitím zcela jiného klasifikačního modelu.

Použitá metoda byla zvolena zejména na základě její jednoduchosti. Vytvořený model naplňuje očekávání pro funkcionalitu aplikace, jeví se jako schopný rozeznávat správně znaky a tím kontrolovat uživatele. Model je také dostatečný pro převod videozáznamu na souvislý text.

Je ale třeba podotknout, že model byl vytvořen na základě omezeného datasetu. Dataset obsahuje pouze dvě osoby a tím může negativně přispět ke schopnosti generalizace daného modelu. Model se mohl naučit specifika těchto dvou osob a tím značně omezit schopnost rozeznávat gesta ruky osob jiných.

4.3.6 Převod modelu na tfjs

K použití TensorFlow modelu na straně webového prohlížeče je nutné převést daný model do formátu, který je používán knihovnou TensorFlow.js. Ukázkový převod lze vidět ve zdrojovém kódu 2 na posledním řádku.

4.4 Problém určení času zápisu

Výše popsanými metodami lze tedy rozpoznat znak z jednoho snímku. K plynulému rozpoznání slov či vět je potřeba nějakým způsobem určit, kdy je znak zobrazen a kdy jej zaznamenat. S tímto problémem se lze často setkat v doméně rozpoznání znakové řeči pod zkratkou *CSLR* – Continuous sign language recognition. Tento problém velmi komplikuje nedokonalost klasifikátoru, která místo reálně zobrazovaného znaku může rozeznat znak jiný.

4.4.1 Možná řešení

Možné přístupy k řešení tohoto problému jsou:

- Získání potvrzení ze třetí strany, například stiskem klávesy na klávesnici či zobrazením speciálního gesta signalizujícího zápis minulého znaku nebo mezeru.
- Časová prodleva – daný znak se považuje za zapsaný, pokud jej uživatel zobrazí po určitou dobu.
- Detekce pohybu ruky, změny znaku, například při výrazné a rychlé změně pixelů obrazu.

4.4.2 Popis použitého algoritmu

Výsledná aplikace implementuje pro přepis sekvence znakování na text jednoduchý heuristický přístup. Tento přístup s sebou nese některá omezení. Vyžaduje relativně stabilní rychlost znakování a specifické akce uživatele pro vložení mezery nebo pro zapsání zdvojených písmen.

Algoritmus pracuje s řetězcem znaků, který vzniká postupným přidáváním vždy nejpravděpodobnějšího znaku rozpoznaného v jednotlivých snímcích z webkamery s určitou frekvencí. Tato frekvence je určena konstantou *msToNextPredict* v komponentě *GestureRecognition* a zásadně ovlivňuje očekávanou rychlost znakování uživatele. Samotný převod zmíněného řetězce všech rozpoznání na výsledný text zajišťuje funkce *convertRecognitionsToText(input)* implementovaná v souboru *CSLR.ts*.

Základem algoritmu je zpracování vstupního řetězce a jeho převedení na výstupní text na základě frekvence výskytu znaků a předdefinovaných prahových hodnot.

- *MIN_GROUP_LENGTH* (výchozí hodnota 6):
  Minimální počet po sobě jdoucích (nebo téměř po sobě jdoucích) výskytů stejného znaku, aby byl považován za platně zobrazený.
- *SPACE_INSERT_THRESHOLD* (výchozí hodnota 20):
  Práh počtu výskytů, při jehož překročení se za zapsaný znak přidá mezera.
- *ERROR_CONTEXT_SIZE* (výchozí hodnota 1):
  Počet znaků kolem potenciální chyby v rozpoznávání, která se kontroluje pro možnost ignorování této chyby.
- *MULTI_CHAR_TOKENS* (obsahuje 'Ch'):
  Definuje vícepísmenné znaky, které se mají zpracovávat jako jeden celek.

Těchto pět konstant nepřímo určuje toleranci algoritmu a očekávanou rychlost znakování. Výchozí nastavení je cíleno na začátečníky s prstovou abecedou, což znamená pomalejší tempo, než jakým by znakoval zkušený uživatel. Možným vylepšením by mohla být dynamická úprava těchto konstant podle rychlosti uživatele.

Konkrétní postup funkce *convertRecognitionsToText*:

1. Tokenizace: Vstupní řetězec je nejprve rozdělen na jednotlivé tokeny – znaky, přičemž se bere ohled na vícepísmenné znaky definované v konstantě.
2. Iterace: Algoritmus prochází sekvenci tokenů a počítá, kolikrát se aktuálně sledovaný token opakuje za sebou.
3. Tolerance chyb: Při kontrole opakování se uplatňuje tolerance vůči chybám. Pokud se objeví odlišný token, ale token před ním i za ním vzdálen o *ERROR_CONTEXT_SIZE* pozic odpovídá aktuálně sledovanému tokenu, je daný token ignorován a sekvence se považuje za nepřerušenou.
4. Registrace znaku: Jakmile sekvence aktuálního tokenu skončí a objeví se jiný token, který nelze ignorovat dle pravidla tolerance chyb, nebo nastane konec vstupu. Tak algoritmus zkontroluje počet nasbíraných výskytů *count*.

- Pokud *count* dosáhne alespoň *MIN_GROUP_LENGTH* (6), token je považován za platný znak a přidá se do výsledného textu.
- Výjimka: Pro tokeny 'g' a 'f' stačí poloviční počet výskytů.
- Vložení mezery: Pokud *count* dosáhl nejen prahu pro registraci znaku, ale i vyššího prahu *SPACE_INSERT_THRESHOLD* (20), je stav interpretován jako signál konce slova a za právě přidaný znak se do výsledku vloží mezera.

5. Pokračování: Pokud vstupní sekvence ještě neskončila, nastaví se nový aktuální token a proces pokračuje další iterací v kroku 2.

Znaky 'g' a 'f' vyžadují pouze polovinu rozpoznání, protože jsou častěji zaměňovány s jinými znaky než ostatní písmena. Písmeno 'f' je zaměňováno s písmenem 't'. Písmeno 'g' je zaměňováno s písmeny 'c' a 'o'. K těmto jednostranným záměnám dochází z důvodu nedokonalosti rozpoznávání a velké podobnosti zmíněných znaků písmen. Zmíněné podobnosti ale nejsou jedinými a i ostatní znaky písmen mají často velmi podobné znaky.

4.4.3 Příklad vstupu a výstupu

Z rozpoznaného řetězce písmen:
MMMMMMMMMMMMAAASAAAAAAAAMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMNZZKKKKKKKKKKKChOOOOOOCCCCCCC
CCCCDChKKKKKKKKKKKKKKChChUUUUUUUUUUUUUUUUUUUUU
UUUSSSSSSSSSSSSSSSSSSSSSEOEEEEEEEERRRRRRRRRRRRAAAAACh
RRRRRRRRRRRROOOOOOOOOHRRRRRRRRRRRRREEEEEEEEEEEOO
MMMMMMMMMMMMM
zmíněný algoritmus vyhodnotí skutečnou sekvenci znaků na skutečně zobrazené slovní spojení: *MAM KOCKU S ERROREM*.

V příkladě lze vidět, že vstup obsahuje necílené znaky při změnách skutečně zobrazovaných znaků (např. změna znaku z písmene 'm' na písmeno 'k' – MMMMMMNZZKKKK). Tyto krátké sekvence nejsou ale rozpoznávány.

Také si lze všimnout ignorování osamělých znaků v rámci některé sekvence (např. zobrazené písmeno 'a' bylo nechtěně rozpoznáno jako písmeno 's' – AAASAAA).

Algoritmus vyhodnocuje zdvojená písmena krátkým přerušením sekvence rozpoznání stejného znaku (např. rozpoznaného vstupu při znakování sekvence 'rr' – ...RRRRAAAAChRRRRR...). Uživatel v takový moment musí krátce zobrazit jiný znak – přerušit ukazování zdvojeného znaku a tím dojde k rozpoznání jiného znaku.

U posledních písmen ve slově lze vidět podstatně delší sekvenci rozpoznání.

4.4.4 Propojení s LLM

Proces rozpoznání záznamu kamery na souvislý text je náročný a zcela určitě bude docházet k chybám. Tím se nabízí problém, jak detekovat tyto chyby a jak

je opravit.

Jedno z tradičních řešení je například vytvoření databáze všech možných slov, kde by algoritmus měl za cíl pro slovo s chybou najít takové slovo, kterému se nejvíce podobá. Jak už bylo zmíněno, některé znaky jsou zaměňovány častěji než jiné, tedy do tohoto algoritmu by bylo možné zakomponovat výběr chtěného slova na základě toho, s jakou pravděpodobností byl chybně rozpoznaný znak zaměněn za jiný.

Implementace takového algoritmu ale není rozhodně přímočará a zcela nefunkční pro slova, která by nebyla v databázi slov. Zajímavým zamyšlením také je, že pravděpodobnost výskytu slova ve větě je provázána s okolními slovy v dané větě a i v okolních větách – tedy s kontextem celého textu. Tato úvaha, která v posledních pár letech zcela změnila svět, mě přivedla na použití řešení tohoto problému pomocí velkých jazykových modelů.

Výstup popsaného algoritmu *CSLR* je možné provázat se vstupem velkého jazykového modelu. Vlastností takových modelů je porozumění kontextu slov v textovém vstupu a tím vzniklá schopnost zanedbání drobných chyb či překlepů takového vstupu.

Zmíněné provázání *CSLR* algoritmu s velkým jazykovým modelem je využito v režimu *Konverzace*, který je popsán v podkapitole 6.6.

5 Webová aplikace

Spojením výše popsaných dvou částí vznikla webová aplikace, jejímž cílem je podpora výuky a procvičování české jednoruční prstové abecedy. Aplikace rozšiřuje své funkcionality i pro dvouruční českou prstovou abecedu a nabízí podporu procvičování znaků českého znakového jazyka.

5.1 Použité technologie

V této sekci jsou popsány technologie, na kterých je založena vyvinutá webová aplikace.

5.1.1 SvelteKit

SvelteKit je moderní framework pro vývoj webových aplikací. SvelteKit je založen na UI frameworku Svelte, který umožňuje jednoduchou tvorbu komponent, které jsou překládány do JavaScriptu a efektivně vykreslovány v prohlížeči. [15]

5.1.2 Flowbite Svelte

Flowbite Svelte je open-source knihovna UI komponent pro Svelte. Poskytuje sadu předpřipravených komponent, které urychlují vývoj uživatelského rozhraní. Komponenty jsou optimalizované pro responzivní design a umožňují rychlé vytváření konzistentního vzhledu aplikace. [16]

5.1.3 Tailwind CSS

Tailwind CSS je utility-first CSS framework, který umožňuje rychlé a flexibilní stylování webových aplikací. Místo předpřipravených komponent používá nízkoúrovňové utility třídy. [17]

5.1.4 Three.js a Threlte

Three.js je populární knihovna pro vytváření a manipulaci s 3D grafikou v prostředí webových prohlížečů pomocí JavaScriptu. Umožňuje snadné vytváření složitých 3D scén, animací a interakcí. [18] Threlte je nadstavba nad Three.js, která integruje tuto knihovnu do frameworku Svelte. Threlte poskytuje komponenty a nástroje, které zjednodušují vytváření a správu 3D scén a animací ve Svelte aplikacích. Threlte je použito ve vytvořené aplikaci pro zobrazení modelu postavy a jeho animací. [19]

5.1.5 TensorFlow.js

TensorFlow.js je knihovna pro strojové učení v JavaScriptu, která umožňuje spouštění natrénovaných modelů ve webovém prohlížeči.

5.2 Architektura aplikace

Architektura webové aplikace je založena na komponentách, kde komponenta je ucelená nezávislá funkcionalita. Tyto komponenty lze vkládat do jiných komponent a tím vytvořit komplexní strukturu, ve které je přehledně rozdělena zodpovědnost a závislost pomocí domluvených rozhraní.

Hlavními prvky aplikace jsou stránky, mezi kterými se uživatel přepíná pomocí směrování. Tyto stránky definují vlastní obsah, který se většinou skládá z kombinace zmíněných komponent. Stránka je zodpovědná za komunikaci mezi komponentami umístěnými uvnitř ní.

5.2.1 Struktura projektu

Struktura celého projektu se skládá z následujících adresářů.

- */*

  Kořenový adresář obsahující konfigurační soubory a soubor *package.json* definující závislosti webové aplikace.

- */src*

  Adresář obsahující zdrojové kódy webové aplikace. Obsahuje dva podadresáře a soubory, které obalují celou aplikaci.

  - */src/lib*

    Adresář obsahující všechny komponenty aplikace a soubory s modely a pomocnými funkcemi pro práci s textem, inferencí AI modelů, zpracováním souřadnic a CSLR vyhodnocením.

  - */src/routes*

    Adresář obsahující soubory jednotlivých stránek a soubory definující celkový layout aplikace. Adresářová struktura tohoto adresáře definuje směrování aplikace.

- */static*

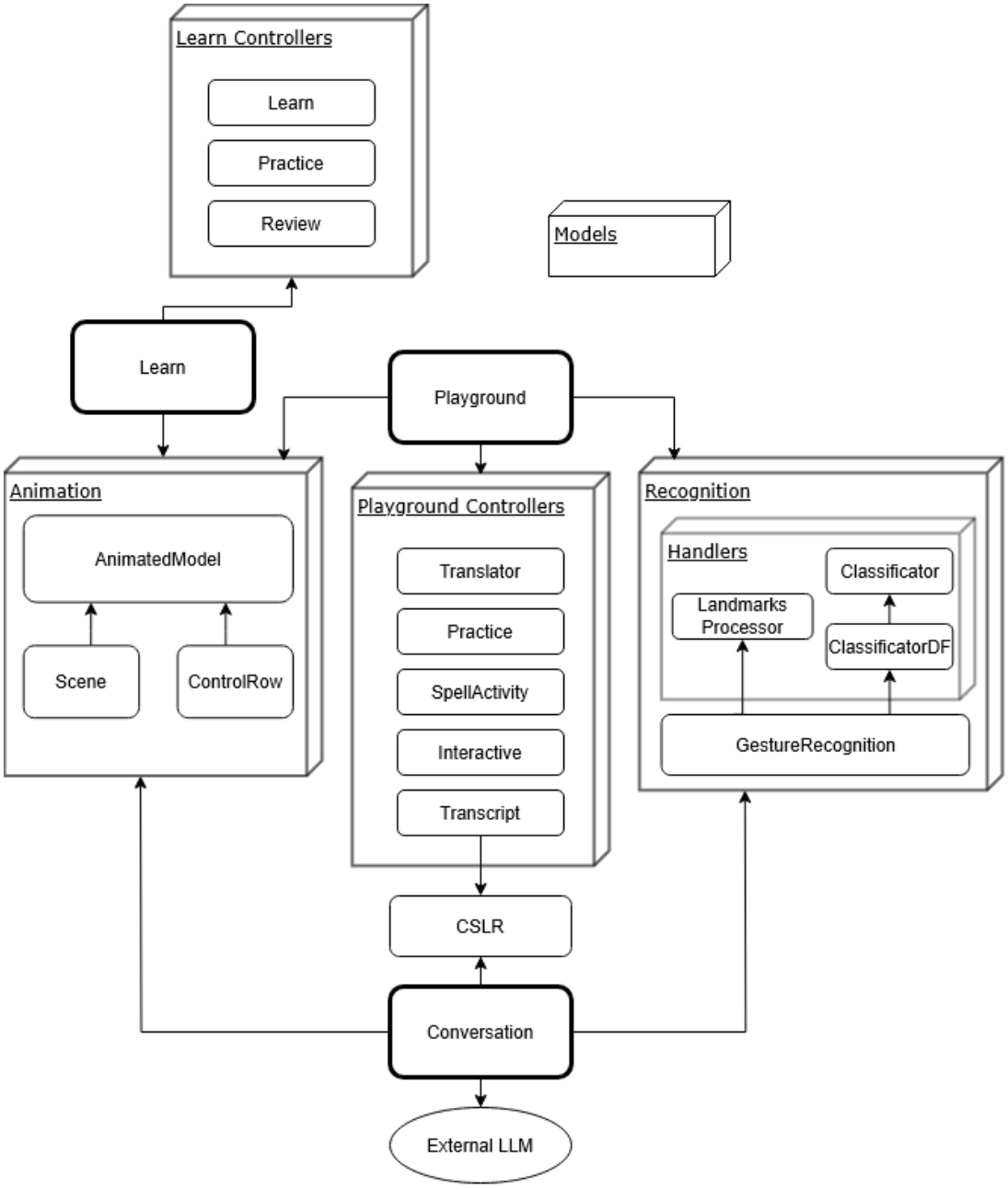
  Adresář obsahuje všechny statické soubory webové aplikace. Nachází se zde ikony, manifest, AI modely, 3D model, seznam slov a wasm soubory.

- */3D*

  Adresář s 3D modelem.

- */python_recognition*

  Adresář obsahující dataset a zdrojové kódy použité při tvorbě klasifikačního modelu.



Obrázek 9: Diagram architektury celé aplikace

5.2.2 Diagram aplikace

K přehledu provázanosti komponent a přehledu architektury celé aplikace slouží následující diagram, viz obrázek 9.

5.2.3 Podrobnější popis hlavních komponent a architektury

V aplikaci se nachází dvě hlavní komponenty.

- AnimatedModel

  Komponenta *AnimatedModel* obsahuje logiku pro vykreslování 3D modelu a pro řízení a přehrávání animací. Tato komponenta se používá v další komponentě *Scene*, která definuje scénu, do které je model vykreslen. Ostatní stránky nebo komponenty poté mohou využívat komponentu *Scene* pro snadné vložení 3D scény s modelem. *Scene* poskytuje ostatním komponentám instanci *AnimatedModel*, kde lze pomocí definovaného rozhraní řídit chování modelu. Stěžejní funkcí komponenty *AnimatedModel* je *playAnimationForText(text, language)*, která pro zadaný text spustí sekvenci animací, znakující daný text v zadaném jazyku.

  Pomocí další komponenty *ControlRow* lze přidat možnost ovládání animací. Tato komponenta umožňuje volbu rychlosti přehrávání animací a možnost pozastavení celé animace.

- GestureRecognition

  Druhou významnou komponentou je *GestureRecognition*. Tato komponenta je zodpovědná za zobrazení záznamu webkamery, rozeznání souřadnic ruky pomocí MediaPipe frameworku, zpracování rozpoznaných souřadnic a klasifikaci gest. Při použití této komponenty se lze přihlásit k odběru zpráv, které obsahují pravděpodobnosti právě vyobrazeného znaku.

  Logika zpracování rozpoznaných souřadnic ruky se nachází v souboru *LandmarksProcessor.ts*. Klasifikátor *ClassificatorDF.ts* implementuje rozhraní *Classificator.ts* a je zodpovědný za klasifikaci zpracovaných souřadnic danou AI metodou.

Na stránce *Cvičiště* je poté mezi tyto dvě komponenty vložena komponenta třetí – Controller, řídící logiku zvoleného režimu. Každý režim má vlastní řídící komponentu, která zpracovává vstup od uživatele a komunikuje s *AnimatedModel* a *GestureRecognition*. Stránka poskytuje Controller komponentám instanci komponenty *AnimatedModel* pro umožnění řízení animací a přeposílá jim zprávy o rozpoznaných gestech.

Stránka *Učitel* je vytvořena obdobně. Stránka zobrazující animovaný model umožňuje přepínat mezi třemi Controller komponentami, které zpracovávají vstup uživatele a komunikují s poskytnutou instancí *AnimatedModel*.

Stránka *Konverzace* je založena také na zmíněných hlavních komponentách. Tento režim je vytvořen na jedné stránce, která vykresluje uživatelské rozhraní

a řídí komunikaci mezi komponentami, vstupem uživatele a rozhraním LLM modelů.

Některé komponenty využívají stejné funkce jako ostatní. Tyto funkce jsou extrahovány do pomocných TypeScript souborů za účelem omezení duplikace stejného kódu a oddělení zodpovědnosti. Zajímavou funkci lze například vidět v souboru *CSLR.ts*, jejíž účel je detailně popsán v podkapitole 4.4.

5.3 Přehrávání animací a zobrazení modelu v Threlte

Tato podkapitola popisuje použití vytvořeného 3D modelu člověka, jeho vykreslení, animování a řízení průběhu animace na webové stránce, konkrétně pomocí technologie Threlte.

5.3.1 Scéna

K vyobrazení modelu je potřeba nejprve vytvořit scénu, ve které se bude model nacházet. Tato scéna se skládá z osvětlení, kamery a pozadí. Ve zdrojovém kódu se nachází komponenta *Scene*, která definuje právě tuto scénu. Značkou *<Canvas>* se definuje oblast, ve které se bude 3D grafika vykreslovat. Uvnitř oblasti se nachází *<T.PerspectiveCamera>* a *<T.AmbientLight>*, které definují kameru a osvětlení scény. Tato komponenta také obsahuje komponentu *<T.OrbitControls>*, která umožňuje uživateli ovládat pohled kamery pomocí myši. Posledním objektem uvnitř scény je samotný model, který je zobrazen pomocí Svelte komponenty *AnimatedModel*.

5.3.2 Vykreslení modelu

Komponenta *AnimatedModel* obsahuje definici vykreslení modelu postavy a algoritmus řídící přehrávání animací. Její základ je vygenerován na základě poskytnutého 3D modelu pomocí nástroje *@threlte/gltf*. Obsahuje několik funkcí, kterými ji lze ovládat. Nejdůležitější funkcí je *playAnimationForText(text, language)*, která spustí řízení sekvence animací odpovídající zadanému textu a jazyku.

K načtení modelu nabízí Threlte funkci *useGltf(model, options)*, pomocí které lze načíst model. V options lze například nastavit využití Draco komprese.

5.3.3 Ovládání animací

K animacím modelu se v Threlte přistupuje pomocí actions a mixer, které lze získat pomocí funkce *useGltfAnimation(gltf, ref)*, kde *gltf* je právě načtený model pomocí funkce *useGltf*. Slovník *actions* obsahuje všechny dostupné animace, ke kterým lze přistupovat pomocí klíče odpovídajícího názvu animace uvnitř 3D modelu. Animaci lze spustit pomocí nalezení v kolekci *actions* a zavolání funkce *play()* nad danou instancí.

K určení bodu začátku následující animace pro nadcházející znak je potřeba detekovat konec aktuální animace. Instance *mixer* umožňuje odběr události, která je volána při ukončení animace.

K plynulému přechodu mezi animacemi slouží funkce *transitionTo(action, duration)*. Tato funkce nastaví následující animaci a nad aktuálně přehrávanou animací zavolá funkci *crossFadeTo(action, duration)*, která plynule přejde z jedné animace do druhé.

5.3.4 Ukázka kódu frameworku Threlte

Vytvoření scény, ve které se nachází 3D model postavy, lze vidět ve zdrojovém kódu 3.

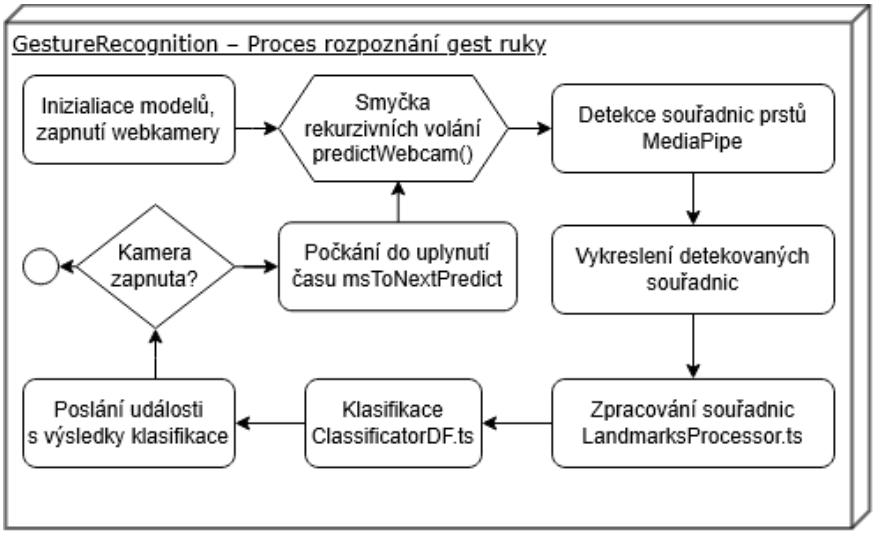
```
1  const gltf = useGltf('models/Model.glb', {useDraco: true})
2
3  // Přehrávání animací
4  const { actions, mixer } = useGltfAnimations(gltf)
5
6  // Spustí animaci, když se změní hodnota currentActionName
7  $: $actions[currentActionName]?.play()
8
9  // Změna rychlosti přehrávané animace
10 mixer.timeScale = speed;
11
12 <Canvas>
13    <T.PerspectiveCamera makeDefault position={[0, 0, 0.8]} fov={60}>
14       <OrbitControls autoRotate={false} enableZoom={true}
             enableDamping={false} />
15    </T.PerspectiveCamera>
16
17    <T.AmbientLight intensity={5} />
18
19    <T is={\$gltf.scene} position = {[0, -1.5, 0]} />
20 </Canvas>
```

Zdrojový kód 3: Ukázka Threlte scény s 3D modelem, kamerou a osvětlením

5.4 Rozpoznání gest ruky

K rozpoznání gesta ruky na webové stránce je potřeba získat snímek z webkamery. Na takovém snímku poté provést stejné rozpoznání a zpracování souřadnic ruky jako při trénování z datasetu a následně výsledek klasifikovat pomocí vytvořeného klasifikátoru.

Celý proces rozpoznání je zobrazen na následujícím diagramu znázorněném na obrázku 10.



Obrázek 10: Diagram procesu rozpoznání gest ruky

5.4.1 Použití MediaPipe Solutions v JavaScriptu

V následujícím zdrojovém kódu 4 lze vidět ukázku použití MediaPipe Solutions v JavaScriptu. Funkce *detectForVideo* rozpoznává souřadnice prstů ruky a vrací 21 souřadnic při úspěšném rozpoznání.

```
1  import {HandLandmarker, FilesetResolver} from '@mediapipe/tasks-
       vision';
2
3  const vision = await FilesetResolver.forVisionTasks('wasm-taskvision
       ');
4
5  handLandmarker = await HandLandmarker.createFromOptions(vision, {
6     baseOptions: {
7        modelAssetPath: `hand_landmarker.task`,
8        delegate: 'GPU'
9     },
10    runningMode: 'VIDEO',
11    numHands: 1
12 });
13
14 const results: HandLandmarkerResult = handLandmarker.detectForVideo(
       video, startTimeMs);
```

Zdrojový kód 4: Ukázka MediaPipe Solutions v JavaScriptu

5.4.2 Použití TensorFlow Decision Forests modelu v JavaScriptu

Pomocí zmíněné knihovny TensorFlow.js se nejprve načte model klasifikátoru. Poté je nad instancí tohoto modelu volána funkce *executeAsync*, jejímž vstupem jsou zpracované souřadnice ruky a výstupem jsou pravděpodobnosti každého znaku na základě vstupu. Ukázku použití lze vidět ve zdrojovém kódu 5.

```
1 import * as tf from '@tensorflow/tfjs-core';
2 import * as tfdf from '@tensorflow/tfjs-tfdf';
3 tfdf.setLocateFile(`inference.wasm`);
4
5 // Načtení modelu
6 model = await tfdf.loadTFDFModel("model.json");
7
8 // Klasifikace dat, kde tensor obsahuje zpracované souřadnice
9 const output = (await model.executeAsync(tensor)) as tf.Tensor;
```

Zdrojový kód 5: Ukázka TensorFlow.js

5.5 Rozhraní a výkon aplikace

Rozhraní aplikace je responzivní, aplikaci lze používat jak na mobilním zařízení, tak i na větším monitoru. Pro komfortní použití aplikace je potřeba stále vidět animaci, záznam kamery i ovládací prvky, a proto doporučuji používat aplikaci na zařízení s větší obrazovkou.

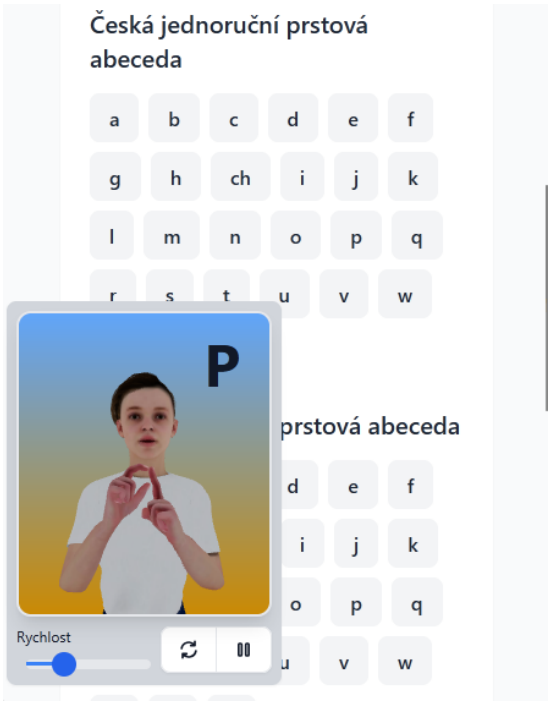
Zobrazení na menších obrazovkách využívá netradiční řešení. Místo zmenšení nebo změny uspořádání prvků aplikace využívá fixní pozicování. V režimu *Konverzace* je v levém horním rohu okno webkamery a v pravém dolním rohu tlačítko pro odesílání zprávy. V režimu *Učitel* je animace v levém dolním rohu obrazovky. Zmíněné prvky jsou ve fixních pozicích stále, nezávisle na aktuálně zobrazené části stránky. Tímto přístupem lze stále vidět podstatnou část stránky a při tom využívat ovládací prvky, které by jinak na menších obrazovkách nebyly současně vidět. Obrázek 11 zobrazuje vzhled aplikace na mobilním zařízení.

Aplikace, obzvlášť v režimech s animací a rozpoznáním zároveň, je znatelně náročná na systémové požadavky zařízení. Na průměrně výkonných telefonech může být přehrávání animací viditelně neplynulé, a proto doporučuji aplikaci používat na počítačích. Na běžně výkonném počítači je aplikace zcela plynulá.

5.6 Nasazení aplikace

Aplikace je vytvořena pro webový server node.js. Pro jednoduchost ale obsahuje Dockerfile pro spuštění v Docker kontejneru.

Při nasazení mimo lokální síť je potřeba webovou aplikaci provozovat přes protokol https, prohlížeče v takových situacích z bezpečnostních důvodů blokují přístup k webkameře.



Obrázek 11: Webová aplikace na zařízení s malou obrazovkou

Pro správnou funkčnost aplikace v PWA režimu je potřeba, aby webový server měl validní SSL certifikát.

6 Uživatelská příručka webové aplikace

Tato kapitola popisuje jednotlivé funkční části aplikace z hlediska jejich funkčnosti a používání.

6.1 Obecné ovládání

V následujících dvou podkapitolách je vysvětleno, jak uživatel může a měl by přistupovat k ovládání a používání aplikace.

6.2 Ovládání kamery animace

Úhel přiblížení nebo pozici kamery lze v animačním okně s postavou ovládat. Držením levého tlačítka myši lze ovládat rotaci kamery. Pomocí pravého tlačítka lze pohybovat celou kamerou. Použitím kolečka lze měnit přiblížení.

Rychlost přehrávání animace lze měnit posuvníkem pod oknem animace.

6.3 Ovládání rozpoznání prstové abecedy z kamery

Ke správnému rozpoznávání znaků z kamery je dobré vědět, jak aplikace uživatelovo znakování zaznamenává a podle toho se přizpůsobit očekávanému chování uživatele ze strany aplikace.

6.3.1 Rozpoznání jednotlivých znaků

V režimech, kde aplikace očekává jen jeden konkrétní znak, je použití intuitivní. Stačí daný znak zřetelně ukázat před kamerou a aplikace jej zaznamená. Tento způsob chování se například nachází v režimu *Cvičiště – Znakování*, kde aplikace čeká na zobrazení následujícího znaku zatím nerozpoznané části slova.

6.3.2 Rozpoznání souvislého textu

Tyto režimy jsou náročnější, protože se snaží přepsat plynulou sekvenci znaků do navazující sekvence slov. Tento režim se nachází na stránce *Cvičiště* v režimu *Přepis* a na stránce *Konverzace*. Pro úspěch je klíčové:

- Sledovat zpětnou vazbu

  Vždy věnujte pozornost textovému poli, kde se zobrazuje rozpoznaný text. Aplikace potřebuje, abyste každý znak chvíli podrželi, než ho zaregistruje. Nepřecházejte na další písmeno, dokud neuvidíte, že se to aktuální objevilo v textovém poli.

- Vytváření mezer

  Chcete-li ukončit slovo a začít nové (vložit mezeru), musíte znak posledního písmena slova podržet znatelně déle než ostatní znaky. Aplikace delší zobrazení znaku interpretuje jako signál pro vložení mezery.

- Znakování stejného písmene vícekrát za sebou

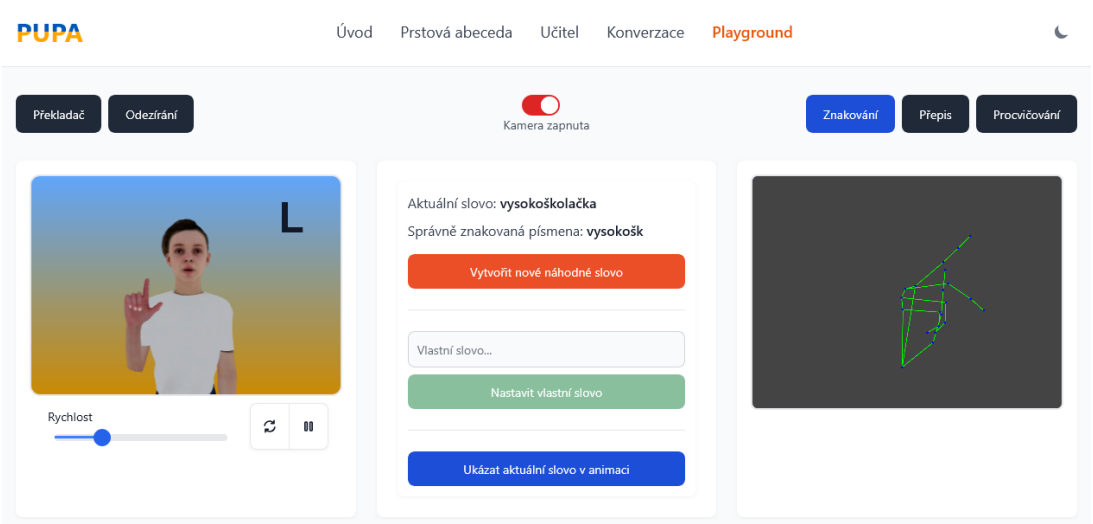
  Tato situace vyžaduje specifický postup, protože aplikace nezaregistruje stejné písmeno dvakrát, při delším podržení daného znaku.

  – Ukažte znak písmena a počkejte, až jej aplikace zapíše (uvidíte písmeno v textovém poli).
  – Okamžitě poté, co se písmeno zapíše, krátce změňte gesto ruky. Nejjednodušší je ruku na chvíli uvolnit do neutrální polohy.
  – Poté znovu ukažte stejné písmeno, které chcete zapsat podruhé.

  Pokud uživatel neuvolní ruku mezi stejným znakem dostatečně rychle, dojde k rozeznání mezery. Pokud uživatel uvolní ruku na příliš dlouho, může dojít k zapsání jiného znaku.

6.4 Cvičiště

Stránka *Cvičiště* slouží jako prostředí, ve kterém si uživatel může vyzkoušet různé režimy ve volnější formě než u ostatních stránek. Uživatel má na výběr z pěti režimů. Režimy *Překladač* a *Odezírání* nevyžadují použití webkamery. Zbývající tři režimy *Znakování*, *Přepis* a *Procvičování* vyžadují pro svou funkčnost záznam webkamery. Vzhled stránky lze vidět na obrázku 12.



Obrázek 12: Webová aplikace – Stránka Cvičiště

6.4.1 Překladač

Překladač slouží k překladu textu do sekvence animací odpovídající zadanému textu a jazyku. Uživatel si může vybrat z jednoruční nebo dvouruční prstové abecedy a českého znakového jazyka. Volba znakového jazyka přehraje znak slova,

pokud pro něj existuje animace. Pokud pro daný znak animace neexistuje, model zobrazí dané slovo v jednoruční prstové abecedě.

6.4.2 Odezírání

Odezírání je režim, ve kterém je uživateli ukázáno náhodné slovo v podobě animace a uživatel má za úkol dané slovo rozpoznat a zapsat. Aplikace poté zkontroluje, zda zadané slovo odpovídá náhodně zvolenému slovu.

Aplikace obsahuje zhruba 23 tisíc náhodných českých slov [20]. Při volbě znakového jazyka se slova generují stále stejným způsobem a je tedy pravděpodobné, že pro dané slovo nebude existovat animace. Pro procvičování znakového jazyka pouze s existujícími animacemi může uživatel využít procvičení na stránce *Učitel*.

6.4.3 Znakování

Tento režim slouží pro procvičení znakování. Uživatel si může vymyslet své vlastní slovo, nebo si jej může vygenerovat jako v režimu *Odezírání*. Rozdílem oproti předchozímu režimu ale je, že uživatel má nyní dané slovo ukázat před webkamerou. Webová aplikace přitom kontroluje, zda uživatel dané slovo znakuje správně. Pokud uživatel nebude vědět, jak dané slovo znakovat, může si slovo přehrát v animaci stiskem tlačítka.

6.4.4 Přepis

Režim *Přepis* má dvě volby – *Automatický* a *Manuální*. V automatickém režimu aplikace rozpoznává znaky, které uživatel ukazuje před webkamerou, a aplikace se snaží přepsat sekvenci rozpoznání do věty. V manuálním režimu namísto automatického zaznamenávání musí uživatel manuálně potvrdit zaznamenání aktuálně ukázaného znaku.

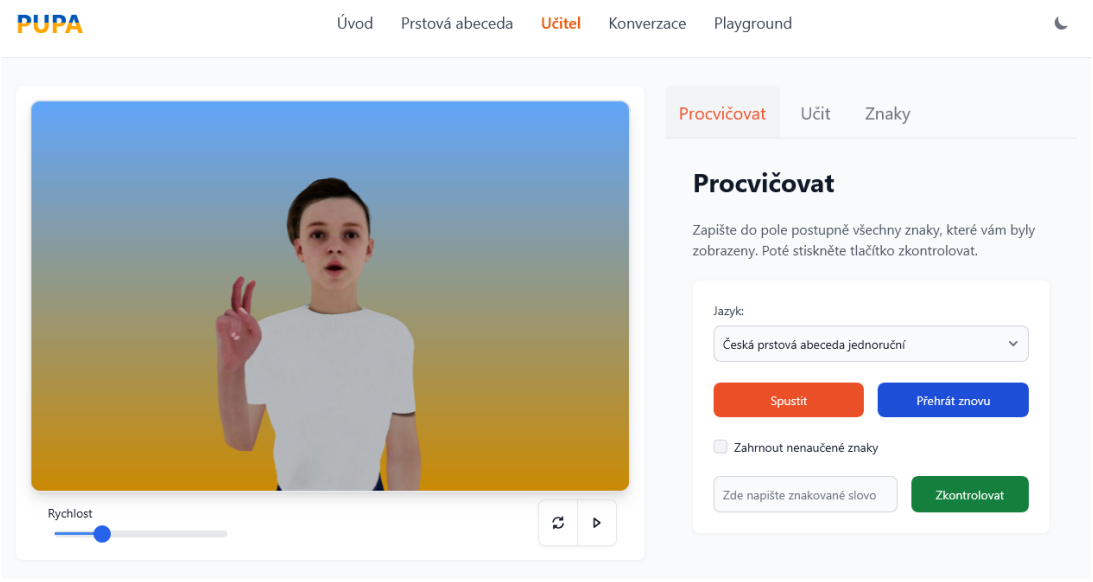
6.4.5 Procvičování

Procvičování na stránce *Cvičiště* je nejinteraktivnějším režimem. Uživatel si zde může vymyslet vlastní text, který poté střídavě znakuje spolu s animací.

Každý znak nejprve ukáže model v animaci a poté musí uživatel zobrazený znak zobrazit správně před webkamerou. Až po úspěšném zobrazení dochází k přechodu k dalšímu znaku. Cíl uživatele je takto projít celým textem.

6.5 Učitel

Stránka *Učitel* provádí uživatele postupně všemi znaky jazyka. Nabízí českou jednoruční a dvouruční prstovou abecedu a také český znakový jazyk. Tento režim lze rozdělit na tři části – *Procvičovat*, *Učit* a *Znaky*. Vzhled stránky lze vidět na obrázku 13.



Obrázek 13: Webová aplikace – Učitel

6.5.1 Procvičovat

Režim *Procvičovat* slouží k procvičování již naučených znaků. Aplikace náhodně vygeneruje jeden z naučených znaků a přehraje jej v rámci okna animace. Uživatel poté musí daný znak rozeznat a zapsat. Uživatel může svou odpověď zkontrolovat a případně daný znak přehrát znovu.

6.5.2 Učit

Učení nových znaků probíhá v režimu *Učit*. Zde je uživateli přehrán následující, zatím nenaučený, znak v animaci. Uživatel si může danou animaci přehrát vícekrát a rozhodnout se, že si přehrávaný znak již pamatuje a označit jej za naučený.

6.5.3 Znaky

V režimu *Znaky* lze sledovat aktuální postup již naučených znaků. Lze si zde všechny znaky přehrávat a také je lze zrychleně označit za již naučené dvojitým kliknutím.

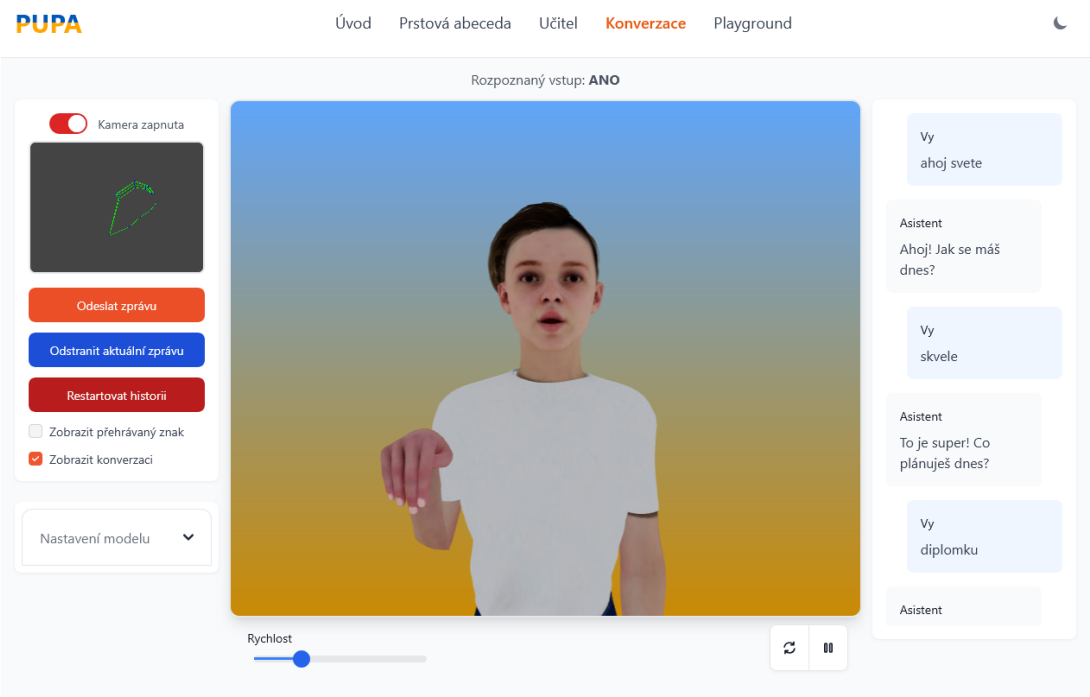
6.6 Konverzace

Stránka *Konverzace* poskytuje nejinteraktivnější možnost procvičování prstové abecedy. Stránka se skládá ze tří hlavních částí. Uprostřed obrazovky se nachází 3D model, se kterým uživatel komunikuje. V levé části lze vidět záznam webkamery a také ovládací prvky. V pravé části se nachází historie konverzace. Režim

je založen na konverzaci, kde uživatel komunikuje s modelem prostřednictvím české jednoruční prstové abecedy. Uživatel nejdříve vyznakuje sekvenci znaků, která je převedena do věty. Tato věta je odeslána nastavenému LLM modelu a odpověď tohoto modelu je uživateli prezentována v podobě animace.

Velkou výhodou použití velkých jazykových modelů je jejich schopnost pochopení textu s chybami. Díky této vlastnosti nemusí uživatel cílit na zcela správně rozpoznaný text a může pokračovat ve znakování dále, ačkoliv systém rozpoznal některý z předchozích znaků špatně, nebo se v něm uživatel spletl.

Rozhraní stránky *Konverzace* lze vidět na obrázku 14.



Obrázek 14: Webová aplikace – Konverzace

6.6.1 Nastavení LLM

Aplikace podporuje dvě rozhraní pro komunikaci s LLM modelem. Prvním z nich je *ollama* – populární nástroj pro spouštění jazykových modelů [21] a druhým rozhraním je *platforma OpenAI* [22]. Aplikace obsahuje pole pro nastavení komunikace mezi těmito rozhraními, avšak pokud by uživatel chtěl použít rozhraní jiné, nic mu nebrání ve vytvoření vlastního překladače dotazů mezi webovou aplikací a zvoleným modelem.

6.6.2 Ollama

Nejdříve je potřeba nainstalovat program ollama, nastavit CORS pro povolení cross-origin mezi webovou aplikací a programem ollama. Posledním krokem je instalace některého modelu a jeho spuštění.

Webová aplikace má již přednastavenou výchozí adresu programu a stačí pouze změnit název modelu na ten, který si uživatel nainstaloval a spustil.

Postup instalace ollama na vlastním počítači popisuje následující seznam kroků.

1. *Instalace programu ollama*

   Stáhnutí instalačního balíčku na adrese: *ollama.com/download* a jeho nainstalování.

2. *Povolení CORS*

   Nastavení proměnné prostředí *OLLAMA_ORIGINS = adresa_aplikace* a restartování programu ollama. Například v operačním systému Windows stačí v příkazové řádce zadat:
   'SET OLLAMA_ORIGINS = https://158.194.92.104'.

3. *Stažení modelu*

   Vybrání modelu na adrese: *ollama.com/search* a jeho stažení pomocí příkazu *ollama run název_modelu*. Před vybráním modelu je potřeba se zamyslet nad jeho velikostí. Velikost modelu by neměla přesahovat velikost volné operační paměti, případně velikost paměti grafické karty.

4. *Spuštění modelu*

   Již stažený model lze zapnout stejným příkazem:
   ollama run název_modelu.

5. *Ukončení modelu*

   Pro vypnutí modelu a uvolnění paměti stačí napsat do příkazové řádky s běžícím model příkaz */bye*.

6.6.3 OpenAI

Pro komunikaci přes rozhraní společnosti OpenAI je nejprve potřeba získat API klíč. Ten lze získat na platformě zmíněné společnosti, ke které lze přistoupit na adrese: *platform.openai.com*. Tento přístup je placený, ale značně ulehčuje zprovoznění režimu a také nabízí řádově rychlejší a větší modely, které nelze spustit na běžném počítači. Zmíněné větší modely poskytují přesnější odpovědi a díky tomu dokážou vést zajímavější a zábavnější konverzaci s uživatelem. Uživatel v nastavení aplikace pouze vloží svůj API klíč a název vybraného modelu a může režim plně využívat.

6.6.4 Klávesové zkratky

Uživatel může využívat klávesy *Mezerník* pro vytvoření mezery v aktuálním místě vstupu a také *Backspace* pro smazání posledně zaznamenaného znaku. Klávesa *Enter* odešle aktuální zprávu.

6.7 Používání aplikace bez internetu

Vytvořená webová aplikace je PWA aplikací – progressive web application. Umožňuje instalaci na uživatelovo zařízení a nainstalovanou aplikaci lze poté používat bez přístupu k internetu. Nainstalovanou aplikaci lze ale také samozřejmě používat i s připojením k internetu a využívat *OpenAI API* na stránce *Konverzace*.

V desktopovém prohlížeči lze aplikaci nainstalovat pomocí otevření webové aplikace v prohlížeči a stisknutí ikony zobrazené v pravém okraji adresního řádku prohlížeče.

Na mobilním zařízení lze stránku nainstalovat pomocí otevření stránky v prohlížeči a zvolení položky *Přidat na domovskou obrazovku*, která po stisknutí nabídne instalaci aplikace.

Instalace byla otestována a je funkční v prohlížeči *Google Chrome*. Nainstalovaná aplikace nabízí všechny funkcionality jako při používání ve webovém prohlížeči.

7 Rozšíření

V této kapitole budou probrána některá z možných a vhodných rozšíření aplikace. Tato rozšíření nejsou implementována. Zde je pouze zmíněn možný přístup k jejich řešení a zdůvodnění jejich významu.

7.1 Implementace pro znakový jazyk

Aplikace je lehce rozšířitelná tak, aby podporovala slova ze znakového jazyka, ale není v ní tento režim znatelně zastoupen z důvodu náročnosti tvorby animací.

7.1.1 Animace a snímání pohybu

Zajímavým přístupem, jak efektivně animovat 3D model člověka, je snímání pohybu (anglicky motion capture). Zjednodušeně si lze tuto metodu představit tak, že počítač pohybuje s 3D modelem člověka podle reálné předlohy. Problém nastává s přesností zaznamenávání pohybu. Motion capture je velmi využíván ve tvorbě filmů a her, kde herci využívají speciálních nástrojů, které dokážou snímat jejich pohyb velice přesně. Dostupnějším řešením je použití kamery, pomocí které lze rozpoznávat souřadnice specifických částí těla, podobně jako použitý model na rozpoznání souřadnic ruky v této práci. Volně dostupné programy převádějící vstup kamery na animace se ukázaly jako nedostatečně přesné.

7.1.2 Rozpoznání

Znaky slov jsou mnohem komplexnější oproti znakům prstové abecedy. Jejich rozpoznání by vyžadovalo použití rozpoznání prstů a postavy. Použitý framework MediaPipe obsahuje Holistic Landmarker, který kombinuje modely pro rozpoznání obličeje, postavy a prstů. Na základě těchto dat by bylo možné natrénovat model obdobně jako při rozpoznávání prstové abecedy. Nicméně takový model by nebyl dostatečný, jelikož mnoho znaků nemá statickou reprezentaci, ale vyžadují souvislý pohyb. Na tento problém lze narazit i u prstové abecedy.

7.2 Rozpoznání pohyblivých gest

Některé znaky jsou založeny nejen na statické pozici prstů, ale také na pohybu ruky. Příkladem může být písmeno J, jehož znak kombinuje pohyb ruky spolu s pozicí prstů, nebo čárky a háčky, které jsou také znakovány pomocí pohybu ruky. Tyto pohyby ale nejsou detekovány použitou metodou rozpoznávání a také nebyla probrána možná řešení tohoto problému. Tímto problémem jsem se nezabýval, protože zobrazení háčku a čárek není složité a z mého pohledu by dané rozpoznávání neúměrně celý proces ztížilo.

Závěr

Práce se zabývá vývojem webové aplikace pro podporu výuky české jednoruční prstové abecedy. Aplikace integruje převod textu na 3D animaci a rozpoznávání znaků z videozáznamu, čímž vytváří komplexní prostředí pro efektivní učení a procvičování prstové abecedy.

Proces převodu textu na animaci vychází z předem vytvořeného 3D modelu postavy. Pro tento model byly vytvořeny animace české jednoruční i dvouruční prstové abecedy včetně reprezentace diakritiky a také animace několika základních slov českého znakového jazyka. Tento model je následně vykreslován ve webovém prohlížeči pomocí knihovny Three.js a nabízí uživateli možnosti měnit rychlost přehrávání animace či ovládat pohled kamery.

Rozpoznávání prstové abecedy ze záznamu webkamery bylo implementováno pro českou jednoruční prstovou abecedu. Aplikace je schopna spolehlivě rozpoznávat znaky zobrazované uživatelem. Tato funkce je v aplikaci využita dvěma hlavními způsoby: zaprvé slouží k ověření správnosti zobrazeného znaku, zadruhé umožňuje pokus o přepis souvislé sekvence znakování do souvislé textové podoby pomocí vyvinutého heuristického algoritmu.

Samotný proces rozpoznání gesta je dvoufázový. Nejprve jsou extrahovány souřadnice klíčových bodů ruky pomocí modelu Hand Landmark Detection z frameworku MediaPipe Solutions, který poskytuje robustní a přesnou detekci. Následně jsou tyto souřadnice transformovány a klasifikovány modelem typu Gradient Boosted Trees, natrénovaným s využitím knihovny TensorFlow. Trénovací dataset pro tento model obsahuje 7169 snímků od dvou osob a byl vytvořen v rámci této práce.

Spojením těchto dvou klíčových funkcionalit vznikla webová aplikace postavená na frameworku SvelteKit. Aplikace integruje zmíněné technologie a vytváří tak interaktivní prostředí pro výuku české prstové abecedy. Nabízí několik režimů použití: volnější režim Cvičiště pro experimentování, strukturovanější režim Učitel pro systematickou výuku a inovativní režim Konverzace, ve kterém může uživatel oboustranně komunikovat s velkým jazykovým modelem prostřednictvím znakování české jednoruční prstové abecedy.

I přes možná další rozšíření je výsledná aplikace plně funkční a představuje cennou pomůcku pro výuku. Zejména režim konverzace se ukázal jako zajímavý a zábavný způsob procvičování, který může významně přispět k osvojení prstové abecedy. Navíc implementace české dvouruční prstové abecedy a snadná rozšiřitelnost pro znaky znakového jazyka otevírají dveře k dalšímu rozvoji aplikace.

Conclusions

This thesis addresses the development of a web application to support the learning of the Czech one-handed finger alphabet. The application integrates text-to-3D animation conversion and finger alphabet recognition from video recordings, creating a comprehensive environment for effective learning and practicing the finger alphabet.

The text-to-animation conversion process is based on a pre-existing 3D character model. For this model, animations were created for the Czech one-handed and two-handed finger alphabets, including the representation of diacritics, as well as animations for a few basic words of Czech Sign Language. This model is subsequently rendered in a web browser using the Three.js library and offers the user the options to change the animation playback speed or control the camera view.

Finger alphabet recognition from webcam recordings was implemented for the Czech one-handed finger alphabet. The application is able to reliably recognize signs displayed by the user. This functionality is used in the application in two main ways: firstly, it serves to verify the correctness of the displayed sign and secondly, it allows attempting the transcription of a continuous sequence of signing into continuous text form using the developed heuristic algorithm.

The gesture recognition process itself is two-phase. First, the coordinates of key hand landmarks are extracted using the Hand Landmark Detection model from the MediaPipe Solutions framework, which provides robust and accurate detection. Then these coordinates are transformed and classified by a Gradient Boosted Trees model, trained using the TensorFlow library. The training dataset for this model, containing approximately 7,000 images from two individuals, was created as part of this work.

Combining these two key functionalities resulted in a web application built on the SvelteKit framework. The application integrates the mentioned technologies, thus creating an interactive environment for learning the Czech finger alphabet. It offers several modes of use: a freer Playground mode for experimentation, a more structured Teacher mode for systematic learning, and an innovative Conversation mode in which the user can interact with a large language model through signing the Czech one-handed finger alphabet.

Despite possible further expansions, the resulting application is fully functional and represents a valuable aid for learning. In particular, the conversation mode proved to be an interesting and entertaining method of practice, which can significantly contribute to the acquisition of the finger alphabet. Additionally, the implementation of the Czech two-handed finger alphabet and the easy extensibility for sign language signs open the door to further development of the application.

A Obsah přiloženého datového média

`text/`
: Text práce a veškeré soubory sloužící k jeho vytvoření.

`src/`
: Veškeré zdrojové kódy aplikace.

`dataset/`
: Vytvořený dataset české jednoruční prstové abecedy.

`README.txt`
: Návod pro zprovoznění aplikace.

Literatura

[1] *Aplikace Znakujte s Tamtamem.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://www.tamtam.cz/co-delame/mobilni-aplikace⟩.

[2] *Aplikace Hand Talk Translator.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://www.handtalk.me/en/app⟩.

[3] *Aplikace Sign Langauge ASL Pocket Sign.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://play.google.com/store/apps/details?id=com.mobireactor.signlanguage⟩.

[4] *Webová stránka spreadthesign.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://www.spreadthesign.com⟩.

[5] *Webová stránka ruce.cz.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨http://ruce.cz⟩.

[6] Šimčíková, Zuzana. *Prstová abeceda a její role ve vzdělávání žáků se sluchovým postižením na 1. stupni základní školy* [online]. 2012 [cit. 2025-3-23]. Dostupný z: ⟨https://library.upol.cz/arl-upol/cs/csg/?repo=upolrepo&key=44081307441⟩.

[7] *Blender.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://www.blender.org⟩.

[8] *Mixamo.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://www.mixamo.com⟩.

[9] *MediaPipe.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://ai.google.dev/mediapipe⟩.

[10] *TensorFlow.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://www.tensorflow.org⟩.

[11] *TensorFlow Decision Forests.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://www.tensorflow.org/decision_forests⟩.

[12] *Gradient Boosted Decision Trees.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://developers.google.com/machine-learning/decision-forests/intro-to-gbdt⟩.

[13] *Hand gesture recognition.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://github.com/kinivi/hand-gesture-recognition-mediapipe⟩.

[14] Shin, Jungpil; Matsuoka, Akitaka; Hasan, Md. Al Mehedi; Srizon, Azmain Yakin. American Sign Language Alphabet Recognition by Extracting Feature from Hand Pose Estimation. *Sensors* [online]. 2021, [cit. 2025-3-23]. Dostupný z: ⟨https://www.mdpi.com/1424-8220/21/17/5856⟩.

[15] *Svelte.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://svelte.dev⟩.

[16] *Flowbite svelte.* [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://flowbite-svelte.com⟩.

[17] *Tailwind css*. [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://tailwindcss.com⟩.

[18] *Three.js*. [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://threejs.org⟩.

[19] *Threlte*. [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://threlte.xyz⟩.

[20] *Seznam českých slov*. [online]. [cit. 2023-3-23]. Dostupný z: ⟨http://szj.cz/seznam-ceskych-podstatnych-jmen⟩.

[21] *Aplikace ollama*. [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://ollama.com⟩.

[22] *Platforma OpenAI*. [online]. [cit. 2025-3-23]. Dostupný z: ⟨https://platform.openai.com⟩.