VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

I-CT FRAMEWORK A APLIKACE PRO PŘEKLAD ZNAKOVÉHO JAZYKA

I-CT FRAMEWORK AND APPLICATION FOR SIGN LANGUAGE TRANSLATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. VOJTĚCH MECA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ FIALA
SUPERVISOR

BRNO 2017

Zadání diplomové práce/19426/2016/xmecav00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Meca Vojtěch, Bc.**
Obor: Inteligentní systémy
Téma: **i-CT Framework a aplikace pro překlad znakového jazyka**
i-CT Framework and Application for Sign Language Translation
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se s problematikou *speciálních vzdělávacích potřeb* (SVP) u osob s mentálním postižením.
2. Prostudujte stávající vhodné aplikace pro SVP zaměřené na překlad znakového jazyka na systémech iOS a Android. Analyzujte zejména jejich cenovou dostupnost, rozšiřitelnost a použitelnost.
3. Seznamte se s požadavky na návrh a vývoj aplikací z pohledu *Human Computer Interaction* (HCI) a požadavky návrhových principů *počítačové terapie* (i-CT) a rámce *i-CT Framework*.
4. Navrhněte a implemetujte rozšíření pro rámec *i-CT Framework* a aplikaci pro překlad znakového jazyka na tomto rámci. Aplikace bude funkční na systémech iOS a Android a dostupná jako open source.
5. Dle požadavků vedoucího testujte aplikaci v prostředí osob s mentálním hendikepem. Proveďte vyhodnocení s užitím metod HCI a diskutujte přínos praktického uplatnění a další rozvoj.

Literatura:

- Projekt I-SEN (otevřená komunita rodičů, pedagogů, terapeutů, IT odborníků), [online], [cit. 2016-09-10]. Dostupné na: http://www.i-sen.cz
- Jacobson Ivar a kol. *The Unified Software Development Process*, Addison Wesley, 2000.
- Fiala Jiří a Kočí Radek. *Počítačová terapie jako koncept nové formy terapie pro osoby s mentálním postižením: teorie i praxe.* Journal of Technology and Information Education. Olomouc: Universita Palackého, 2014, roč. 6, č. 1, s. 89-103.
- *i-CT Framework* (počítačová terapie) v2.0. [online], [cit. 2016-08-21]. Dostupné na: http://www.fit.vutbr.cz/~ifiala/prods.php?id=493&notitle=1

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Fiala Jiří, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem práce bylo vytvořit dvě aplikace pro využití v prostředí osob se specifickými vzdělávacími potřebami resp. osob s mentálním postižením. Jedna se o rámec i-CT Framework, který je zároveň spustitelnou aplikací pro výše uvedenou cílovou skupinu. Jako spustitelná aplikace rámec pracuje ve formě centrálního nástroje pro správu uživatelů, aplikací, oprávnění a bezpečnostních restrikcí. Dalším cílem práce pak bylo vytvoření nástroje Gesture Translator, díky kterému je možné překládat gesta znakového jazyka daného uživatele v uvedeném prostředí. Obě části práce fungují na operačních systémech Android a iOS.

Abstract

The aim of this thesis was to create two applications for people with learning difficulties. The first, i-CT Framework, is also an executable application to be used with the target group mentioned above. As an executable application, i-CT Framework operates as a central tool for managing users, applications, and user restrictions. Development of Gesture Translator application presented another aim of the thesis: this application can translate sign language gestures for people with learning difficulties. Both applications are functioning on Android as well as iOS operating systems.

Klíčová slova

Android, iOS, počítačová terapie, mentální postižení, znakový jazyk, OpenCV, HMM

Keywords

Android, iOS, Computer as Therapy, mental handicap, sign language, OpenCV, HMM

Citace

MECA, Vojtěch. *i-CT Framework a aplikace pro překlad znakového jazyka.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Fiala Jiří.

i-CT Framework a aplikace pro překlad znakového jazyka

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jiřího Fialy.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Vojtěch Meca
24. května 2017

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu diplomové práce Ing. Jiřímu Fialovi za věcné připomínky a cenné rady při psaní práce. Dále bych chtěl vyjádřit vděčnost za podporu ze strany firmy Red Hat, Inc. Jmenovitě děkuji také Ing. Kamilu Behúňovi z UPGM VUT FIT za cennou pomoc při řešení úloh spojených s rozpoznáním objektu a OpenCV. Na závěr bych chtěl poděkovat svoji ženě Marii, která mi byla oporou po dobu psaní této práce.

Obsah

Kapitola 1

Úvod

Chytré telefony nebo tablety se v současné době vyskytují všude ve světě. Člověk si už mnohdy ani nedovede svůj život bez těchto užitečných pomůcek přestavit. Je snadné získávat filmy, knihy, hudbu či nejrůznější aplikace pro tato zařízení různých operačních systémů z internetových obchodů. Tento trend v informačních technologiích (dále ICT) se velmi rychle rozvíjí, což je způsobeno především silným důrazem na vývoj mobilních zařízení a zároveň snahou snižovat pořizovací cenu těchto zařízení.

Přirozeně vyvstává otázka dostupnosti a využití mobilních zařízení: je možné využít tuto oblast ICT ke zlepšení či rozšíření běžné komunikace lidí s mentálním či jiným postižením? Odpověď lze nalézt například ve výstupech výzkumného projektu počítačové terapie, která např. definuje způsob, jakým lze SW pro tablety či chytré telefony použít ke vzdělání, osobnímu rozvoji a komunikaci lidí s mentálním postižením formou návrhových principů [19].

Cílem diplomové práce je vytvořit dvě volně dostupné aplikace pro dva nejběžnější operační systémy mobilních zařízení, Android a iOS. První aplikace umožní překládat gesta znakového jazyka v prostředí osob se speciálními vzdělávacími potřebami. Totiž uživatel bude pomocí kamery v zařízení zaznamenávat znakující osobu s mentálním handicapem, a aplikace bude následně překládat a na obrazovku vypisovat význam jednotlivých znaků. Druhá aplikace (tzn. nástroj frameworku) je pojítkem všech aplikací počítačové terapie v zařízení, tj. umožňuje spouštění jiných aplikací souvisejících s edukací či ulehčením komunikace lidí s mentálním postižením. Tato aplikace navíc uchovává i zdravotní, sociální a vzdělávací stavy konkrétních osob využívajících dané zařízení. Tato druhá aplikace je vyvíjena ve spolupráci se studentem Bc. Janem Kalinou. Aplikace také splňuje návrhové principy počítačové terapie (viz kapitola 2.2.6).

Druhá kapitola této práce vymezuje pojmy mentální retardace a její druhy. Kromě toho také předkládá komunikační prostředky, které byly v práci použity, a nachází se zde i konkrétnější popis počítačové terapie a jí definované návrhové principy. Kapitola dále vymezuje použité technologie v aplikacích a popisuje operační systémy Android a iOS. Anylýza a hodnocení existujících aplikací s podobným využitím překladu znakového jazyka je zdokumentovaná ve třetí kapitole. Ve čtvrté kapitole je pak předložen samotný návrh aplikací, a v páté je popsána jejich implementace. V šesté kapitole se pak nachází postup testování vyvinutých aplikací a jejich zhodnocení.

Kapitola 2

IT ve speciální pedagogice a znakovém jazyku

2.1 Základní pojmy

V této kapitole bude popsán souhrn definování důležitých pojmů souvisejících s touto diplomovou prací.

2.1.1 Osoby s mentálním postižením

Jedná se o jedince (děti, mládež i dospělí), u kterých dochází k zaostávání vývoje rozumových schopností, odlišného vývoje některých psychických vlastností a k poruchám v adaptačním chování. Hendikep u každého jedince je specifický a dá se říct, že jedinečný. Nicméně u většiny z nich se projevují (ve větší či menší míře) společné znaky, které vycházejí v závislosti na hloubce a rozsahu mentálního postižení, na rovnoměrnosti psychického vývoje jedince v rámci mentální retardace a na míře postižení jednotlivých psychických funkcí. [38]

2.1.2 Mentální postižení a jeho klasifikace

Mentální postižení (retardace) vzniká díky organickému poškození mozku, které dochází příčinou jeho abnormálního vývoje nebo strukturálního poškození mozkových buněk. Nejedná se o nemoc, ale jde o trvalý stav jedince. Klasifikaci mentální retardace se nachází v 10. revizi Mezinárodní klasifikace nemocí označená MKN-10 [35], zpracovaná Světovou zdravotnickou organizací v Ženevě. Mentálního postižení je rozděleno do těchto základních skupin:

Lehké mentální postižení (F70)

Rozsah IQ u lidí lehce mentálně retardovaných je 50 až 69. Chování u dospělých lidí odpovídá mentálnímu věku 9 až 12 let. Projevuje se problémy při výuce, nicméně mnoho z dospělých je schopno práce a často jsou schopni si udržet sociální vztahy se svým okolím. Patří sem lehká slabomyslnost, lehká mentální subnormalita a debilita [35].

Středně těžké mentální postižení (F71)

IQ se pohybuje přibližně mezi 35 až 50, což u dospělých odpovídá mentálnímu věku 6 až 9 let. Projevy jsou patrné ve vývojovém opoždění v dětství, nicméně mnozí z nich jsou schopni

částečné soběstačnosti, nezávislosti a dosahují přiměřené komunikace a školních dovedností. Dospělí potřebují různý stupeň podpory v jejich práci a v činnosti ve společnosti. Zde patří střední mentální subnotmalita, střední slabomyslnost, imbecilita [35].

Těžké mentální postižení (F72)

IQ dosahuje hodnot mezi 20 až 34. U dospělých osob to odpovídá věku 3 až 6 let. Lidé s touto formou postižení potřebují trvalou podporu. Zde se řadí těžká mentální subnormalita, těžká slabomyslnost, idioimbecilita [35].

Hluboké mentální postižení (F73)

IQ dosahuje nejvýše 20, což odpovídá u dospělých mentálnímu věku od narození do 3 let. Lidé s hlubokým mentálním postižením nejsou schopni samostatnosti a potřebují pomoc při pohybování, komunikaci a hygieně. Patří sem těžká mentální subnormalita, hluboká slabomislnost, idiocie [35].

2.1.3 Speciální vzdělávací potřeby

Ve většině případů je u osob s mentálním postižením nutné využít k jejich vzdělání speciálních vzdělávacích potřeb, angl. Special Education Needs (dále SEN). Tyto potřeby mají vést k jednoduššímu a pohodlnějšímu učení a rozvoji těchto osob. Důvod vzniku SEN je, že osobu s mentálním postižením nelze učit stejným způsobem jako osobu bez postižení. Konkrétně u lidí s mentálním postižením, které mají navíc poruchy komunikace, je podstatné dodržet tyto základní body:

1. **Zpomalit tempo řeči** - tedy osoba s mentálním postižením má dostatek času na zpracování sdělených informací.
2. **Udržovat zrakový kontakt** - tím se může navázat lepší kontakt s touto osobou a docílit efektivnější komunikace.
3. **Očekávat odpověď s trpělivostí** - osoba může při komunikaci využít i různých gest.
4. **Zprvu klást otázky s jednoznačnou odpovědí** - využívá se spíše u dětí, postupem času je možno klást otázky i složitější.

Důležité je uvědomit si skutečnost, že k osobám s mentálním postižením nelze přistupovat vždy stejně. Každá z nich se chová rozdílně, a proto je podstatné k těmto osobám přistupovat individuálním způsobem [33].

2.1.4 Alternativní a augmentativní komunikace

Alternativní a augmentativní komunikace (dále jen AAK) je zavedený způsob, jak lze kompenzovat (po určitou dobu či trvale) poruchy komunikace či samotného postižení u osob se závažnými expresivními komunikačními problémy (tj. se závažným postižením řeči, jazyka či psaní). AAK má za cíl umožnit lidem se závažnými poruchami komunikace účinně se dorozumívat a reagovat na podněty ve svém okolí, ideálně v takovém rozsahu, aby se stejně jako ostatní mohli aktivně účastnit života společnosti.

Augmentativní systémy podporují již existující, ale nedosahující úrovně samostatného dorozumívání, schopnost komunikovat.

Alternativní komunikační systémy se používají jako náhrada mluvené řeči. Tyto systémy mají snahu se zaměřit na maximální využití všech schopností daného uživatele. Dále způsob komunikace pomocí těchto systémů, by měl být co nejpřirozenější, jak je to jen v dané situaci možné. [37]

2.1.5 Hluchoněmota

Samotná hluchota je sluchové postižení vyskytující se u lidí, kteří mají nedostatečné schopnosti naslouchat všem zvukům ze svého okolí. To vede ke značnému omezení jejich komunikace s druhými lidmi. Hluchota může být vrozená (tj. existuje již při narození dítěte vlivem dědičnosti) či získaná v průběhu života (tj. poškození sluchového ústrojí po zánětech, infekčních chorobách, úrazech, apod.). Dále se hluchota dělí na úplnou a částečnou. Částečně hluchý (tj. nedoslýchavý) má různou schopnost vnímat okolí sluchem, tedy slyší:

- řeč, tj. dokáže částečně rozeznávat většinu hlásek
- hlas, tj. sluchem rozeznává jen samohlásky
- zvuky a tóny, tj. lidská řeč splývá v nejasnou melodií
- hluk, hřmot, tj. slouží k částečné orientaci venku

Přirozeným a nejtěžším následkem hluchoty je němota. Člověk s vadou sluchu nedokáže přesně napodobovat zvuky, kterými se dorozumívají lidé v jeho okolí. Tedy neví, jak použít řeč a mluvidla, i když tato ústrojí má plně funkční. Jedná se tedy o člověka hluchoněmého, který má kombinovanou vadu sluchu a řeči. Dále se tato vada může vyskytovat jako tzv. kombinovaná vada i u osob s mentálním postižením. Je tedy kladen důraz na využití vyvíjené aplikace pro tuto skupinu osob, pokud jejich intelekt dovoluje zvládnutí osvojení alespoň některých znaků znakového jazyka. [15]

2.1.6 Znakový jazyk

Jedná se o jazyk hluchých či hluchoněmých lidí. Tak jako u mluveného jazyka lze rozlišovat různé druhy znakových jazyků na úrovni států či dokonce dialektu konkrétního státu. Znakový jazyk se skládá z jednotlivých znaků, které se postupně znakují v daném pořadí tak, aby vyhovovali dané gramatice jazyka. V průběhu 60. až 70. let bylo stanoveno 6 základních komponent znaku:

1. Umístění znaku v prostoru
2. Tvar ruky
3. Pohyb ruky v prostoru
4. Orientace dlaně a prstů
5. Kontakt
6. Vzájemná poloha rukou

Tyto komponenty znaku nenesou význam samy o sobě, ale jsou schopny význam rozlišovat např. slova ŘEDITEL a SPORT se v českém znakovém jazyce liší pouze umístění stejného znaku v prostoru - ŘEDITEL se znakuje u pasu, ale SPORT v horní části těla.

Dále komponenty znaku existují simultánně. To znamená, že komponenty se vrší na sebe a tím lze vnímat celý znak v jednom okamžiku [36].

2.1.7 Český znakový jazyk

Českým znakovým jazykem (dále ČZJ) mluví převážně lidé s různou formou hluchoty v českém prostředí. Na základě výzkumu tohoto jazyka v Institutu pro neslyšící v Berouně roku 1993 byl vyvinut systém notace znaků ČZJ. ČZJ využívá pouze některé komponenty znakového jazyka:

1. Místo artikulace
2. Tvar ruky/rukou
3. Vztah ruky/rukou k tělu, tj. orientace dlaně a orientace prstů
4. Vztah ruky k ruce, tj. vzájemná poloha rukou u znaků artikulovaných dvěma rukama
5. Pohyb/y ruky/rukou

Složením těchto komponent můžeme vytvořit každý ze znaků ČZJ[1]. Způsob, jak manuální složku znaku zrekonstruovat, je zapsán v systému notace ČZJ, nicméně způsob zachycení nemanuální složky znaků ještě nebyl dodán [36].

2.2 Informatické prostředky a metody

Informatické technologie (ICT) dnešního moderního člověka lze rozdělit do dvou hlavních skupin:

- Zařízení, které nelze přenášet - zde patří desktopové počítače, servery, počítače různě zabudované ve spotřebičích, autech, apod.
- Mobilní zařízení - zde patří chytré telefony, tablety či notebooky.

Každé z těchto zařízení má své opodstatnění a využití, nicméně běžný uživatel se dostává nejčastěji k mobilním zařízením. V dnešní době je snaha tyto zařízení vylepšovat, jak po stránce hardwaru (zlepšování výkonu zařízení), tak i po stránce softwaru (zpřístupnění dostatečného množství nových aplikací).

Protože je těžiště této práce zaměřeno na mobilní dotyková zařízení, budou v této kapitole uvedeny operační systémy Android a iOS, dále pak současné prostředky pro vývoj multiplatformních aplikací na mobilní zařízení s těmito operačními systémy. Poté se představí projekt „Počítačové terapie“ a nakonec budou předvedeny současné přístupy v ICT pro překlad znakového jazyka.

[1] Slovník nejen českého znakového jazyka lze nalézt na portálu ruce.cz. Dostupné z: http://ruce.cz/slovnik.

2.2.1 Operační systémy mobilních zařízení

Aplikace pro překlad znakového jazyka a i-CT Framework byly pro potřeby této práce vytvářeny pro dva nejčastější operační systémy současné doby, a to operační systém iOS od společnosti *Apple Inc.* a operační systém Android od společnosti *Google Inc.* Bylo tak učiněno z důvodu přirozené potřeby překonávat platformní odlišnosti plynoucí z uvedených návrhových principů (kapitola 2.2.6). Díky tomu mohou být tyto aplikace dostupné pro početnější cílovou skupinu, což je jeden z požadavků těchto aplikací (kapitola 3.2).

V této části práce budou popsány základní informace o operačních systémech (dále OS) Android a iOS se zaměřením na funkce a vlastnosti, které byly využity při vývoji obou aplikací.

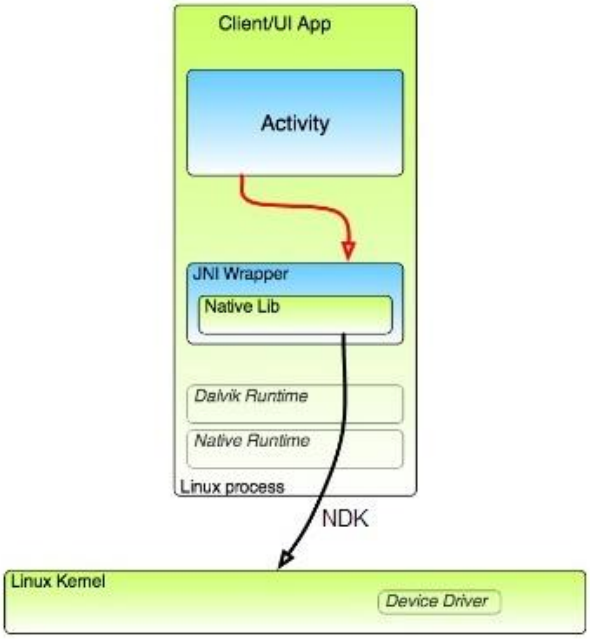
2.2.2 Operační systém Android

OS Android je *open source* platforma určená pro mobilní zařízení (tj. tablety a tzv. smartphone). Základem tohoto OS je nejčastěji linuxové jádro 2.6 různých verzí, které zabezpečuje systém jako celek, správu paměti, správu procesů apod. OS Android je od roku 2007 vyvíjen konsorciem *Open Handset Alliance.* Od té doby bylo v OS Android provedeno mnoho změn. Výhodou tohoto OS je jádro Androidu, které je tvořeno pro více druhů hardwaru, to znamená, že nezáleží na konkrétním chipsetu, velikosti nebo rozlišení obrazovky daného zařízení [31]. Další výhodou je ta skutečnost, že v Androidu můžeme využít Apache Cordova Framework 3.6.2 a také nativní knihovny OpenCV, vhodné pro rozpoznávání obrazu (kapitola 2.2.7 sekce *Metody pro rozpoznávací mechanismus s OpenCV*).

OpenCV v Androidu

OpenCV je nativní knihovna v Androidu. Pro spuštění nativních externích knihoven v Adnroidu slouží *The Native Development Kit* (dále jen NDK). NDK se zasloužilo k tomu, že nativní knihovny pracují přímo s vrstvou *Linux Kernel.* Jádro Androidu tedy s nativními knihovnami pracuje rychle a efektivně. Obecně ale platí, že nativní knihovna je napsaná v programovacím jazyce C či C++, kdežto Android je založen na programovacím jazyku Java. Proto vzniklo speciální spojení mezi Android aplikaci a nativními externími knihovnami, které se nazývá *Java Native Interface* (dále JNI). JNI Wrapper (obálka) zabaluje volání C a C++ metod. Na obrázku 2.1 je zjednodušeně znázorněno volání nativní knihovny v androidní aplikace a práce NDK [6].

OpenCV v Androidu je zrealizováno způsobem, kde všechny nativní metody v jazyce C a C++ a jejich binární soubory jsou uloženy ve zvlášť speciální aplikaci s názvem „OpenCV Manager“. Dále pak aplikace využívající OpenCV knihovnu mají ve speciálních třídách zrealizovaný JNI Wrapper, který pak „komunikuje“ s aplikací „OpenCV Manager“. To má za výhodu, že pro vývojáře je vše nachystané a je velmi snadné s knihovnou OpenCV pracovat. Další výhoda z tohoto přístupu je ta, že každá koncová aplikace s OpenCV nemusí zvlášť obsahovat binární soubory a tedy se šetří fyzickou pamětí mobilního zařízení. Také všechny metody v „OpenCV Manageru“ jsou optimalizované a i v tomto centrálním systému se jednoduše provádí update verzí OpenCV knihovny [10].



Obrázek 2.1: Volání nativní knihovny a znázornění práce NDK.

2.2.3 Operační systém iOS

Operační systém iOS je vymezen pro mobilní zařízení společnosti *Apple Inc.*, která tento OS vyvíjí. Ačkoliv v dnešní době je iOS využíván v zařízeních jako iPhone, iPad nebo Apple TV druhé generace, jeho původní účel byl určen pouze pro iPhone. Výhoda tohoto OS je, že firma *Apple Inc.* vyvíjí současně jak hardware, tak i software. Z toho plyne, že iOS je při používání koncovými uživateli svižný a plynulý. Jedna z dalších výhod je, že plně využívá Apache Cordova Framework 3.7.0. Nevýhodou tohoto OS je skutečnost, že iOS není možno použít na žádné jiné zařízení než od výrobce *Apple Inc.* Nutno podotknout, že zařízení od společnosti *Apple Inc.* jsou v současné době v České republice ve vyšší cenové relaci. Na druhou stranu patří v naší zemi mezi nejpoužívanější mobilní dotyková zařízeni ve speciální pedagogice a to i s ohledem na připravenost prostředí tohoto OS pro některé speciální vzdělávací potřeby.

OpenCV v iOS

V operačním systému iOS je také možné použít knihovnu Open CV. Bohužel nevýhoda je, že knihovna OpenCV zde nevyužívá GPU, ale pouze CPU. Tím není zcela využit maximální výkon mobilního zařízení. Nicméně zatím není známa jiná knihovna, která by mohla být náhradou čisté knihovny OpenCV. Ovšem iOS má možnost využít OpenCL knihovnu, která již využívá GPU. Pokud by tedy vznikl port mezi OpenCV a OpenCL (zatím existuje OpenCV OML, což je zavedený modul, který simuluje port mezi převodem vlastnosti OpenCV na OpenCL, pouze jen pro desktop PC), bude možnost maximálně využít výkon na zařízení iOS s použitím OpenCV. Tento port nebyl k dispozici na iOS zařízení v době implementace této diplomové práce [24].

Vrstva Cocoa Touch (jedna ze 4 vrstev, na kterých je postaven iOS) obsahuje frameworky, které se vyvolávají při sestavování aplikace. Tyto frameworky definují vzhled

aplikaci. Dále podporují základní infrastrukturu aplikace, jako je multitasking, zpracování vstupu od uživatele („pokliknutí“na tlačítko), tzv. push notifikace a mnoho vysokoúrovňových služeb v iOS.

Open CV je umístěná ve vrstvě Cocoa Touch. Tedy OpenCV chová jako další „framework“, který se volá z této vrstvy.

2.2.4 Současné prostředky pro multiplatformní vývoj

V této kapitole je představen stručná charakteristika multiplatformních nástrojů, které lze v současné době využít.

Apache Cordova Framework

Apache Cordova Framework je volně dostupný framework určený pro vývoj aplikací na mobilních zařízeních. Framework zpřístupňuje použití standardních webových technologií, jako HTML5, CSS3, JavaScript, atd., pro multiplatformní vývoj (iOS, Android, Windows Phone 7/8/10, Blackbarry, atd.). Samotná vizualizace cordova aplikace (modul View popsaný v kapitole 4.1) je tvořena obecným kódem webových stránek s použitím výše popsaných webových technologií pro všechny platformy.

Pro přístup ke standardním API aplikace, jako například přístup k datům, ke kameře, push notifikace, atd., se využívá tzv. Cordova pluginů. Každý z těchto pluginů je složen z několika modulů; jeden modul pro jednu konkrétní platformu s konkrétní implementaci přístupu k danému API. Jednotný přístup ke všem modulům v pluginu je zajištěn pomocí JavaScriptu.

Cordova pluginů se využívá i v dalších multiplatformních vývojových prostředích jako Oracle MAF popsaný níže v kapitole 2.2.4.

Xamarin

Xamarin je multiplatformní vývojové prostředí založené na *.NET* frameworku. Je tedy možno generovat z jednoho jediného zdrojového kódu spustitelné balíčky různých operačních systémů Androidu a iOS, ale i Windows Phone 10. Programovací jazyk, který využívá Xamarin, je C#. Xamarin SDK je od 31. Března 2016 zveřejněno pod *open source* licencí [29].

Pro jednotný vývoj se v Xamarinu využívá *Xamarin.Forms API*. Toto API je vhodné pro aplikace, kde se nejedná o přílišně velké rozdílnosti pro specifika jednotlivých platforem, a dále kde sdílení kódu je více důležitější než samotné GUI aplikací. GUI v *Xamarin.Forms API* je zajištěno pomoci XAML souborů, kde jsou Xamarinem nadefinované základní komponenty [12].

V Xamarinu je možno vyvíjet i aplikace určené přímo pro jednu platformu. K tomu se využívá *Xamarin.Android API* určený pro Android a *Xamarin.iOS API* určený pro iOS. Toto řešení se využívá hlavně v případech, když aplikace vyžaduje nativního chování (externí nativní knihovny, GPS, přehrávače videí, audia, apod.), dále když je kladen větší důraz na GUI než na sdílení kódu a také když GUI se pro jednotlivé platformy výrazně liší.

Pro Xamarin existují dvě vývojové prostředí a to Xamarin Studio a pak Visual Studio od firmy *Microsoft Corporation*. Xamarin Studio je určeno primárně pro uživatele používajíc Mac OS na tvorbu Android a iOS aplikací. Visual Studio je v nabídce jako nástroj pro tvorbu Android, iOS a Windows Phone 10 aplikací s těmito licenčními třídami [5] [13]:

- Visual Studio Community - Tato verze je zdarma. Je primárně určena pro studenty, *open source* projekty a pro malé týmy, kde počet členů musí být menší než 6 lidí. Není žádný limit na velikost výsledného balíčku a můžou se zde vyvíjet aplikace s výše popsanými API bez omezení.

- Visual Studio Professional - Tato verze je již placená. Oproti verzi zdarma se jedná o verzi, kterou používají již profesionální vývojáři menších firem. Je zde plnohodnotná podpora ze strany Xamarinu.

- Visual Studio Enterprise - Tato verze je taky placená. V případě koupi této verze se již nejedná o žádné omezení, snad jen při použití Xamarin testovacího Cloudu, kde je sleva 25% procent na tuto službu.

Bohužel vývoj aplikací začal dříve, než byly podmínky u Xamarinu takto uvolněny, proto byl vybrán nástroj Mobile Application Framework od firmy Oracle popsaný níže v kapitole 2.2.4.

Oracle Mobile Applicatoin Framework

Oracle Mobile Application Framework (MAF) je hybridní framework[2] založen na programovacím jazyku Java, HTLM5, CSS, Javascriptu a Apache Cordova Pluginů. Díky těmto technologiím se z MAF stává multiplatformní nástroj, kdy z jednoho zdrojového kódu je možno generovat spustitelné balíčky jak pro iOS, Microsoft Windows 10 platformy, tak i pro Android [7]. Architektura MAF vychází z architektonického vzoru MVC popsaného v kapitole 4.1. Dále je MAF volně dostupný doplněk vývojového prostředí Oracle JDeveloper. Existuje také upravený MAF v podobě frameworku do vývojového prostředí IDE Eclipse pod projektem *Oracle Enterprise Pack for Eclipse* [7]. V této práci bylo použito pro vývoj obou aplikací vývojové prostředí jDeveloper.

Vrstva View a z části i vrstva Controller z architektonického vzoru MVC jsou v MAF zastoupeny komponentou **Web View**. **Web view** má na starost zobrazování a zpracovávání obsahu webových stránek, z kterých je GUI aplikace tvořeno. Tato komponenta se skládá z těchto částí:

- Application Mobile XML (dále AMX) - jedná se speciální soubory určené pro zobrazení vzhledu jednotlivých stránek aplikace. AMX předkládá vývojáři předem nadefinovaných zhruba 50 komponent, mezi které patří tlačítka, tabulky, seznamy, obrázky a jiné.

- Local HTML - Vývojář může využít svých znalostí a nadefinovat vzhled jednotlivých stránek aplikace podle standardů HTML5 a CSS bez předkládaných AMX komponent to vše v prostředí nabízeném JDeveloperu.

- Server HTML - obsah výše popsaných částí **Web view** je generováno serverem a je uživateli předkládáno jako GUI aplikace formou webových stránek. HTML kód, řídící logika a logika interakce mezi jednotlivými stránkami je generovaná na vzdáleném serveru. Dále pak server HTML může přistoupit pomocí Javascriptu k Apache Cordova pluginům a tak zpřístupnit aplikaci nativní funkce mobilního zařízení.

[2] Kompromis mezi nativními a webovými aplikacemi. GUI aplikací je tvořeno webovými technologiemi a přístup k nativním funkcím mobilního zařízení se provádí pomocí Apache Cordova Javascript API, jako fotoaparát, SMS, Push Notifikace, atd.

Vrstva Model a také vrstva Controller ze vzoru MVC jsou reprezentovány komponentou **Java Virtual Machine** (dále JVM) v MAF aplikacích. Jedná se o běhové (ang. runtime) Java prostředí pro MAF aplikace. JVM je implementováno nativním kódem zařízení a do každé MAF aplikace je vloženo zvlášť. Je to z toho důvodu, že systém iOS nemá vlastní Java runtime prostředí. Tvůrci MAFu se bohužel rozhodli, že i pro aplikaci v Androidu JVM zahrnou a nevyužijí tak vestavěného JVM Dalvik. Proto u Androidních aplikací jsou dva duplicitní Java runtime prostředí a plýtvá se tak fyzické paměti zařízení (JVM MAFu zabírá přibližně 30 MB). Dále je JVC založeno na specifikaci JavaME Connected Device (CDC) a využívá navíc těchto částí:

- Business Logic - jedná se o logiku MAF aplikací. Tato logika je zajišťována formou *Managed Beans*. *Managed Beans* jsou Java třídy, díky kterým je možno rozšířit funkcionalitu MAF aplikací. Dále lze pomocí *Managed Beans* řídit a zpracovávat data vrácené ze serveru. Podobně jako JVC tak i *Managed Beans* spadají pod JavaME CDC specifikaci.
- Model - spojuje *Managed Beans* a webové rozhraní aplikace (AMX či Local HTML). Tato část také poskytuje logiku pro vyvolání REST a SOAP webových služeb.
- JDBC - jedná se o API, díky kterému je možné přistoupit k šifrované SQLite databáze.

2.2.5 Výzkumný projekt Počítačová terapie

Počátkem roku 2013 byl v prostředí osob s mentálním postižením zformován projekt tzv. počítačové terapie, jehož autorem je Ing. Jiří Fiala, vedoucí této práce. Tento projekt nadále úspěšně pokračuje ve spolupráci s dalšími subjekty a to vedle zázemí Fakulty Informačních Technologií VUT v Brně také díky společnosti *Red Hat, Inc.* [11], odborné komunitě iSEN [8] a podpoře neziskového sektoru zaměřeného na edukaci a terapie osob s mentálním postižením skrze ICT prostředky. Přestože má tento projekt speciálně pedagogická a sociálně pracovní východiska, z informatického pohledu se jedná o aplikovaný výzkum v oblasti softwarového inženýrství s využitím dalších přidružených informatických metod, jako jsou např. metody z oblasti interakce člověka s počítačem (HCI – „*human computer interaction*“), viz kapitola 3.1. Tato práce tak představuje dosud nepokrytou oblast ICT výzkumu pro osoby s mentálním postižením, která je dlouhodobě v neuspokojivém stavu.

Již na počátku v roce 2013 byly identifikovány první nedostatky ve vývojovém procesu u aplikací pro osoby s mentálním postižením. Vychází se zde z přímých zkušeností speciálních pedagogů zejména z odborné komunity iSEN [8], ze zkušeností z obecné práce v sociálních službách, řady odborných studií na toto téma, kritického zhodnocení současných aplikací, ale také přímo z vlastní zkušenosti vedoucího této práce – 4 roky technické vedení osob s mentálním postižením v programu na jejich podporu skrze ICT prostředky. A právě za účelem identifikace vývojových nedostatků a společných aplikačních požadavků pro tuto oblast, byla vedoucím práce vytvořena a vedena stejnojmenná terapie v prostředí osob s mentálním postižením, která sloužila také ke zpětnému testování a verifikaci navrhovaných řešení [19].

2.2.6 Návrhové principy počítačové terapie

Byly tedy definovány první návrhové principy s cílem odstranění identifikovaných nedostatků, tedy s cílem zajištění efektivního vývoje koncových aplikací pro tuto cílovou skupinu a také pro zaručení vyšší přístupnosti a použitelnosti těchto aplikací než je tomu u současně

dostupných řešení. Protože se ve svém důsledku předložené návrhové principy přímo promítají do celého vývojového procesu a ovlivňují i architekturu návrhu můžeme některé z těchto principů vyjádřit také jako návrhové nebo architektonické vzory.

Návrhové principy počítačové terapie jsou v rámci výzkumu dále vyvíjeny a ověřovány. První příspěvek o těchto principech byl již popsaný v časopise *Journal of Technology and Information Education* [19]. Aktuální verze návrhových principů počítačové terapie byla naposledy rozšířena na 14 návrhových principů viz [20]. Tato práce, resp. její návrhové principy, vychází z této poslední verze.

Základní významy těchto návrhových principů jsou stručně uvedeny pro potřeby této práce v bodech:

- Odstínění vývoje pro koncová dotyková zařízení s důrazem na multiplatformní vývoj.
- Využití otevřeného vývojového prostředí, které je dostupné zdarma.
- Vlastní implementace ve vyšším programovacím objektově orientovaném jazyce.
- Vývoj realizován podle současných standardů.
- Vyvinutá aplikace je snadno dostupná a je zdarma, publikována jako *open-source* pod vhodnou licencí.
- Aplikace musí být vyvíjena cíleně (např. cílem je kompenzovat nedostatečnost v řeči).
- Aplikace musí být snadno použitelná i pro běžné prostředí osob s mentálním postižením (sociální prostředí), tedy i mimo školní prostředí.
- Aplikace musí být použitelná i v režimu *offline*.
- Aplikace musí být vyvíjena s možností konfigurace pro individuální potřeby osob s mentálním postižením.
- Obecná použitelnost aplikace pro široké využití osob s mentálním postižením.
- Oddělení režimu běžného uživatele, pedagoga či asistenta a správce aplikace.
- Aplikování bezpečnostních zásad a zásad pro ochranu osob s mentálním postižením.
- Aplikace by měla mít lokalizaci na více než Češtinu.

Návrhové principy byly následně implementovány do první verze společného frameworku (tzv. *framework počítačové terapie*[3] *nebo zkráceně též i-CT Framework*), který obsahuje základní metody a atributy společné napříč různými oblastmi práce (vzdělávání, terapie, soc. péče) u osob s mentálním postižením. Framework je dále doplněn o praktický nástroj, který může poskytnout svou funkcionalitu i ve formě spustitelné aplikace. Přínos tohoto přístupu, prvních implantací frameworku i aplikací a výsledky získané z již uskutečněného ověřování v praktickém prostředí osob s mentálním postižením již byly diskutovány a publikovány např. na *ICACET* konferenci v Singapuru, kde vystoupili zakladatelé této terapie [18], v bakalářské práci pana Tůmy [30], v bakalářské práci autora této diplomové práce pana Mecy [23], dále v diplomové práci pana Vejtasy [32] a v diplomové práci pana Kaliny [22].

[3] Soubor programů (nejčastěji aplikací na mobilní zařízení), které jsou vyvíjeny v rámci projektu Počítačové terapie. Dostupné z: http://www.fit.vutbr.cz/~ikrajice/prods.php?id=405&notitle=1

Tato práce se s odkazem na rozšíření původně identifikovaných vývojových nedostatků a z nich vycházejících návrhových principů zaměřuje na novou implementaci frameworku, která je současným stavem výzkumu vyžadována, a dále pak na vývoj nadstavbové praktické koncové aplikace Gesture Translator, která je v případě často vyskytované kombinované (sluchové či řečové) vady u osob s mentálním postižením nezbytnou součástí pro zvládání jejich základních životních potřeb. Tato nadstavbová aplikace, spuštěna nad vytvářeným frameworkem, ve svém důsledku vytváří s tímto frameworkem jeden celek, který poskytuje koncovému uživateli tíženou kompenzace či vzdělání ve smyslu jeho postižením.

Vývoj frameworku je výsledkem kooperace dvou studentů, Jana Kaliny (zdokumentovaná v jeho diplomové práci [22]) a autora této diplomové práce Vojtěcha Meci (dále popsána v kapitole 4.2) vždy pod společným vedením vedoucím této práce.

2.2.7 Současné přístupy pro překlad znak. jazyka

V dnešní době se stále více výzkumné týmy zaměřují na vývoj překladače znakové řeči v reálném čase. Vznikají různé výzkumné skupiny nejen na universitách po celém světě, kde se snaží tuto výzvu zdolat a tak zpřístupnit komunikaci hluchoněmým lidem s lidmi slyšícími. V této práci je zanalyzovaná činnost alespoň jedné z těchto výzkumných skupin (viz kapitola 3.4.2).

Výzva překladu znakové řeči se dá rozdělit do dvou menších částí:

- **rozpoznávací mechanismus** - získávání pozic rukou a hlavy
- **rozpoznávací logika** - logika, která dokáže díky dané sekvenci pozic rukou a hlavy vyhodnotit, které gesto sekvence reprezentuje

Každou z těchto částí je třeba řešit jinou technikou či kombinace technik. Tyto techniky jsou rozděleny do několika základních skupin [27]:

- techniky detekce - vhodné pro **rozpoznávací mechanismus**. Zde lze zařadit detekce barvy kůže rukou, 3D model snímané osoby, detekce pohybu rukou, apod.
- sledovací (angl. tracking) techniky - také vhodné pro **rozpoznávací mechanismus**. Zde patří například zjišťování pozic rukou pomocí počítání optického toku (angl. optical flow), atd.
- rozpoznávací techniky - vhodné pro **rozpoznávací logiku** - zde můžeme zahrnout například Skryté Markovovy Modely (angl. Hidden Markov Models, dále jen HMMs)

Metody pro rozpoznávací mechanismus s OpenCV

OpenCV je *open source* knihovna a využívá se pro manipulaci s obrazem, především pro počítačové vidění a zpracování obrazu v reálném čase. Výhodou této knihovny je multiplatformnost a to jak na desktopové aplikace, tak na mobilní aplikace pro Android OS a iOS. Nevýhodou je, že v Apache Cordova Frameworku není plugin, který by implementoval zároveň tuto knihovnu pro iOS tak i pro Android OS. Tedy ve vývoji pro samotnou aplikaci využijeme paralelního programování mezi jednotlivými OS.

Avšak samotná knihovna OpenCV nepředkládá systém, který by rozpoznával ruce či hlavu a získával tak všechny potřebné informace o provádění gesta. OpenCV ale předkládá celou jednu větev vývoje určenou pro analýzu pohybu objektů popřípadě sledování objektů ve videu. V této sekci vývoje OpenCV lze nalést třídy a funkce řešící např. Kalmanův

filtr [25], algoritmus Mean Shift [17], algoritmus „Gaussian mixture model background subtraction“ [34] či algoritmus získání optického toku založený na Lucas-Kanade metodě s pyramidami [16].

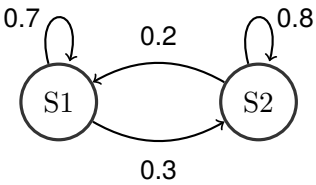
Vhodnou kombinací funkcí či tříd je možné dosáhnout vytvoření detektoru rukou a hlavy z videa pomocí knihovny OpenCV. Postup a výběr konkrétních funkcí či tříd použitých v této práci je popsán v kapitole 5.2.3.

Technika řešení rozpoznávací logiky

V současné době se v inteligentních systémech pro potřebu řešení různých překladačů (ať už překladač znakového jazyka či mluvené řeči) v reálném čase využívá několik modelů. Mezi tyto modely dominuje použití Skrytého Markovského Modelu (angl. Hidden Markov Model, dále jen HMM).

Úvod do Skrytých Markovských Modelů

Předpoklad pro pochopení HMM je Markovův model, ze kterého HMM vychází. Markovův model se skládá z množiny stavů, dále množiny symbolů, které model může generovat, a množiny přechodů mezi těmito stavy. Na obrázku 2.2 lze vidět Markovský model, který generuje řetězce o symbolech A1 a B1, se stavy S1 (generuje symbol A1) a S2 (generuje symbol B1) a s přechody, které ukazují, jaká je pravděpodobnost přechodu z jednoho stavu do druhého (shrnuto v matici vpravo). Hodnoty pravděpodobnosti jsou normalizovány v intervalu 0 až 1. Model se v jednom okamžiku může nacházet pouze v jediném stavu a může generovat pouze jeden symbol řetězce. Tento okamžik bude označen jako q_t.



$$\mathrm{A} = \begin{array}{c} \\ S1 \\ S2 \end{array} \begin{array}{c} \begin{array}{cc} S1 & S2 \end{array} \\ \begin{pmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{pmatrix} \end{array}$$

Obrázek 2.2: Příklad Markovského modelu

Navíc je definováno, že součet všech přechodů vycházejícího u každého stavu je přesně 1, což si lze z obrázku modelu povšimnout. Dále pak matice A je složená z prvku a_{ij} [28], kde

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i) \tag{2.1}$$

Pomocí tohoto modelu lze tedy určit pravděpodobnost po sobě jdoucích stavů modelu - tedy když se předloží modelu řetězec složený ze symbolů těchto stavů je možné vypočítat celkovou pravděpodobnost generování předloženého řetězce vůči modelu. Pro názorný příklad se zvolil řetězec O složených z těchto symbolů B1,A1,B1,A1,A1. Zavádí se předpoklad, kde pravděpodobnost inicializace modelu pro stav A1 je roven 0 a pro stav B1 je roven 1 (obecně je pak inicializační vektor definován rovnici 2.3). Vyjádřeno rovnici:

$$\begin{aligned} P(O|Model) &= P(B1, A1, B1, A1, A1|Model) = \\ &= P(B1) * P(A1|B1) * P(B1|A1) * P(A1|B1) * P(A1|A1) = \\ &= 1 * 0,2 * 0,3 * 0,2 * 0,7 = 0,0084 \end{aligned} \tag{2.2}$$

Výsledná pravděpodobnost, že model vygeneruje řetězec O je tedy 0,0084.

Inicializace modelu se nastavuje pomocí inicializačního vektoru π, pro který platí [28]

$$\pi = \{\pi_i\} \tag{2.3}$$

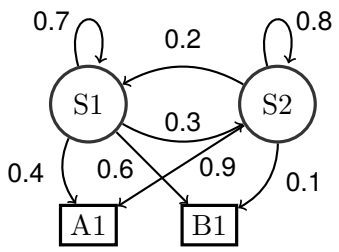
kde

$$\pi_i = P(q_1 = S_i) \tag{2.4}$$

Hodnoty řetězce daného Markovského modelu budou v textu dále označovány:

$$V = v_1, v_2, ...$$

Markovský model má omezení, že v daném stavu jednoho okamžiku je možno generovat pouze jeden předem definovaný symbol řetězce. A zde přichází rozšíření modelu v podobě HMM, kde je možné z jednoho stavu generovat více symbolů řetězce. Stále platí, že v jednom okamžiku je možné generovat pomocí HMM pouze jeden symbol řetězce, ale z jednoho stavu je možné generovat více symbolů v průběhu času výpočtu. Názorná ukázka HMM je na obrázku 2.3.



Obrázek 2.3: Příklad HMM

HMM je mimo matici pravděpodobností přechodů mezi jednotlivými stavy (matice A, obrázek 2.2 napravo) a vektorem pravděpodobností počátečního stavu modelu (vektor π 2.3) rozšířena ještě o jednu matici, která popisuje s jakou pravděpodobností daný stav vygeneruje konkrétní symbol 2.5. I zde je stanoveno pravidlo, že součet pravděpodobností generování všech symbolů řetězce u jednoho stavu je roven 1.

$$B = \{b_j(k)\} \tag{2.5}$$

kde

$$b_j(k) = P(v_k\, v\, t | q_t = S_j), 1 \leq j \leq N, 1 \leq k \leq M \tag{2.6}$$

kde N je počet stavů HMM a M číslo symbolů, které je možno generovat z daného stavu [28]. Konkrétně pro případ na obrázku 2.3 matice B nabývá tvaru:

$$\mathrm{B} = \begin{array}{c} \\ S1 \\ S2 \end{array} \begin{array}{c} \begin{array}{cc} A1 & B1 \end{array} \\ \begin{pmatrix} 0.4 & 0.6 \\ 0.9 & 0.1 \end{pmatrix} \end{array}$$

Definice celé HMM je pak:

$$\lambda = (A, B, \pi) \tag{2.7}$$

Existují tři hlavní otázky při použití HMM:

1. **Evaluace** (Evaluation) - Je dán řetězec O a model $\lambda = (A, B, \pi)$. Jaká je pravděpodobnost, že model λ vygeneruje daný řetězec O - tedy pravděpodobnost $P(O|\lambda)$?

2. **Dekódování sekvence stavů** (Estimation) - Jaká je nejpravděpodobnější sekvence skrytých stavů při generování daného řetězce O?

3. **Učení HMM** (Optimisation) - Jak změnit parametry modelu $\lambda = (A, B, \pi)$, tak aby generoval daný řetězec O s větší pravděpodobností?

Pro vyřešení jednotlivých zmíněných otázek zpravidla existují algoritmy, které dané problém řeší:

- **Evaluace** - Forward-Backward algoritmus
- **Dekódování sekvence stavů** - zde se využívá Viterbiho algoritmus
- **Učení HMM** - Baum-Welsch algoritmus či jeho optimalizace pomocí scaling

Každý z těchto algoritmů je detailně popsán v práci pana Rabinera s názvem *A tutorial on hidden Markov models and selected applications in speech recognition* [28]. V kapitole 4.3.1 bude stručně uveden návrh použití těchto algoritmů v aplikaci Gesture Translator.

Kapitola 3

Specifikace a analýza požadavků

V této kapitole bude popsána analýza požadavků na aplikaci i-CT Frameworku a pak na systém aplikací překladu znakové řeči.

Aplikace vyvíjené v rámci této práce budou vycházet z návrhových principů počítačové terapie, které se ukazují jako vhodné pro zkvalitnění vývojového procesu aplikací speciální pedagogiky, naplnění jejich potřeb a nakonec i pro zvýšení vlastní koncové použitelnosti v praxi 2.2.6. Tento přístup má značné výhody popsané v dokumentu *Mentally challenged as design principles and models for their applications* [20] a proto se využije v této práci, tj. návrhové principy počítačové terapie se budou považovat jako zdrojová doména pro požadavky kladené na vývoj aplikací této práce. Toto bude následně promítnuto zejména v následujících podkapitolách této kapitoly:

- 3.2 Obecné společné požadavky na vývoj
- 3.3 Specifikace požadavků na i-CT Framework
- 3.5 Specifikace požadavků na aplikaci GestureTranslator

3.1 Interakce člověk-počítač

V angl. *human-computer interaction* (dále HCI). Jedná se o průnikový obor (tj. mezioborová disciplína), který se zabývá výzkumy v oblastech vztahu člověk-počítač, mentálních modelů, informačního chování, tyto oblasti jsou výstupem HCI. HCI výrazně přispívá ke zdárné implementaci informačních technologií a to i v případě návrhových principů počítačové terapie.

Jak již bylo zmíněno, tak HCI je průnikový obor, který je ovlivňován a naplňován zejména těmito disciplínami:

- Počítačová věda
- Design
- Psychologie
- Umělá inteligence
- Inženýrské obory
- a jiné.

Podobně jako informační věda, tak i HCI detailně zkoumá koncového uživatele, jeho okolí, chování a informační potřeby. Tedy HCI lze chápat jako obor s úzkými vazbami a rozhraním k informační vědě. Tato práce pro své potřeby bude vycházet z přístupu HCI a to zejména s odkazem na design a počítačové vědy [26].

3.2 Obecné společné požadavky na vývoj

Protože se diplomová práce zaměřuje na vývoj dvou aplikací nikoliv pouze jedné, nastala potřeba stanovit společné požadavky pro vývoj těchto aplikací v souladu s návrhovými principy počítačové terapie (kapitola 2.2.6). Důležité je uvědomit si, že i-CT Framework a jeho spustitelný nástroj je určený pro širokou škálu lidí s mentálním postižením. Kdežto aplikace pro překlad znakového jazyka v reálném čase je určena pouze pro konkrétní skupinu lidí se speciálními vzdělávacími potřebami v kombinaci s postižením hluchoněmosti. Tedy při definování společných obecných požadavků na vývoj těchto aplikací bude brán v úvahu i tento faktor. Na základě zmíněných pohledů na obecné požadavky byly definovány tyto body:

1. Vývoj aplikací musí být multiplatformní. To z důvodu, aby nejen byly splněny požadavky vyplývající z návrhových principů počítačové terapie, ale hlavně, aby aplikace byly dostupné pro co nejvíce lidí s mentálním postižením.

2. Bude se jednat o aplikace na mobilní dotyková zařízení nikoliv aplikace pro desktop PC. A to z praktického pohledu, aby lidé, co budou používat tyto aplikace, byli nezávislí na konkrétním místě např. jeden z cílů aplikace s překladačem znakového jazyka je začlenění uživatele do všedního života při nákupech, setkáních apod.

3. Pro obě aplikace bude lokalizace nastavena na český jazyk s možností změny na jiný jazyk dle potřeby uživatele. Je pochopitelné, že mnoho lidí s mentálním postižením v České republice neumí jiný jazyk než jazyk národní. Proto ve snaze zprostředkovat aplikace této majoritní skupině se zachová jako hlavní jazyk aplikací čeština.

Konkrétní požadavky na i-CT Framework jsou popsány v kapitole 3.3 a požadavky na samotnou aplikaci rozpoznávající gesta lze nalézt v kapitole 3.5.

3.3 Specifikace požadavků na i-CT Framework

S odkazem na vhodnost přístupu s využitím návrhových principů počítačové terapie (viz úvod kapitoly 3) bude i při vývoji i-CT Frameworku kladena snaha zachovat všechny principy, které jsou vrámci tohoto projektu definovány (kapitola 2.2.6) [20]. Konkrétním cílem tohoto frameworku bude uspořádání všech aplikací do jednoho balíčku a dále jednoduchá správa všech uživatelů, kteří budou tyto aplikace používat (tzv. *user manager*, *app manager* a *security manager* pro další společná nastavení oprávnění). Mezi uživatele jsou řazeni administrátor, speciální lektoři a na závěr samotní lidé s mentálním postižením (dále klienti). Ve spolupráci s vedoucím práce byly nadefinované jednotlivé pravidla a možnosti (vycházejících z kapitoly 2.2.6 a 3.2), které se budou ve frameworku implementovat:

1. Administrátor bude moci vytvářet nové speciální lektory i dané klienty.

2. Speciální lektoři si budou moci zakládat nové klienty a také měnit údaje stávajícím klientům, které mají ve své správě. Mimo jiné také budou moci spravovat své podřízené

asistenty a jejich klienty. Tedy nemůže nastat situace, kdyby daný speciální lektor měnil údaje klientovi, kterého nezná. Tím je zaručená ochrana klientů a jejich změnou údajů od jednotlivých lektorů. Tento požadavek vychází i z praxe sociálních služeb.

3. Klientský profil se bude skládat z identifikačních údajů jako je jméno, příjmení atd., ale dále také ze tří dalších profilů a to zdravotní, vzdělávací a sociální profil. U každého klienta se bude zaznamenávat i historie jednotlivých rozšiřujících profilů a to z důvodu viditelného zlepšení a celkového vývoje daného klienta.

4. Zdravotní profil bude obsahovat záznamy o zdravotních problémech klienta a jejich anamnézy.

5. Vzdělávací profil bude obsahovat záznamy o potřebách klienta a dané kompetence těchto potřeb.

6. Sociální profil bude obsahovat informace o problematice s jídlem, hygienou a uživatelovy vztahy s lidmi v jeho blízkosti.

7. Tyto profily se z důvodu jednoduchosti budou moci přenášet z jednoho klienta na druhého při vytváření nového klienta.

8. Dále u klienta se také bude nastavovat seznam aplikací, které může používat a restrikce, které se budou vztahovat už k daným aplikacím (jako velikost písma, nastavení času, jak dlouho bude mít maximálně či minimálně klient spuštěnou aplikaci, atd.).

9. Z frameworku se budou jednotlivé aplikace spouštět a při tomto spuštění se předají informace o uživateli, který právě aplikaci spustil, a taky nastavení restrikcí, které byly zmíněny výše.

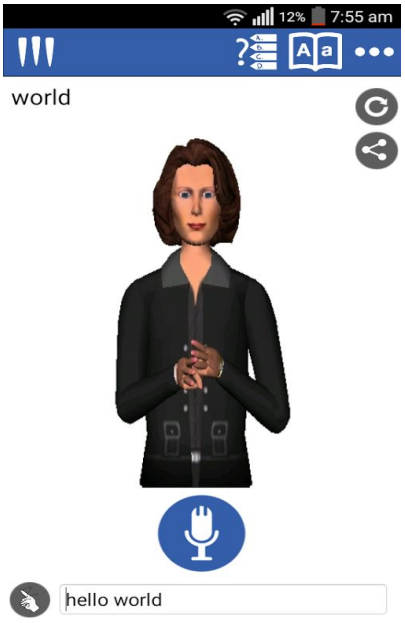
10. Po opuštění z nástavbové aplikace se popřípadě mohou uložit nové záznamy profilů klienta, které nastaly při jejím použití (např. do vzdělávacího profilu se vloží záznam, že se klient naučil další sadu slovíček v jedné z výukových aplikací).

3.4 Stávající aplikace pro práci se znakovým jazykem

V této podkapitole budou zhodnoceny existující aplikace *Mimix Sign Language Translator* a *MotionSavvy: UNI* s podobným zaměřením jako zde vyvíjená aplikace s překladem znakového jazyka. V závěru podkapitoly budou shrnuty výhody a nevýhody jednotlivých aplikací a čím byly inspirativní pro vývoj překladové aplikace. Přehledné porovnání aplikací bude zobrazeno v tabulce 3.1.

3.4.1 Mimix Sign Language Translator

Aplikace je určená pro rychlý překlad zapsaných vět do gest znakového jazyka (opačná úloha, než je tomu v případě aplikace této práce, přesto je vhodné pro srovnání přístupu řešení). Na obrázku 3.1 lze vidět dole pod znakujícím modelem ženy kolonku, kde lze zapsat v anglickém jazyce větu pro překlad. Při vkládání vět do kolonky je možné využít klávesnici složenou z obrázků písmen ve znakové řeči. Tato věta se následně přeloží způsobem, že postupně pro jednotlivá slova „žena“udělá gesto ve znakové řeči. Při překladu aktuálního slova, se objeví jeho význam v levém horním rohu aplikace pro přehled překládaných slov.



Obrázek 3.1: úvodní obrazovka aplikace *Mimix Sign Language Translator*

Pro jednoduchost je zde vytvořena varianta vkládání věty na přeložení z mluveného slova. Aplikace má uloženo přes 5500 znaků ASL[1]. Aplikace zabírá 183 MB fyzické paměti.

V této aplikaci je také několik možnosti, jak se znakovou řeč naučit. Toto se děje formou hry, kdy člověk, který se učí znakovou řeč, dostane na výběr 4 gesta daného slova. Za úkol má zvolit správné gesto, které v sobě skrývá význam zadaného slova. V průběhu jedné hry zkoušený dostane 7 slov, u kterých má určit správný význam. Dále se v této hře vedou statistiky o úspěšnosti určování správných gest k jednotlivým slovům.

Zhodnocení

Výhodou je rychlý překlad z textových vět do gest. Jedná se o svižnou aplikaci s velikou slovní zásobou znaků, ale jen ASL nikoliv jiných mezinárodních znakových jazyků, jako třeba čeština. Taky celá aplikace je pouze v jazyce anglickém, což v našem českém prostředí není vyhovující. Celá aplikace stojí 3.99$, kde se naimportují nejen další nové znaky, ale i přes tisíc nových frází. Rozšíří se i možnost testování znalostí znaků na 14 slov v jedné hře. Dále se zprovozní samotné překládání, jelikož ve volně dostupné verzi znakující „žena“ chybí. Mezi další nevýhodu patří, že se jedná o aplikaci pouze pro Android OS a nejedná se multiplatformní aplikaci.

3.4.2 Řešení MotionSavvy: UNI

Jedná se o skupinu lidí (tým *MotionSavvy*), která se zaměřila na vývoj rozpoznávání pohybu rukou pomocí speciálního hardwaru (*produkt UNI*) vycházejícího z projektu *LEAP Motion*[2]. Speciálním hardwarem se myslí snímající kontrolér (vychází z projektu *LEAP Motion*)

[1] Americká znaková řeč angl. *American Sign Language*.

[2] Jedná se o speciální hardware (soustava kamer) s vlastním SDK a je možné vyvíjet aplikace na desktopy PC s přímým využitím tohoto hardwaru. Projekt má vlastní dokumentaci a je multiplatformní (programovací jazyk Java, C#, Javascript, atd.). Dostupné z: https://www.leapmotion.com/.

Obrázek 3.2: Hardwarové řešení *UNI Pro*

a pouzdro na tablety, které nejen chrání tablet před pády, ale propojuje snímající kontrolér a samotný tablet. Kontrolér je složen z několika kamer, díky kterým je možné zaznamenat pohyb rukou v prostoru nad tabletem (vychází z řešení známého ze zařízení XBOX). Tento produkt se nazývá *UNI Pro*. Celou hardwarovou sestavu lze vidět na obrázku 3.2.

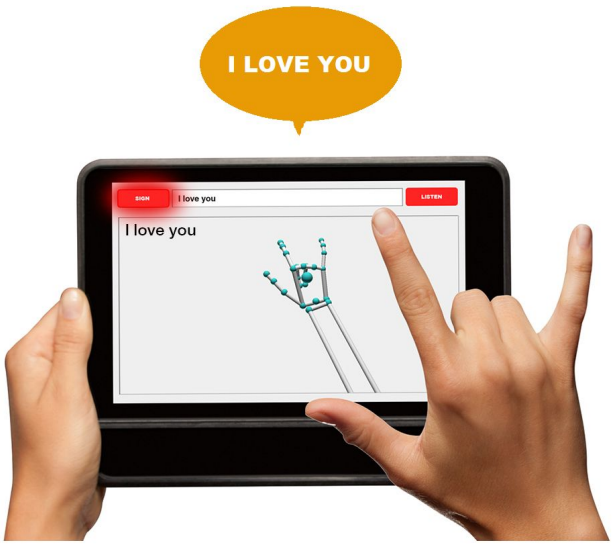
Dále tým *MotionSavvy* vytvořil aplikaci s názvem *UNI* na tablety, která velmi rychle zpracovává data získané z kontroléru. Aplikace funguje tak, že jakmile uživatel začne znakovat nad tabletem, začne zaznamenávat pohyb rukou na displej tabletu, vyznačuje jednotlivé znaky v pohybech rukou a také překládá znaky do souvislých vět. To všechno v reálném čase. Dále dokáže při překladu znaků na věty použít syntetický zvukový překladač[3]. Je zde také možné přidávat si vlastní nové znaky a tyto znaky sdílet přes cloud. Aplikace je schopna rozpoznat nad 2000 znaků ASL nebo SEE[4]. Dále má možnost zaznamenávat věty, které řekla osoba komunikující s neslyšícím. Tedy neslyšící rychle vidí reakci druhých osob psanou formou. V průběhu hovoru neslyšícího a slyšícího člověka se také zobrazují věty a tím krátkodobě zaznamenává historii hovoru.

Zhodnocení

Jedná se o velmi kvalitní produkt, který se snaží začlenit hluchoněmé lidi do normálního života. Mezi výhody patří rychlé zpracování znaků, následný překlad těchto znaků do vět, zvukový syntetizátor a zpracování zvuku druhých lidí do vět. Mezi nevýhody však patří to, že tablet musí využívat speciální hardware. V současné době *UNI* nelze použít pro jiné mobilní zařízení kromě tabletů s operačním systémem Android. Také cena produktu je dosti vysoká 499$ s tím, že každý měsíc za software je třeba zaplatit dalších 20$.

[3] Využívá *Dragon Speech Recognition Software*. Dostupné z: `http://www.nuance.com/dragon/index.htm`.

[4] Systém manuální řeči snažící se přesně reprezentovat gramatiku a slovíčka anglického jazyka. Angl. *Signed Exact English*.



Obrázek 3.3: Ukázka aplikace *UNI*

	Mimix Sign Language Translator**	MotionSavvy: UNI
Cena	96 Kč*	11976 Kč*
Multiplatformnost	Android	Android
Speciální hardware	ne	ano
Znakový jazyk	ASL	ASL, SEE
Český jazyk	ne	ne
Překlad znakového jazyka	ne	ano
Prostředí lidí se specifickými vzdělávacími potřebami	ne	ne

* Přibližná cena v českých korunách po převodu z Amerického dolaru (při kurzu $1:24 Kč).

** Tato aplikace se uvádí z důvodu popsaného v kapitole 3.4.1

Tabulka 3.1: Srovnání výhod a nevýhod popsaných aplikací

3.5 Specifikace požadavků na aplikaci Gesture Translator

Podobně jako v předchozí kapitole, tak se i u této aplikace bude vycházet z poznatků výzkumného projektu počítačové terapie a je tedy vyvíjena s ohledem na základní návrhové principy počítačové terapie (kapitola 2.2.6 a 3.2). Cílem této aplikace je vytvořit překladač znakového jazyka pracující v reálném čase (s možností úpravy i pro jiný znakový jazyk). Mezi konkrétní požadavky aplikace patří:

1. Aplikaci je možné spustit samostatně či z i-CT Frameworku a při její spuštění se předají základní informace z i-CT Frameworku o daném klientovi a časových restrikcích (jak dlouho má být aplikace minimálně spuštěná či jak dlouho maximálně může být aplikace spuštěna).

2. Cílem této aplikace je zaměřit se na překlad znakového jazyka od konkrétního uživatele (osoby se speciálními vzdělávacími potřebami), kde se očekává, že gesta prováděná

tímto uživatelem mohou být velmi individuální od obecného vzoru gesta předkládaného českým znakovým jazykem. Tedy primárně trénování znaků se vybírá ze vzorků daného jedince.

3. Dále aplikace bude mít tyto části: přidávání nových znaků, překlad znaků, přehrávání provedení znaků a jednoduché nastavení aplikace.

4. Přidávání znaků bude moci pouze asistent či administrátor. Vlastní kamerou zařízení se bude snímat osobu, která provede několikrát stejný znak, zadáním významu tohoto znaku a následným potvrzením se znak uloží do databáze. Od této chvíle aplikace bude moci rozpoznávat tento znak. Při nainstalování tohoto překladače bude aplikace rozpoznávat několik předdefinovaných základních znaků z ČZJ.

5. Druhá část aplikace slouží k samotnému rozpoznávání znaků znakového jazyka jedince. Kamerou se bude snímat znakující osoba. Předpokládá se, že překlad bude probíhat po jednom znaku. Aplikace přeloží význam jednotlivých znaků v realném čase a tento význam vypíše na obrazovku mobilního zařízení.

6. Přehrávání provádění znaků je část, kde se po zvolení znaku postupně znak přehraje na obrazovce zařízení a tak se prezentuje, jak se znak provádí. Přehrávání znaku bude možné pozastavit, spustit od místa pozastavení a znovu znak přehrát.

7. V aplikaci bude možné vybrat konkrétní skupinu znaků, které bude aplikace aktuálně rozpoznávat pro daného uživatele. Ostatní znaky aplikace bude pak ignorovat.

8. Po přihlášení z i-CT Frameworku v roli speciálního lektora či administrátora je možné v nastavení aplikace zadat, zda-li uživatel, který není přihlášený přes i-CT Framework, může editovat gesta nahrané v aplikaci. Dále pro všechny uživatele s možností editovat gesta je možné nastavit, zda se budou body získané sekvence od uživatele filtrovat od začátku startu pozice gesta, či se zachovají všechny body od startu snímání znakující osoby.

Kapitola 4

Návrh

V této kapitole bude představen návrh implementace systému - i-CT Frameworku a aplikace Gesture Translator na základě stanovených návrhových principů vycházejících z výzkumného projektu Počítačové terapie (kapitola 2.2.6).

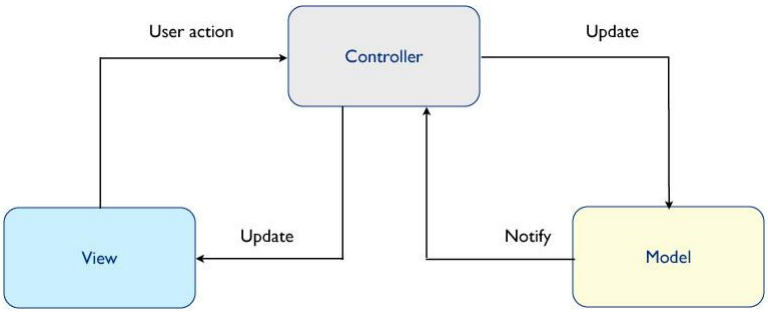
4.1 Architektonický vzor Model-View-Controller

Jedná se o architektonický vzor, který rozděluje aplikaci do tří základních komponent: Model (data aplikace), View (vizuální reprezentace dat) a Controller (řídí komunikaci mezi komponentami Model a View). U těchto komponent je snaha, aby modifikace jedné z nich měla minimální vliv na ostatní zbylé. Tedy každá komponenta má v režii určitou část aplikace (obrázek 4.1):

1. **Model** – komponent zahrnující funkcionalitu pracující s daty aplikace (ukládání, načítání dat), obvykle komunikuje s databází.

2. **View** – komponent, který dostává od Controlleru údaje určené na vypsání. View pouze reprezentuje tyto data, proto je důležité, aby znal jejich typ a formát. Avšak neví odkud tyto data pocházejí.

3. **Controller** – komponent, který zpracovává požadavky přicházejících z uživatelského rozhraní (View). Z těchto požadavků je schopen rozpoznat, které data z Modelu budou vyvolány či modifikovány a kterém pohledu (View) se eventuálně tyto data zobrazí.



Obrázek 4.1: Architektonický návrhový vzor MVC

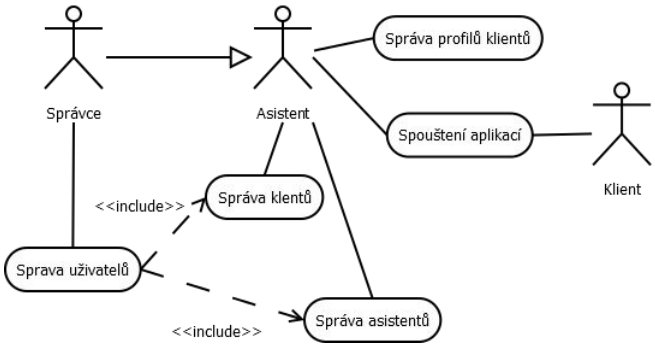
4.2 Návrh i-CT Frameworku

Na vývoji aplikace i-CT Frameworku pracoval autor této práce ale také jeho kolega Jan Kalina [22]. Proto návrh byl společný a po provedení návrhu se rozdělila implementační část mezi tyto dva studenty. Tedy v této kapitole se nejen shrne návrh samotná aplikace i-CT Framework, ale i návrh celkového systému nástavbových aplikací a i-CT Frameworku.

Návrh aplikace i-CT Framework vychází ze specifikace popsané v kapitole 3.3 (to zahrnuje již zde uvedenou potřebu návrhu tzv. *user manageru*, *app manageru* a *security manageru* do jedné centrální aplikace, viz následující popis). Dá se tedy říct o této aplikaci, že se jedná o centrální přihlašování klientů, jejich asistentů či správce a také úložiště informací o těchto klientech a jejich asistentech v rámci celého systému aplikací pro práci s lidmi s mentálním postižením. Tím se odstraní nutnost vyvíjet podobné úložiště pro každou aplikaci systému zvlášť.

Další z hlavních cílů aplikace i-CT Frameworku je také možnost spouštět výukové aplikace pro daného klienta. Toto bude možno provádět v režimu správy nastavení aplikace a v režimu užívání samotným klientem. Pokud se výuková aplikace spustí v režimu klientova užívání je nutno zabránit tomu, aby se klient dostal do nastavení aplikace a tím toto nastavení mohl měnit. Kdežto je-li aplikace spuštěna pod režimem správy asistentem, je možné měnit dané nastavení aplikace pro konkrétní klienta i bez toho, aby klient byl přihlášený. Podobně je tomu i v případě, že asistent spustí aplikaci pro klienta bez jeho nutného přihlášení. Podrobné informace o této komunikaci je popsána v kapitole 4.2.2.

Další případ užití iCT-Framewroku je správa uživatelů (klienti, asistenti a správce) systému a u klientů navíc správa jejich záznamů profilů (sociální, zdravotní i vzdělávací). Asistent může spravovat uživatele a jejich záznamy profilů v případě, že se jedná o jeho klienty či své podřízené asistenty a jejich klienty. Správce může spravovat všechny uživatele systému.



Obrázek 4.2: Diagram případu užití aplikace i-CT Frameworku.

Dále z důvodu, že cílová skupina uživatelů jsou lidé s mentálním postižením, bude navržen i způsob, jak zabránit nechtěnému přístupu klientů k citlivým datům jiných klientů či asistentů a podobně i zabránění případů, kdy sám klient by mohl změnit nastavení aplikací určených pro něho samotného či pro jiné klienty.

Z toho vyplývá, že prováděný návrh se zaměří na samotnou komunikaci mezi jednotlivými výukovými aplikacemi celého systému a i-CT Frameworkem (i komunikaci opačným směrem) tak, aby komunikace z výše popsaných důvodů byla bezpečná a nedala se třeba

i nechtěně zneužít. Dále se zde zahrne přístup a práce s daty iCT-frameworku. Také se předloží tok přepínání jednotlivých obrazovek i-CT Frameworku.

4.2.1 Rozdělení celkového systému aplikací a i-CT Frameworku

Přirozeně je představitelné, že celý systém by mohla být pouze jedna jediná aplikace, která by byla rozdělena do dílčích logických celků, které by představovali jednotlivé výukové aplikace a samotný i-CT Framework. V tomto případě by se musel celý systém instalovat najednou se všemi dostupnými jednotlivými výukovými aplikacemi a nebyla by možnost doinstalovat nově vyvinuté další celky a nebo instalovat jen určité výukové aplikace podle potřeby místo všech najednou.

Dále u většiny mobilních platforem není povoleno aplikacím zapisovat do adresářů se svými spustitelnými soubory. U multiplatformních hybridních aplikací je možné využít datových adresářů, což by problém vyřešilo, ale ztrácí se tím schopnost přímo využívat nativních kódů platformy (lze to jen do jisté míry přes cordova pluginy).

Nakonec se tedy zvolil přístup, kdy systém je navržen jako skupina zcela nezávislých aplikací. Vzniká tedy potřeba stanovit a navrhnout způsob vzájemné komunikace mezi těmito aplikacemi.

4.2.2 Komunikace i-CT Frameworku a výukových aplikací

V rámci platformy iOS je možné komunikovat mezi aplikacemi pomocí odkazu URL s URL schématem registrovaným cílovou aplikací. Tento druh komunikace je podporován i na platformně Android a proto využijeme komunikace přes URL i v této práci.

Na začátku URL adresy je předpona tvořena URL schématem, které je registrované u jedné z nainstalovaných aplikací v zařízení. Tedy při vyvolání URL adresy s daným URL schématem se právě otevře konkrétní jedna aplikace, která může dále toto URL zpracovat.

Navíc je možno do URL adresy za konkrétní URL schéma vložit libovolný text, což je právě vhodné pro uložení dat, které se budou mezi aplikacemi posílat.

Ve fázi návrhu se rozhodlo, že i-CT Framework a výukové aplikace budou na sobě nezávislé aplikace, kde každá z těchto aplikací bude mít zaregistrované své identifikační URL schéma a tím se získá možnost vzájemné komunikace přes URL. Z aplikace i-CT Frameworku bude možno spouštět pro daného klienta konkrétní výukové aplikace přes URL s daným URL schématem aplikace a při vyvolání výukové aplikace se předají informace nutné k fungování dané aplikace pro konkrétního klienta. Dále pak výuková aplikace při svém ukončení přesměruje uživatele do i-CT Frameworku a podobně i ona může i-CT Frameworku zaslat informace přes URL.

Při vyvolávání výukové aplikace z i-CT Frameworku pro konkrétního klienta se navíc může předat i informace o uživateli, který je aktuálně přihlášen v i-CT Frameworku (zpravidla speciální lektor klienta či administrátor). Tento přístup přináší tu výhodu, že speciální lektor (či administrátor) se již nemusí znovu přihlašovat ve výukové aplikaci.

Dále jak již bylo výše naznačeno před spuštěním výukové aplikace z i-CT Frameworku je přihlášen speciální lektor (popřípadě administrátor), který již v i-CT Frameworku vybere konkrétního klienta, pro kterého danou aplikaci spustí. Navíc aplikaci může spustit celkem ve dvou režimech:

- Režim správy - v případě, že je nutné nastavit ve výukové aplikaci prostředí pro konkrétního klienta.

- Režim klienta - v případě, kdy klient dostává zařízení se spuštěnou výukovou aplikací bezpečném režimu, tj. klient nemůže ať už úmyslně či neúmyslně zasahovat do konfigurace aplikace.

Nakonec je nutné zmínit způsob vypínání výukových aplikací. V prvním režimu, kdy spuštěnou aplikaci ovládá speciální lektor, by vypnutí aplikace mělo být snadné, kdežto v druhém režimu by pro klienta mělo být nemožné aplikaci opustit. K tomuto se jeví jako vhodné použít odemykací gesta, podobné těm, které se využívají na zamykací obrazovce mobilních zařízení. Toto lze ale navíc změnit v případě, kdy pro daného klienta je možné aplikace samostatně opouštět.

Definování URL parametrů pro přechod z i-CT Frameworku do výukové aplikace

Pro přechod do výukové aplikace z i-CT Frameworku se definují tyto URL parametry:

- `return` - URL schéma pro navrácení do i-CT Frameworku z výukové aplikace
- `userId` - ID zvoleného uživatele (klienta)
- `userLogin` - přihlašovací jméno zvoleného uživatele
- `userName` - křestní jméno zvoleného uživatele
- `userSurname` - příjmení zvoleného uživatele
- `userLevel` - práva zvoleného uživatele (`ADMIN` - administrátor, `ASSISTANT` - speciální lektor, `CLIENT` - klient)
- `userManagerId` - ID nadřízeného u zvoleného uživatele (speciální lektor u klienta)
- `userLanguage` - jazyk zvoleného uživatele
- `loggedId` - ID přihlášeného uživatele (speciální lektor, který aplikaci spustil)
- `loggedLogin` - přihlašovací jméno přihlášeného uživatele
- `loggedGesture` - přihlašovací gesto přihlášeného uživatele (je-li požadováno pro opuštění aplikace)
- `loggedName` - křestní jméno přihlášeného uživatele
- `loggedSurname` - příjmení přihlášeného uživatele
- `loggedLevel` - úroveň přihlášeného uživatele (viz `userLevel`)
- `loggedManagerId` - ID nadřízeného u přihlášeného uživatele (nadřízený asistenta)
- `loggedLanguage` - jazyk přihlášeného uživatele
- `canEscapeApp` - značí, jestli zvolený uživatel může výukovou aplikaci opustit bez gesta přihlášeného uživatele
- `allowedTime` - maximální čas, kdy výuková aplikace bude spuštěna. Po uplynutí tohoto času výuková aplikace sama zpětně spustí i-CT Framework. Parametr je celé číslo v minutách, -1 pro případ, kdy čas použití aplikace není omezen.

- `manage` - jedná se o volitelný parametr,který značí, jestli je aplikace spuštěná v režimu správy (`true`) či v režimu uživatele (`false` nebo parametr chybí)

```
myapplication://?return=ictframework&userId=2&userLogin=klient&userName=
    Jan&userSurname=Hodny&userLevel=CLIENT&userManagerId=1&userLanguage=cs
    &loggedId=1&loggedLogin=asistent&loggedGesture=1234&loggedName=Marie&
    loggedSurname=Krasna&loggedLevel=ASSISTANT&loggedManagerId=4&
    loggedLanguage=en&canEscapeApp=false&allowedTime=30&manage=false
```

Ukázka kódu 4.1: Příklad URL adresy pro spuštění výukové aplikace

V ukázce kódu 4.1 lze vidět URL adresu, kterou se spustí výuková aplikace z i-CT Frameworku. Výukovou aplikaci v tomto příkladě spouští speciální lektor Marie Krásná (uživatelské ID je 1, lokalizace anglická) pro klienta Jana Hodného (s ID 2, lokalizace česká). Marie je přímá lektorka klienta Jana. Nadřízeným Marie je uživatel v roli správce s ID 4.

Pro vypnutí výukové aplikace je nutné, aby bylo zadáno heslo lektora. Dále aplikace poběží maximálně 30 minut, kdy po uplynutí tohoto časového intervalu se zpátky přepne na úvodní přihlašovací obrazovku i-CT Frameworku a nakonec je vidět, ze je aplikace spuštěná bez režimu správy (teď bez možnosti editace).

Definování URL adres pro přechod z výukové aplikace do i-CT Frameworku

Pro přechod z výukové aplikace zpět do i-CT Frameworku je třeba ověřit uživatele, který výukovou aplikaci spustil tak, aby se případný klient nedostal k účtu speciálního lektora. V případě, že výuková aplikace nepodporuje možnost zadat odemykací gesto speciálního lektora, tak po vyvolání i-CT Frameworku, by se mělo provést odhlášení speciálního lektora.

Pokud je systém používán na platformně iOS je vhodné použít Asistovaný režim systému iOS. S tímto režimem je možné předat zařízení klientovi. Samotný návrat do i-CT Frameworku z výukové aplikace je pak podobný jako, když je použit režim správy.

Asistenční režim systému iOS zabraňuje uživateli opustit aplikaci a lze také nastavit, že nefungují i jiná hardwarová tlačítka zařízení. Při spuštění výukové aplikace je možné trojím stisknutím tlačítka menu tento Asistenční režim spustit. Pro ukončení výukové aplikace je tedy nutno Asistenční režim vypnout. Vypnutí Asistenčního režimu se provede podobně jako v minulém případě trojím stisknutím menu a navíc zadáním předem nadefinovaného hesla speciálním lektorem. Pokud je systém používán na zařízení s OS Android, je opouštění možné zabránit přes nastaveni přímo v profilu uživatele a také při využití Kiosk pluginu (součást i-CT Frameworku), který přebírá kontrolu nad HW tlačítky v zařízení.

Pro přechod zpět do i-CT Frameworku jsou definované tyto URL adresy:

- `ictframework://?fromManage=true` - v případě, když se vrací speciální lektor z výukové aplikace (není nutné ho tedy odhlašovat)
- `ictframework://?afterGesture=true` - v případě, když se vrací uživatel zpět po zadání gesta speciálním lektorem (opět lze pokračovat pod účtem lektora)
- `ictframework://?becauseNotInKiosk=true` - v případě, kdy se uživatel vrací v iOS prostředí těsně po vypnutí Asistenčního režimem sytému iOS.

Způsob registrace výukové aplikace do i-CT Frameworku

Mimo jiné je také možné přes URL zaregistrovat výukovou aplikaci do i-CT Frameworku. Aplikace, která se registruje do i-CT Framework, se přidá do nabídky všech přístupných výukových aplikací do i-CT Frameworku a v URL předává své URL schéma a také svůj název (viz ukázka kódu 4.2).

```
ictframework://?action=register&scheme=URLschemaMyApp&name=
    NazevMojiVyukoveAplikace
```

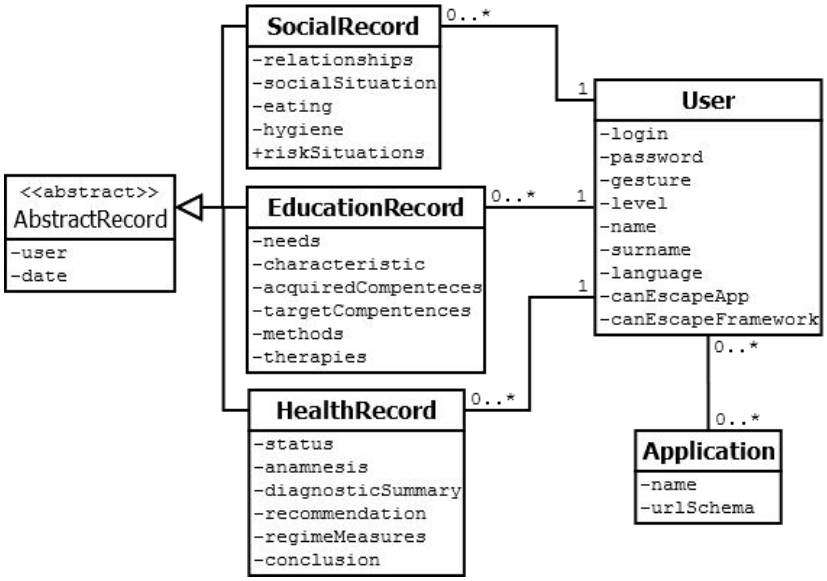
Ukázka kódu 4.2: Příklad URL adresy pro registraci výukové aplikace

4.2.3 Návrh databáze i-CT Frameworku

Jedním z cílů i-CT Frameworku je uchovávat informace o klientech, jejich speciálních lektorech (asistentech), administrátorech aplikace a popřípadě i výukových aplikacích registrovaných v i-CT Frameworku. To se provede pomocí databáze.

Nejvíce je však podstatné uchovávat informace o klientech, u kterých navíc sledujeme jejich profily: zdravotní, sociální a vzdělávací. U každého z těchto profilů si ukládáme jednotlivé záznamy v průběhu času k danému klientovi. Z podstaty věci záznamy jednotlivých profilů jsou dosti odlišné a proto pro každý z těchto profilů byla vytvořena jedinečná třída (objektově orientovaný návrh databáze). Ovšem i přes značnou odlišnost záznamů těchto profilů mají záznamy společné vlastnosti a to kdy byl záznam vytvořen a pro jakého uživatele byl vytvořen. Proto se zavádí abstraktní nadtřída `AbstractRecord`, ze které jsou zbylé třídy tvořeny.

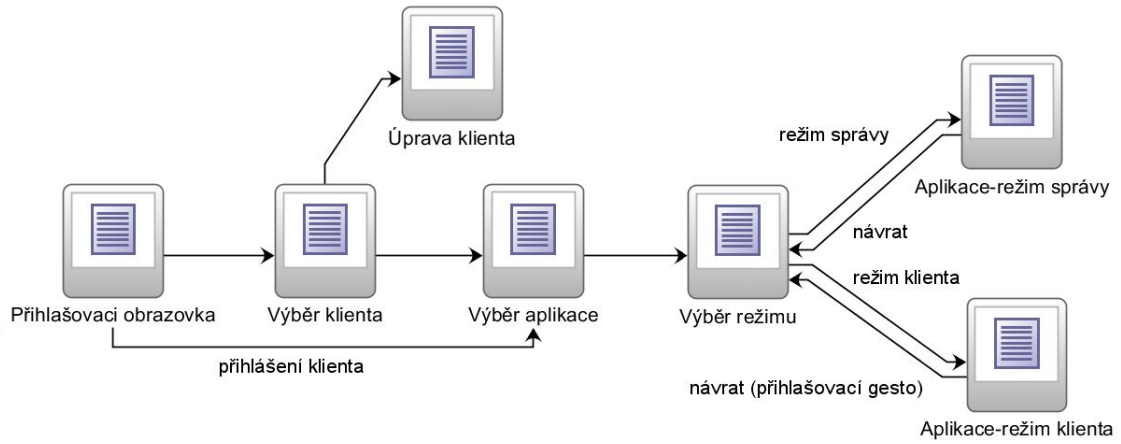
Dále je také důležité uchovat v databázi informaci reflektující relaci mezi uživateli a registrovanými výukovými aplikacemi. Tedy vzniká mezi těmito entitami vztah N:M - každému z uživatelů může být přiřazeno více výukových aplikací a každá z výukových aplikací může být přiřazena k více uživatelům.



Obrázek 4.3: Diagram tříd modelu domény pro datovou část aplikace

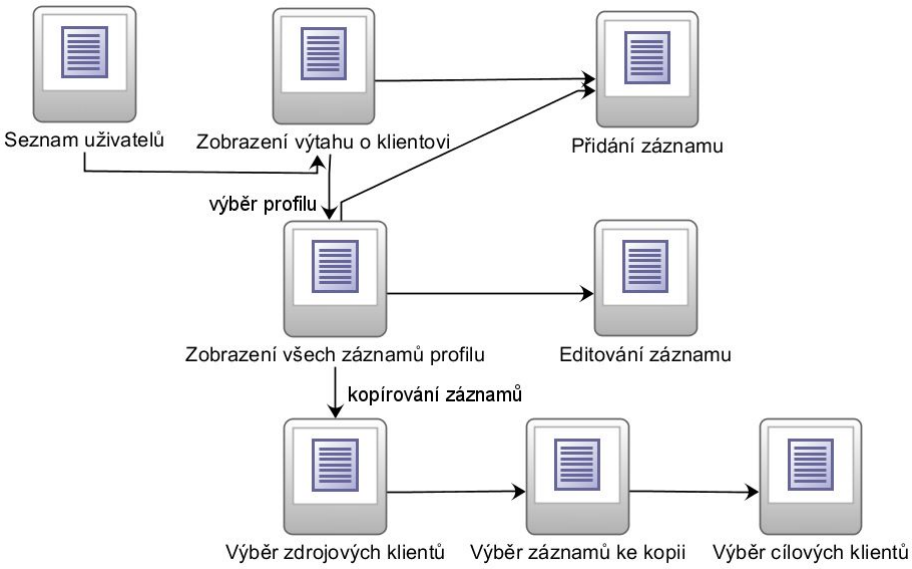
4.2.4 Návrh posloupnosti obrazovek i-CT frameworku

Po spuštění i-CT Frameworku se zobrazí uživateli obrazovka s přihlašovacím formulářem. Po přihlášení asistenta se objeví na další obrazovce seznam uživatelů, které má asistent (např. pedagog, soc. pracovník apod.) na starosti. Po výběru klienta může asistent upravit informace o klientovi či zobrazit jeho aplikace. V případě, kdy asistent vybírá pro klienta danou výukovou aplikaci, může na další obrazovce zvolit v jakém režimu aplikaci otevře (režim správy či režim klienta). Při výběru režimu klienta předá zařízení klientovi. Po uplynutí výuky dostává asistent zpět zařízení od klienta a prokazuje se ověřovacím gestem při návratu do i-CT Frameworku. Tím se zabrání, aby např. mentálně postižený klient nemohl manipulovat s účtem asistenta v i-CT Frameworku. Také je možnost pro samostatného klienta, aby se přihlásil sám do i-CT Frameworku. Pak je mu předložena sada jeho aplikací, které může spouštět. Na obrázku 4.4 je možné tento sled obrazovek pozorovat.



Obrázek 4.4: Diagram posloupností obrazovek pro spuštění výukových aplikací

Při zobrazení detailu uživatele je možné spatřit jeho zdravotní, sociální a vzdělávací profil. U každého z těchto profilů lze vidět několik posledních záznamů a při vybrání jednoho z nich se zobrazí obrazovka se všemi záznamy daného profilu. Zde je možné profily editovat, přidávat či mazat. Navíc se zde uchovává vlastnost kopírování profilů od jiného klienta a to z důvodu, kdy někteří klienti mohou prožívat podobné životní záležitosti, a tedy se tím zefektivňuje práce pro daného asistenta, který tyto podobnosti zná. Při výběru kopírování záznamů profilu se zobrazí asistentovi obrazovka se seznamem klientů, odkud se budou záznamy kopírovat. Po vybrání klientů se objeví seznam jejich záznamů a zde si asistent dané záznamy může zvolit. Nakonec vybere ze seznamu klientů klienty, kterým záznamy bude kopírovat a po potvrzení se tyto záznamy k daným klientům uloží. Na obrázku 4.5 lze vidět tuto posloupnost obrazovek.



Obrázek 4.5: Diagram posloupností obrazovek pro kopírování profilů klientům

4.3 Návrh aplikace Gesture Translator

Součástí implementace této práce je také aplikace Gesture Translator, která je určená pro překlad znakového jazyka v prostředí osob se speciálními vzdělávacími prostředky, kde se očekává, že každá osoba pro význam stejného gesta, provádí gesto individuálním způsobem (viz kapitola 3.5). V této kapitole bude navržen způsob fungování této aplikace.

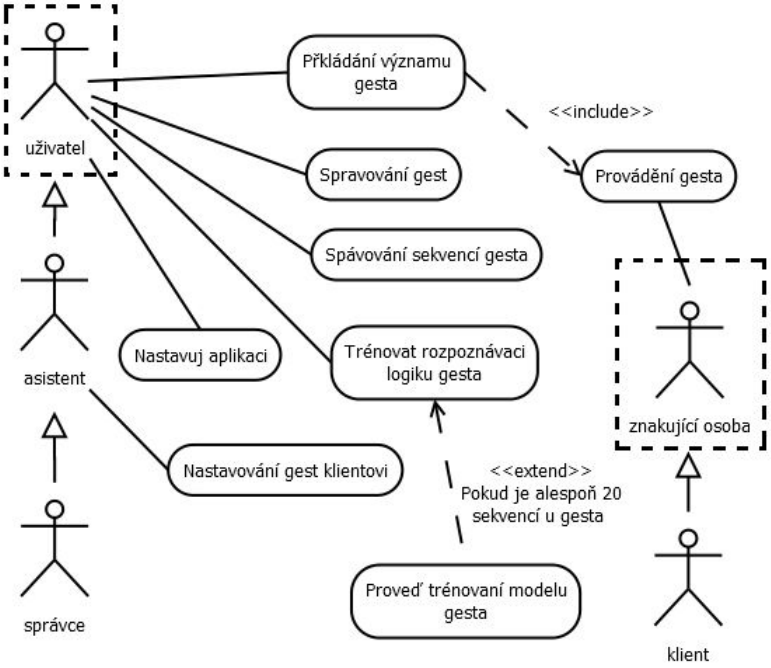
Diagram případů užití aplikace Gesture Translator lze vidět na obrázku 4.6, kde aktéři uživatel a znakující osoba (označeni čárkovanými obdélníky), jsou aktéři rozšíření aplikace Gesture Transaltor. V případě tohoto rozšíření, které bylo navrženo a implementovanou nad rámec zadaní, je gesto trénováno ze sekvencí nejen jedné konkrétní osoby se speciálními vzdělávacími potřebami a tvaru jeho individuálního gesta, ale obecného překladu daného gesta. Avšak toto rozšíření není hlavním cílem této práce. Proto se předpokládá, že uživatel aplikace nejčastěji bude právě asistent, který se stará o daného klienta, a znakující osoba bude klient se speciálními vzdělávacími potřebami.

Hlavním cílem aplikace Gesture Translator bude překlad jednotlivých znaků (gest) znakového jazyka prováděných osobou se speciálními vzdělávacími potřebami. Tedy uživatel (nejčastěji asistent), který bude chtít znát význam prováděného gesta, bude mobilní zařízení obsluhovat a kamerou mobilního zařízení snímat osobu, která gesto realizuje. Po realizaci gesta aplikace přeloží jeho význam a následně tento význam vypíše na obrazovku zařízení. Tento hlavní případ užití se nazve **překládaní významu gesta**.

Dalším případem užití je možnost **spravování gest** uložených v aplikaci, tj. editace a mazání stávajících gest uložených v aplikaci či přidání nového gesta do aplikace.

Uživatel (nejčastěji asistent) může také **spravovat sekvence** k danému gestu. Zde je třeba říct, že mimo základní operace, jako je sekvenci smazat či přidávat novou sekvenci, je možnost přehrát video s uloženou sekvencí (pro představu o způsobu provedení gesta).

Jeden z dalších případů užití je **trénování rozpoznávací logiky** pro konkrétní gesto, ale to jen tehdy, když je vytvořených alespoň 20 sekvencí k danému gestu (viz kapitola 4.3.1).



Obrázek 4.6: Diagram případů užití aplikace Gesture Transaltor

V **nastavení aplikace** bude možné uživatelem (asistentem) zvolit, zda-li bude chtít při překladu gesta filtrovat body získané sekvence od počátečního místa rukou prováděného gesta, či bude chtít zpracovat všechny body poloh rukou sekvence od startu provádění gesta (viz kapitola 5.2.3 v sekci „Filtrování získaných středů rukou“).

Poslední případ užití souvisí jen s asistenty daných klientů používající i-CT Framework. Takový asistent může vybrat danému klientovi konkrétní sadu gest, která obsahuje klientova gesta, tj. klient prováděním gest vytvářel sekvence gesta a tyto sekvence asistent přidával do aplikace. To má tu výhodu, že při překládání významu gesta, respektive po získání sekvence gesta, aplikace vybírá jen možnou shodu s přidělenými klientovými gesty.

4.3.1 Rozpoznávací logika aplikace Gesture Translator

Skryté Markovovi modely (kapitola 2.2.7 v sekci „Úvod do Skrytých Markovský Modelů“) jsou řešením, které se využívá pro různé druhy překladačů a to i v případě překladu znakového jazyka. Ve zmíněné kapitole jsou předloženy konkrétní otázky, jak HMM lze použít. Pro případ této práce je zcela přirozené využít otázku trénování HMM a také evaluaci, která říká, jak moc je pravděpodobné, že daná sekvence je generovaná daným HMM. Poslední otázka, dekódování sekvence stavů, je v tuto chvíli nepotřebná.

Byl tedy navržen koncept, kdy každé gesto má svůj vlastní HMM. Tato HMM je natrénována ze sekvencí určujících význam konkrétního gesta. Pro trénink HMM je důležité, aby byl dostatečný počet sekvencí, z kterých se HMM natrénuje. Tato hodnota byla stanovena v průběhu vývoje a testování celého mechanismu rozpoznávání (viz kapitola 6.3), kdy nejvíce optimální se ukazuje hodnota kolem 20 sekvencí.

Sekvencí gesta je zde myšleno taková posloupnost pozic rukou v čase, jaká se získávala postupným realizováním gesta. Pro jednu jednotku času se tedy skládá sekvence ze dvou informací a to konkrétně pozice levé ruky a pozice pravé ruky. Informace o hlavě se bude

vynechávat, jelikož pohyby hlavy se po celou dobu provádění gesta téměř nemění (není zde myšlena mimika obličeje, ale výrazné pohyby hlavy doleva či doprava).

Návrh algoritmu rozpoznávací logiky

V této kapitole stručně popsán algoritmu pro způsob rozpoznávání gest.

Vstupem pro rozpoznání gesta tedy bude sekvence hledaného gesta. Tato sekvence se předloží všem gestům, které mají natrénované HMM a pomocí evaluace HMM se stanoví jaká je pravděpodobnost, že dané HMM vygenerovalo vstupní sekvenci. Nakonec se vybere gesto, kde HMM mělo pro vstupní sekvenci největší pravděpodobnost ze všech.

Návrh abecedy sekvencí gest v aplikaci Gesture Translator

HMM je definováno celkem dvěma maticemi a jedním vektorem. Jedna z těchto matic obsahuje pravděpodobnosti generování symbolů sekvence gesta z daného stavu (jedná se o matici B popsanou v kapitole 2.2.7 v sekci „Úvod do Skrytých Markovský Modelů“).

Vzniká tedy problém, jelikož sekvence je složená z pozic rukou (středu ruky), což je dvourozměrná informace (souřadnice x a y obrázku). Je tedy potřeba navrhnout převod této dvourozměrné informace na jednorozměrnou informaci. To by se dalo provést například speciální tabulkou, která by obsahovala složený identifikátor ze souřadnic x a y a vracela by jeden ze symbolů abecedy sekvencí. Tím by ale vzniklo velké množství symbolů abecedy sekvencí, například při představě rozlišení 720x480 kamery mobilního zařízení by vzniklo 345600 možných symbolů abecedy.

Proto bylo navrženo zredukování těchto symbolů abecedy pomocí jedné z cluster funkcí (například pomocí algoritmu *k-means*), která rozdělí obrazovku do oblastí (clusterů) podle všech bodů sekvencí, ze kterých se gesto skládá. Výsledek z cluster funkce jsou středy oblastí takto rozdělené obrazovky. Samotný převod bodu sekvence probíhá tak, že zjistíme nejbližší střed z množiny oblastí obrazovky gesta k tomuto bodu, ten je převeden do jednorozměrné informace, která pak reprezentuje symbol abecedy sekvencí gest (viz kapitola 5.2.3).

4.3.2 Rozpoznávací mechanismy pro polohy rukou a hlavy v OpenCV

Návrh aplikace Gesture Translator se také zaměřuje na samotný způsob rozpoznávání polohy rukou a hlavy. OpenCV předkládá řadu funkcí a tříd, které se zaměřují na rozpoznávání a sledování objektů ve videu (kapitola 2.2.7 v sekci „Metody pro rozpoznávací mechanismus s OpenCV“). V této kapitole se uvedou různé typy detektorů a krátce se popíše jejich očekávaná činnost, kterou by měly být využity při rozpoznávání rukou a hlavy.

Po konzultaci s panem Kamilem Behúňem z UPGM VUT FIT byly navrženy tyto detektory:

- **Detektor obličeje** (angl. face detector) - pomocí tohoto detektoru by mělo být možno zjistit pozici hlavy v obraze
- **Detektor barvy kůže** (angl. skin detector) - pomocí tohoto detektoru se budou vyhledávat oblasti na obrázku, které splňují podmínky barvy kůže
- **Detektor pohybu** (angl. move detector) - díky tomuto detektoru je možné zjistit objekty, které se ve videu pohybují

Dále je pak možné tyto detektory kombinovat a tím ještě přesněji získávat pozici rukou a hlavy. Výsledná implementace detektoru pozic rukou a hlavy je popsána v kapitole 5.2.3.

4.3.3 Návrh struktury databáze Gesture Translator

V aplikaci Gesture Translator, jak bylo zmíněno již v úvodní kapitole 4.3 pojednávající o návrhu této aplikace, je možno spravovat gesta aplikace, popřípadě sekvence daného gesta. Nejenom tyto data jsou dynamická a proto vyvstala potřeba navrhnout databázi aplikace Gesture Translator, kde se dané změny prováděné uživatelem mohou ukládat.

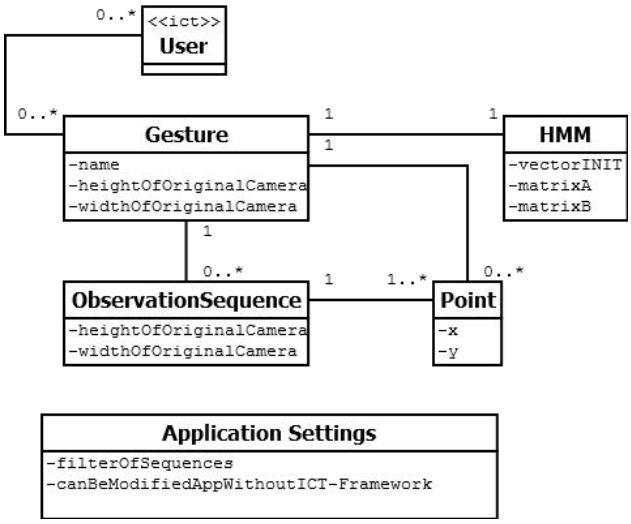
Původní návrh databáze byl několikrát změněn v závislosti na vzniklých problémech při implementaci. Proto je zde uveden a popsán poslední návrh databáze aplikace Gesture Translator.

Na obrázku 4.7 lze vidět přehledový diagram, ze kterého struktura databáze vychází. Entita uživatele je umístěna v i-CT Frameworku tak, aby nevznikala zbytečně duplicitní data. Uživatel může mít přiděleno několik svých gest, ze kterých se pak vybírá název gesta při překladu získané sekvence reprezentující dané gesto. Každému gestu je přidělena rozpoznávací logika v podobě HMM. HMM je samotná entita, která obsahuje parametry modelu v podobě matice A (matice pravděpodobnosti přechodů stavů mezi stavy), matice B (matice pravděpodobnosti generování symbolů řetězce u konkrétních stavů) a vektor π (inicializační vektor pro HMM - s jakou pravděpodobností se začíná generovat daná sekvence gesta z daných stavů). Tyto matice je třeba ukládat do databáze, protože výpočet samotného modelu HMM je náročná operace a tedy při každém vypnutí aplikace by se tyto data jinak ztrácela.

Jednotlivé sekvence, z kterých jsou gesta trénována, jsou reprezentována entitou *Observation Sequence*. Každá ze sekvencí obsahuje pozice rukou, ze kterých je složena. Proto se zavádí entita *Point*, která v sobě zachovává informaci o středu pozice ruky. Je možné si povšimnout, že gesto i sekvence obsahuje „stejné“ vlastnosti a to *heightOfOriginalCamera* a *widthOfOriginalCamera*. Tyto hodnoty nejsou duplicitní, říkají jen, při jakém rozlišení kamery vznikaly jednotlivé sekvence gesta a u entity gesta tyto hodnoty prozrazují velikost rozlišení, podle kterého se budou ostatní sekvence normalizovat při operaci trénování HMM (viz kapitola 5.2.2 sekce „Implementace rozpoznávací logiky v Gesture Tranlator Manager“).

Dále je možné z obrázku zpozorovat vztah mezi entitou gesta a entitou bodu. Totiž v gestu je zachována informace o středu oblastí (tedy abecedy sekvencí jejího HMM), podle kterých se převádějí body sekvencí gest.

Nakonec se zavádí entita nastavení aplikace, kde se uchovává informaci o tom, jestli se dané sekvence gesta mají filtrovat či nikoliv a hodnota, která prozrazuje jestli uživateli, který není přihlášený přes i-CT Framework, je povoleno spravovat gesta aplikace.



Obrázek 4.7: Přehledový návrhový diagram tříd pro model dat aplikace Gesture Translator. (Stereotyp «ict» označuje entitní množinu existující v rámci i-CT frameworku.)

4.3.4 Systém aplikací Gesture Translator Manager a Gesture Translator App

Při vývoji aplikace Gesture Translator vznikly situace, které jsou detailneji uvedeny v kapitole 5.2.1. V krátkosti lze říci, že MAF aplikace vyvinuté nástrojem Oracle MAF, mají jisté omezení při použití cordova pluginů. V této práci byl vyvíjen cordova plugin, který pracoval s knihovnou OpenCV, pomocí které se řešila detekce rukou a hlavy. Při instalování pluginu do MAF aplikace se nastavovala průhlednost obrazovek MAF aplikace tak, aby detekovaný obraz kamery pomocí OpenCV byl zobrazen uživateli. Přestože nastavení proběhlo zcela v pořádku, daný efekt se neprojevil v MAF aplikaci. Plugin byl zkoušen i v prostředí Apache Cordova Framework, kde se kýžená vlastnost s úspěchem projevila.

Po konzultaci tohoto problému s vedoucím této diplomové práce bylo navrženo vytvořit celkem dvě aplikace. Jedna z nich, Gesture Translator Manager (dále jen manager), by implementovala výše zmíněnou strukturu databáze a obsahovala také algoritmy pro trénování HMM popřípadě evaluaci sekvencí gest. Druhá aplikace, Gesture Translator App (dále jen detekující aplikace), by byla určená jen pro zpracování obrazu (detekci rukou a hlavy) pomocí OpenCV. Tím, že vznikla kombinace dvou aplikací řešící překlad znakového jazyka, je potřeba navrhnout, jak dané aplikace budou mezi sebou komunikovat.

Řešení komunikace těchto dvou aplikací vychází z podobného přístupu, jako komunikace i-CT Frameworku s výukovými aplikacemi. I zde se využije URL adres pro vzájemnou komunikaci mezi aplikacemi manageru a detekující aplikací. Je tedy potřeba definovat režimy, kdy aplikace manager vyvolává detekující aplikaci:

- **rozpoznávání** - detekující aplikace se vyvolá s režimem rozpoznávání
- **přehrávání sekvence** - v detekující aplikace přehraje vybranou sekvence gesta
- **přidávání nové sekvence** - jedná se o požadavek z manageru, kdy se v detekující aplikaci vytváří nová sekvence gesta

- **výpočet středů oblastí** - jedná se o případ, kdy aplikace manageru využívá již naimplantované cluster funkce *k-means* z knihovny OpenCV zabudované v detekující aplikaci.

Detekující aplikace je vždy vyvolávána aplikací manageru a po splnění úkolu daného režimu pokaždé nazpět vyvolá aplikaci manageru ve stejném režimu, jako byla vyvolána.

Definování URL schémat aplikací je vypsáno zde:

- **gesturetranslatormanager** - URL schéma pro aplikaci Gesture Translator Manager
- **gesturetranslatorapp** - URL schéma pro aplikaci Gesture Translator App

Obecné definování URL parametrů pro přechod mezi aplikacemi Gesture Translator

V této kapitole bude stanovena definice URL parametrů pro přechody mezi aplikací manageru a detekující aplikací:

- `action` - určuje režim detekující aplikace (*addnewsequence* - pro přidání nové sekvence, *recognize* - pro režim rozpoznání, *kmeans* - pro výpočet středů oblastí obrazovky, *playsequence* - pro přehrání sekvence gesta)
- `filtratedSequence` - určuje jestli se výsledná sekvence filtruje od počátku pozic rukou gesta či nikoliv (nabývá hodnoty *true* či *false*)
- `width` - značí parametr šířky rozlišení kamery
- `height` - značí parametr délky rozlišení kamery
- `lefthandsequence[]` - pole obsahující středy levé ruky, hodnota x a y jsou rozdělené mezerou
- `righthandsequence[]` - pole obsahující středy pravé ruky, hodnota x a y jsou rozdělené mezerou
- `points[]` - pole obsahující středy všech bodů sekvence
- `return` - URL schéma pro navrácení z detekující aplikace do manageru

Sekvence se obyčejně posílá jako parametry `lefthandsequence[]` a `righthandsequence[]`, kde jeden časový okamžik sekvence gesta je znázorněn dvojicí těchto parametrů, tj. střed levé ruky a pravé ruky. Názorný příklad lze vidět v ukázce kódu 4.3, kdy v daném časovém okamžiku sekvence URL adresa obsahuje střed levé ruky se souřadnicemi $x = 458.8$ a $y = 238.0$ a střed pravé ruky se souřadnicemi $x = 179.2$ a $y = 412.4$.

```
...&lefthandsequence[]=458.8 238.0&righthandsequence[]=179.2 412.4&left
    ...
```

Ukázka kódu 4.3: Příklad části URL adresy obsahující jeden časový okamžik sekvence

URL adresy při použití režimu rozpoznávání

V případě překladu gesta, aplikace Gesture Translator Manager vyvolá požadavek na aplikaci Gesture Translator App URL adresou, která je naznačena v ukázce kódu 4.4. Uživatel nechce filtrovat výslednou sekvenci gesta.

```
gesturetranslatorapp://?action=recognize&filtratedSequence=false
```

Ukázka kódu 4.4: URL adresa v režimu rozpoznávání při vyvolání aplikace Gesture Translator App

Po zpracování požadavku aplikace Gesture Translator App vyvolává aplikaci Gesture Translator Manager se získanými daty pomocí URL adresy naznačené v ukázce kódu 4.5.

```
gesturetranslatormanager://?return=gesturetranslatorapp&action=recognize&
    width=720.0&height=480.0&lefthandsequence[]=346.3 371.4&
    righthandsequence[]=347.64 150.5&left....
```

Ukázka kódu 4.5: Příklad URL adresy v režimu rozpoznávání při zpětnému vyvolání aplikace Gesture Translator Manager

Rozlišení kamery při snímání sekvence bylo 720x480 a dále pak první střed levé ruky měl souřadnice $x = 346.3$ a $y = 371.4$ a střed pravé ruky se souřadnicemi $x = 347.64$ a $y = 150.5$. Ostatní středy rukou jsou uloženy dále v URL adrese.

URL adresy při použití režimu přehrávání sekvence

V případě režimu přehrávání sekvence, uživatel v aplikaci Gesture Translator Manager zvolí danou sekvenci, kterou chce přehrát a aplikace je vyvolána Gesture Translator App URL adresou, která lze vidět v ukázce kódu 4.6.

```
gesturetranslatorapp://?action=playsequence&filtratedSequence=false&width
    =1280.0&height=720.0&lefthandsequence[]=362.0 320.0&righthandsequence
    []=220.0 61.0&lefthandsequence[]....
```

Ukázka kódu 4.6: URL adresa v režimu přehrávání sekvence při vyvolání aplikace Gesture Translator App

Po přijetí URL adresy v ukázce kódu 4.6 dekódující aplikace přepočítá všechny získané body z URL adresy vzhledem k poslaným údajům původního rozlišení sekvence (v tomto případě 1280x720) a aktuálního rozlišení kamery. Až po té uživatel může sekvenci přehrát.

Při návratu do aplikace manageru, dekódující aplikace vyvolá manager URL adresou v ukázce kódu 4.7.

```
gesturetranslatormanager://?return=gesturetranslatorapp&action=
    playsequence
```

Ukázka kódu 4.7: Příklad URL adresy v režimu přehrávání sekvence při zpětnému vyvolání aplikace Gesture Translator Manager

URL adresy při přidávání nové sekvence do gesta

V případě režimu vytvoření nové sekvence k danému gestu, aplikace Gesture Translator Manager vyvolá aplikaci Gesture Translator App URL adresou, která lze vidět v ukázce kódu 4.8. Uživatel nechce filtrovat výslednou sekvenci gesta.

```
gesturetranslatorapp://?action=addnewsequence&filtratedSequence=false
```

Ukázka kódu 4.8: URL adresa v režimu přidávání nové sekvence při vyvolání aplikace Gesture Translator App

Po vytvoření nové sekvence aplikace Gesture Translator App vyvolává aplikaci Gesture Translator Manager se získanými daty pomocí URL adresy naznačené v ukázce kódu 4.9.

```
gesturetranslatormanager://?return=gesturetranslatorapp&action=
    addnewsequence&width=720.0&height=480.0&lefthandsequence[]=306.0
    408.0&righthandsequence[]=323.37 142.0&lefthandsequence[]...
```

Ukázka kódu 4.9: Příklad URL adresy v režimu přidávání nové sekvence při zpětnému vyvolání aplikace Gesture Translator Manager

URL adresy při výpočtu středu oblastí obrazovky

Poslední případ, kdy aplikace Gesture Translator Manager vyvolává aplikaci Gesture Translator App, je tehdy, když je potřeba vypočítat středy oblastí obrazovky pomocí knihovny OpenCV, které pak reprezentují abecedu sekvencí v HMM. Před touto operací se v aplikaci Gesture Translator Manager upraví rozlišení bodů všech sekvencí gesta tak, aby byly tyto body ve stejném rozlišení. Po ukončení normalizace rozlišení bodů aplikace manager vyvolává aplikaci Gesture Translator App pomocí URL adresy znázorněné v ukázce kódu 4.10.

```
gesturetranslatorapp://?action=kmeans&clusters=20&points[]=100.68 244.0&
    points[]=...
```

Ukázka kódu 4.10: URL adresa v režimu získávání oblastí obrazovky při vyvolání aplikace Gesture Translator App

Z ukázky kódu si lze povšimnout, že byl přidán ještě jeden URL parametr `clusters`, který určuje kolik má být výsledných středů obrazovky. V URL adrese již nerozlišujeme levé a pravé středy rukou, jelikož tato informace pro clustrovou funkci *kmeans* v knihovně OpenCV nehraje roli.

Po získání všech středů oblastí obrazovky dekódující aplikace vyvolá manager URL adresou v ukázce kódu 4.11, kde URL parametrech `points[]` se posílají postupně získané středy oblastí obrazovky.

```
gesturetranslatormanager://?return=gesturetranslatorapp&action=kmeans&
    points[]=133.2 374.3&points[]=...
```

Ukázka kódu 4.11: Příklad URL adresy v režimu přidávání nové sekvence při zpětnému vyvolání aplikace Gesture Translator Manager

Kapitola 5

Implementace

Kapitola pojednává o implementaci i-CT Frameworku, resp. částech i-CT Frameworku, které naimplemetoval autor této práce, dále pak popisuje implementaci nástavbové aplikace Gesture Translator a nakonec se v kapitole zmíní způsob implementace automatických testů pro rozpoznávací mechanismus.

5.1 Implementace i-CT Frameworku

Na implementaci i-CT Frameworku spolupracovali student Jan Kalina a autor této práce Vojtěch Meca. Každý z těchto studentů vyvíjel určitou část i-CT Frameworku a proto v této kapitole se zaměříme na vyvíjené části autora této práce Vojtěcha Maci.

Vývoj i-CT Frameworku byl rozdělen na dvě hlavní části:

- část vizuální - zde se vyvíjely obrazovky aplikace, lokalizace aplikace, apod.
- část modelová - implementace databáze, komunikace mezi databází a vzhledem aplikace, apod.

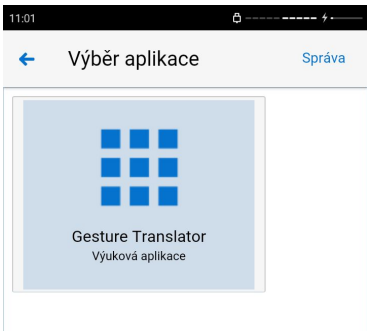
Část vizuální byla implementována autorem této práce a bude v práci popsána a část modelová byla implementavaná Janem Kalinou a je popsána v jeho diplomové práci [22].

5.1.1 Implementace vizuální části i-CT Frameworku

Aplikace i-CT Framework byla vyvíjena v prostředí MAF popsané v kapitole 2.2.4, které nabízí pro vizualizaci speciální Application Mobile XML (dále AMX) soubory mající předdefinované základní komponenty pro vizualizaci (jako např. seznamy, tlačítka, obrázky, atd.). Vzhledem k tomu, že AMX soubory a jejich syntaxi lze snadno pochopit, byly použity v této práci. Navíc v prostředí JDeveloperu, určené pro vývoj MAF aplikací, je možné jednotlivým komponentám v AMX přidávat vlastnosti (např. velikost tlačítka, atd.). Nicméně i po zadání některých těchto vlastností u vybraných komponent se někdy tyto vlastnosti ve výsledné vizualizaci neprojeví.

Obrázek 5.1: Návrh položky seznamu aplikací v i-CT Frameworku

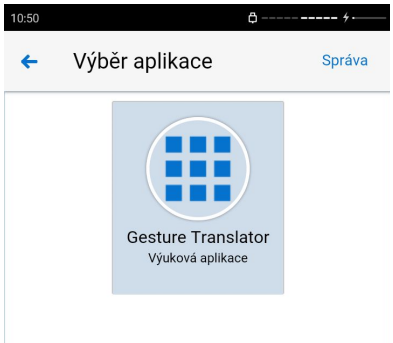
Na obrázku 5.1 lze vidět původní návrh položky ze seznamu aplikací, které jsou zpřístupněny danému uživateli. Výslednou vizualizaci po nastavení všech potřebných vlastností v JDeveloperu pro vzhled této položky seznamu podle zmíněného návrhu lze vidět na obrázku 5.2.



Obrázek 5.2: Položka seznamu aplikací v i-CT Frameworku s využitím JDeveloperu

Vizualizace položky seznamu aplikací, ačkoliv vše potřebné bylo v JDeveloperu zadáno pro zamyšlený výsledný vzhled, není taková, jak by se očekávalo.

JDeveloper pro tyto účely zavádí možnost modifikovat vzhled nadefinovaných komponent v AMX souboru pomocí vlastních CSS stylů. Tento způsob je zaveden vytvořením souboru s nadefinovanými uživatelovými CSS styly. Konkrétní CSS styl je tak možno do AMX souborů přidat tím, že ve výčtu vlastností komponenty je přidána vlastnost `styleClass`, kde se přidá název CSS stylu z přiloženého souboru s CSS styly. Výsledek po zavedení tohoto souboru u položky seznamu aplikací lze spatřit na obrázku 5.3.



Obrázek 5.3: Položka seznamu aplikací v i-CT Frameworku s využitím CSS

Lokalizace aplikace i-CT Frameworku

Vývojáři MAFu pro lokalizaci obecných MAF aplikací vytvořily systém souborů `<BASE_XLIFF_FILE_NAME>_<LANGUAGE_TOKEN>.xlf`, kde `<BASE_XLIFF_FILE_NAME>` je samotný název lokalizačního souboru a `<LANGUAGE_TOKEN>` určuje lokalizaci pro konkrétní světový jazyk podle standardu *ISO-639-lowercase-language-code* [9].

```
</trans-unit>
<trans-unit id="User_name">
  <source>Uživatelské jméno:</source>
  <target/>
</trans-unit>
<trans-unit id="Hint_user_name">
  <source>Uživatelské jméno</source>
  <target/>
```

(a) Screenshot českého lokalizačního souboru

```
</trans-unit>
<trans-unit id="User_name">
  <source>User name:</source>
  <target/>
</trans-unit>
<trans-unit id="Hint_user_name">
  <source>User name</source>
  <target/>
```

(b) Screenshot anglického lokalizačního souboru

Obrázek 5.4: Screenshoty lokalizačních souborů v i-CT Frameworku z prostředí JDeveloper

Na obrázku 5.4 lze vidět samotná realizace nastavení lokalizace pro konkrétní jazyky v i-CT Frameworku. V položce `<trans-unit>` se definuje id, které je pak možné použít v AMX souborech. Při zobrazení takto nadefinovaných AMX souborech se daný text vypíše podle konkrétně nastavené lokalizace zařízení a zobrazí se text z položky `<source>` ze souboru podle daného jazyka.

V této práci byly použity celkem dva tyto lokalizační soubory, jeden pro angličtinu a druhý pro češtinu (`<LANGUAGE_TOKEN>` pro češtinu je cs).

5.2 Implementace aplikace Gesture Translator

V této kapitole budou popsány postupy implementace, které probíhaly v rámci práce nad aplikací Gesture Translator a to zejména implementace části rozpoznávací logiky a také části rozpoznávacího mechanismu.

Samotná implementace aplikace Gesture Translator probíhala v nástroji Oracle MAF, který je popsán v kapitole 2.2.4 podobně jako v případě i-CT Frameworku. Při vývoji nativních operací aplikace se využilo cordova pluginů popsaných v kapitole 2.2.4.

5.2.1 Implementace cordova pluginu s knihovnou OpenCV

Jeden z cílů aplikace Gesture Translator má být rozpoznání rukou a hlavy z obrazu kamery zařízení. Pro práci s videem či zpracování obrazu kamery zařízení je vhodné využít knihovnu OpenCV, která má již předem připravené různé funkce pro rozpoznávání objektu či jeho sledování v obraze (viz kapitola 4.3.2). Bohužel samotná MAF aplikace není schopná používat přímo přístup k nativním knihovnám (OpenCV je nativní knihovna). Proto MAF aplikace využívá cordova pluginů pro zpřístupnění nativních knihoven. V cordova pluginu je tedy třeba zvlášť vytvořit kód pro operační systém Android a zvlášť vytvořit kód pro operační systém iOS.

Při implementaci Gesture Translator bylo zapotřebí, aby aplikace měla nastavenou průhlednost obrazovek, které uživateli jsou předkládány. To z důvodu, aby uživatel mohl vidět získaný obraz z kamery zpracovaný knihovnou OpenCV, který by běžel na pozadí aplikace, a tedy mohl vidět výsledné sekvence gesta a zároveň měl přístup k obrazovkám aplikace.

Tato vlastnost se dá nastavit v aplikaci při instalování cordova pluginu do této aplikace. Bohužel i při bezproblémové instalaci cordova pluginu s OpenCV (sledování výpisů instalování vlastností do aplikace ve třídě *SystemWebViewEngine* projektu Cordova Android) se daný efekt přidané vlastnosti neprojevil s využitím Oracle MAF v prostředí Androidu. Správnost fungování vytvořeného cordova pluginu v prostředí Androidu byla otestována v aplikaci Apache Cordova Framework, kde bylo možno vidět obraz získaný z kamery a následně zpracovaný knihovnou OpenCV.

Proto byla tato situace nakonec vyřešena novým návrhem systému Gesture Translator, který obsahuje aplikace Gesture Translator Manager a aplikace Gesture Translator App (viz kapitola 4.3.4).

5.2.2 Implementace aplikace Gesture Translator Manager

Jedná se o aplikaci, ve které bude implementovaná databáze a dále i rozpoznávací logika překladače gest. Pro vývoj aplikace byl zvolen Oracle MAF a to z toho důvodu, že funkcionalita aplikace již nevyžadovala žádný přístup k nativním knihovnám (nebylo třeba vyvíjet vlastní cordova plugin) a také i pomocí tohoto nástroje bude aplikace multiplatformní.

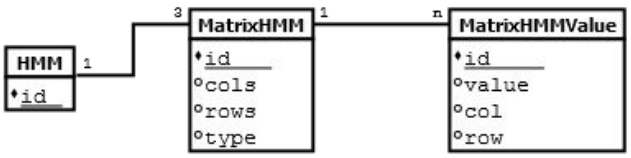
Implementace databáze v aplikaci Gesture Translator Manager

V aplikaci Gesture Translator Manager se využilo databáze SQLite. Tato databáze je totiž využívaná jak na platformě Android, tak i na platformě iOS a tedy neomezí možnost implementovat aplikaci multiplatformně. Jak bylo již zmíněno v kapitole 2.2.4, MAF obsahuje JBDC, díky kterému je možné přímo přistupovat k SQLite databázi. Bohužel JDBC zabudované v MAF aplikacích neumožňuje objektově-relační mapování (ORM).

Pro zlehčení práce s databází bylo využito knihovny OrlLite, která ORM mapování zpřístupňuje v programovacím jazyce Java přes JDBC. Není tedy třeba psát příkazy pro přístup k databázi ručně, např. když se ukládá objekt do databáze nebo se z databáze maže.

Další značná výhoda knihovny OrlLite je, že zabírá necelých 400 kB fyzické paměti. Toto ale přináší i pár nevýhod. Mezi ty nejhlavnější patří to, že s OrmLite není možné plnohodnotně využívat dědičnosti mezi objekty domény.

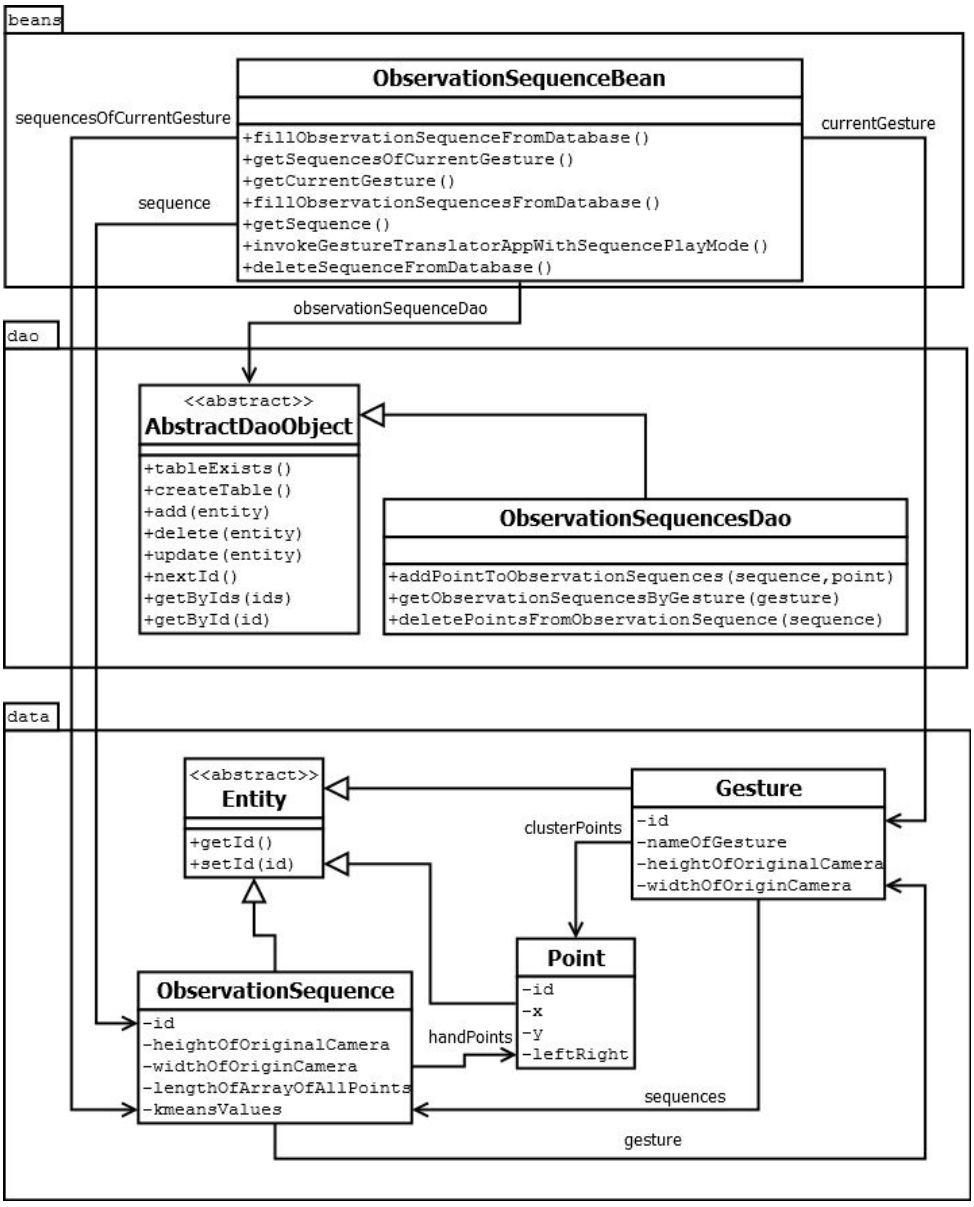
Díky tomu, že OrmLite mapuje objekty na tabulky relační databáze, tak konkrétní struktura databáze aplikace je navržená pomocí diagramu tříd v kapitole 4.3.3. Nicméně u entity HMM by se do databáze měly ukládat celé matice, což v relační databázi SQLite není možné. Proto byla struktura databáze rozšířena o další dvě tabulky MatrixHMM a MatrixHMMValue. Na obrázku 5.5 lze vidět ER model tohoto rozšíření.



Obrázek 5.5: ER model rozšíření struktury databáze o možnosti ukládání matic do databáze

Implementační diagram tříd vyznačený na obrázku 5.6 částečně znázorňuje architekturu aplikace Gesture Translator Manager. Celková architektura aplikace je rozšířena ještě o jeden balíček s názvem *function*, který obsahuje třídu pro čtení souborů *FilesIO* a třídu *GeneralFunctions* pro obecnou práci s vzájemnou pozicí dvou bodů (např. metoda *get-*

DistanceBetweenTwoPoints vracející vzdálenost mezi dvěma body). Architektura aplikace Gesture Translator Manager vychází z architektury aplikace i-CT Framework.



Obrázek 5.6: ER model rozšíření struktury databáze o možnosti ukládání matic do databáze

Také v diagramu 5.6 lze vidět rozdělení tříd do balíčků a vazby mezi jednotlivými třídami. Dále zobrazuje všechny abstraktní třídy aplikace a třídy související se záznamy sekvencí přiřazené k danému gestu. Podobně se tak postupuje i s ostatními typy databázových objektů.

Všechny třídy, které představují data databáze, dědí z nově zavedené abstraktní třídy *Entita* pro snadnější manipulaci s databází. Dále se tento typ třídy používá pro implementaci objektů, které přistupují k tabulkám databáze (Data access object - DAO).

Navíc je v aplikaci umístěna sada souborů se sekvencemi několika znaků českého znakového jazyka. Uživatel tedy po nainstalování aplikace může nahrát tyto sekvence do databáze a po natrénování jednotlivých znaků (gest) může tyto sekvence překládat (pomocí druhé aplikace Gesture Transaltor App).

Implementace rozpoznávací logiky v Gesture Tranlator Manager

Jak již bylo zmíněno výše, aplikace Gesture Translator Manager obsahuje také rozpoznávací logiku překladače znakového jazyka. Metoda `public void trainAction(ActionEvent actionEvent)` implementovaná ve třídě *HMMBean* odchytává událost trénování HMM pro dané gesto. Samotný Baum-Welsch algoritmus (viz kapitola 2.2.7 v sekci „Úvod do Skrytých Markovských Modelů“) je však naimplementovaný v metodě *trainHMM* (jeho optimalizace pomocí scaling se nachází v metodě *trainHMMWithScaling*) s parametrem pole obsahující všechny sekvence, ze kterých se dané HMM trénuje. Tato metoda se nachází také ve třídě *HMMBean*.

Rozpoznávací mechanismus navržený v algoritmu popsaném v kapitole 4.3.1 se pak nachází v metodě *setTextOfRecognizeSequence* se vstupním parametrem sekvence hledaného gesta ve třídě *ModelBean*. Totiž v této třídě se nachází aktuální seznam všech gest uživatele, ze kterých se pak vybírá gesto. Metoda *setTextOfRecognizeSequence* pracuje s třídou *HMMBean*, kde v metodě *getProbabilityFromAlphaWithScaling* je realizován *Forward-Backward* algoritmus optimalizovaný pomocí scaling (viz kapitola 2.2.7 v sekci „Úvod do Skrytých Markovských Modelů“). Metoda *setTextOfRecognizeSequence* se pak volá metodou *decodingOfObservationSequence*, která vrací pravděpodobnost generování sekvence daným HMM gesta. Pro průchodu všech gest a získání pravděpodobností z HMM k dané vstupní sekvenci se vybere gesto s největší pravděpodobností a vypíše se název gesta na obrazovku aplikace. Pokud je ale sekvence neznámá u všech gest (tedy pravděpodobnost vstupní sekvence je rovna 0 u všech HMM), tak se vypíše chybové hlášení, že nebylo gesto nalezeno.

5.2.3 Implementace aplikace Gesture Translator App

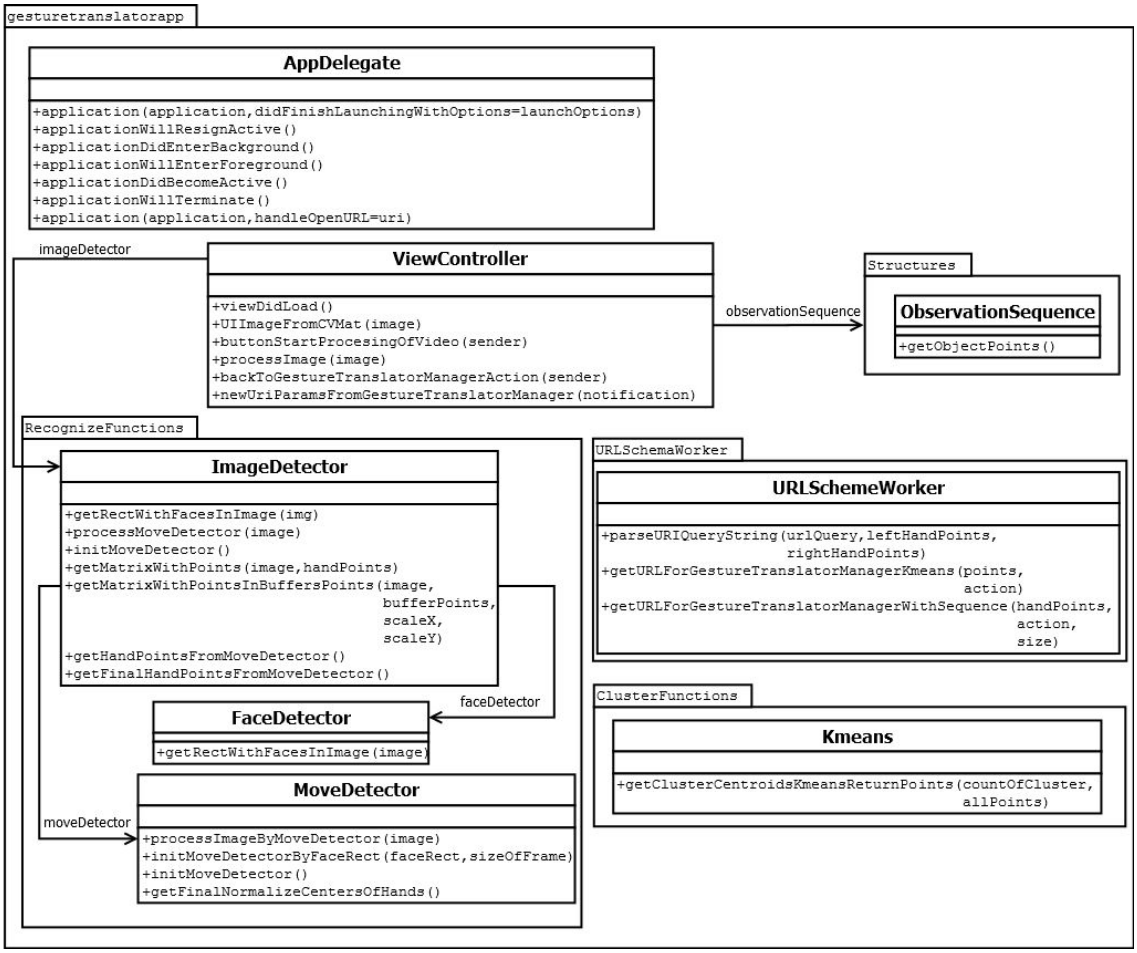
Cílem této aplikace je simulovat funkcionalitu, která se předpokládala vytvořením cordova pluginu, který by zpracovával obraz kamery s knihovnou OpenCV. Podobně jako by to bylo i v cordova pluginu, je třeba implementovat jednu aplikaci pro prostředí operačního systému Android a další aplikaci pro prostředí iOS.

Pro aplikaci určenou na platformu Android se využilo programovacího jazyku Java a vývojového prostředí *Android Studio 2.3.1*, které je doporučeno pro vývoj androidních aplikací na oficiálním webu Android Developers [1].

Aplikace implementovaná pro platformu iOS byla realizovaná v prostředí *Xcode 8* [2], které je vhodné pro vývoj aplikací tohoto typu. Pro vývoj této aplikace byl také využit programovací jazyk Objective-C, který podporuje přímý přístup k metodám a třídám knihovny OpenCV a je k dispozici v prostředí *Xcode 8*.

Architektura aplikace Gesture Translator App

Architektura aplikací, jak pro oprerační systém Android tak i pro iOS, je stejná a lze vidět na obrázku 5.7 s implementačním digramem aplikace Gesture Translator App pro iOS. Aplikace pro Android má stejné třídy a metody jako v iOS jen s tím rozdílem, že místo tříd *AppDelegate* a *ViewController*, které řídí chod aplikace v iOS, je zavedena třída *MainActivity*, která obsluhuje řízení celé androidní aplikace.



Obrázek 5.7: Implementační diagram aplikace Gesture Translator App pro iOS (pro zjednodušení vynechány soukromé metody tříd)

Pro výpočet středů oblastí obrazovek se vybrala funkce *kmeans* naimplementovaná v knihovně OpenCV [4]. Této funkce se volá v metodě *getClusterCentroidsKmeansReturnPoints* třídy *Kmeans*.

Zvláštní pozornost se pak soustřeďuje na objekt třídy *ImageDetektor*. Ta vlastní objekty třídy *FaceDetector* a *MoveDetektor* a implementuje celkový algoritmus pro rozpoznávání rukou a hlavy v obrazu kamery (viz kapitola 5.2.3 v sekci „Implementace rozpoznávacího mechanismu“).

V aplikaci Gesture Translator App se pak také využívá objektu třídy *URLSchemeWorker*, díky kterému se zpracovávají parametry s přijaté URL adresy od aplikace Gesture Translator Managera a dále díky, kterému se vytvářejí požadované URL adresy pro vyvolání aplikace Gesture Translator Manager (viz kapitola 4.3.4).

V neposlední řadě se používá objekt třídy *ObsevationSequence* pro práci s nalezenými body středy rukou.

Implementace rozpoznávacího mechanismu

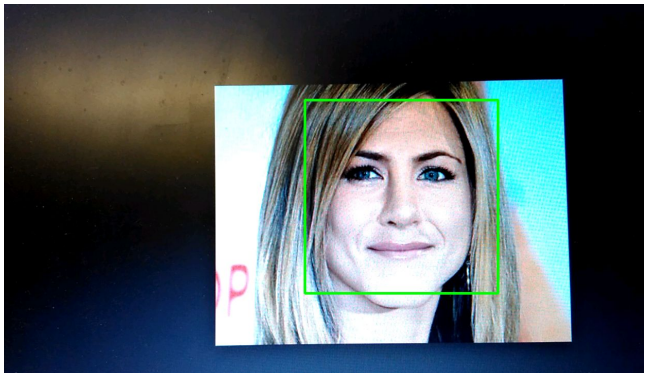
Celkový algoritmus se vytvářel postupnými iteracemi vývoje a v této kapitole se stručně popíše již výsledný algoritmus, který se implementoval v obou aplikacích Gesture Translator

App. V průběhu iterací se konkrétní kroky konzultovali s Ing. Kamilem Behúňem z UPGM VUT FIT.

Algoritmus rozpoznávání rukou a hlavy je založen pohybu rukou. Totiž při provádění každého ze znaku znakového jazyka je potřeba s rukama pohybovat. Jedna z vhodných metod pro řešení tohoto problému je použít objekt třídy **BackgroundSubtractorMOG2** z knihovny OpenCV. Tento objekt pak dokáže porovnat dva obrázky a efektivně zjistit vzájemné změny na úrovni pixelů těchto obrázků. Při použití metody *apply* tohoto objektu se získává maska, kde bílé body značí nalezené změny v rámci dvou obrázků a černé body znamenají, že pixely na obou obrázcích jsou stejné. Objekt této třídy se využije v algoritmu rozpoznávání ve třídě *MoveDetector*.

Pro rozpoznávání obličeje se zdá jako vhodné použití objekt třídy **CascadeClassifier** z knihovny OpenCV. Objekt třídy **CascadeClassifier** lze natrénovat ze souboru **lbpcascade_frontalface.xml** pro rozpoznávání obličeje v předloženém obraze. Je tedy třeba tento soubor umístit do obou aplikací Gesture Translator App. Metoda *detectMultiScale* tohoto objektu při vstupním parametru obrázku kamery vrátí obdélníky, které značí oblasti nalezených všech obličejů obrázku (viz obrázek 5.8). Objekt této třídy bude implementován ve třídě *FaceDetector*.

Co se týče detektoru barvy kůže často vznikal problém, kdy byly vyhodnoceny tímto detektorem oblasti, které neobsahovaly žádné části lidského těla, a oblasti lidského těla (jako obličej, odhalené ruce, apod.) nebyly rozpoznány a proto se nakonec z tohoto přístupu opustilo. Nemalý vliv byl dán také konkrétními světelnými podmínkami, což se také potvridlo i při automatických testech ze vzorku videí, kde byl tento detektor aplikován, více v kapitole 6.3.1. Použitý detektor barvy kůže byl implementován řešením popsaným v dokumentu [14].



Obrázek 5.8: Výsledek obrázku při použití objektu třídy **CascadeClassifier** z knihovny OpenCV

Nastínění algoritmu bude popsáno v následujících bodech:

1. Získej obrázek z kamery. Pokud je již nalezena hlava pokračuj bodem 4.
2. Získej z obrázku pozici hlavy znakující osoby pomocí objektu třídy **CascadeClassifier** z knihovny OpenCV. Pokud jsi získal pozici hlavy pokračuj bodem 3, jinak znovu proveď bod 1.
3. Nastav regiony, kde se můžou pohybovat ruce podle pozice hlavy, tj. region pro levou a pravou ruku zvlášť.

4. Pokud již máš uložený obrázek z minulého cyklu pokračuj bodem 6. Jinak proveď bod 5.

5. Získej první obrázky regionů levé a pravé ruky a zaznamenej si je. Pokračuj bodem 1.

6. Získej další obrázky regionů levé a pravé ruky.

7. Porovnej obrázky mezi sebou pro každý region ruky zvlášť a získej tak masky pohyblivých objektů v regionech (pomocí **BackgroundSubtractorMOG2** z knihovny OpenCV).

8. Největší bílou oblast masky pro region levé ruky označ jako pozice levé ruky a podobně zpracuj i masku oblasti pravé ruky.

9. Vypočítej střed pozice pro levou i pravou ruku a tyto body si poznamenej.

10. Druhé obrázky ulož na místo prvních obrázků regionů rukou a pokračuj bodem 1.

Algoritmus končí v případě, kdy uživatel držící zařízení vypne snímání znakující osoby, která již daný znak provedla. Po ukončení algoritmu jsou uloženy záznamy všech středů rukou (kromě prvních pozic, jelikož první obrázky regionů rukou nebyly porovnány s jinými obrázky) od získání pozice hlavy znakující osoby.

Bod číslo 8 v nastíněném algoritmu je navíc rozšířen o vyřešení případu, kdy jedna ruka opouští svůj region a přesunuje se do regionu druhé ruky. Totiž v případě pohybu jedné ruky do regionu druhé ruky se v daném regionu, odkud se ruka pohybuje, nenalezne žádný pohybující objekt a detekuje se pohyb jedné ruky do regionu druhé ruky. V takovém případě se v druhém regionu ruky přidělí druhé ruce nejbližší nalezená oblast od poslední její pozice a ruce, která se do regionu dostala, se přidělí zbylá oblast pohybujících se objektu v daném regionu. V dalších cyklech se v regionu, kde se nacházejí obě ruce, vždy vybírají nejbližší oblasti rukou v porovnání s oblastmi rukou z minulého obrázku. V případě, že se ruka vrátí zpět do svého regionu, algoritmus pokračuje klasickým scénářem popsaným výše.

Poslední rozšíření nastíněného algoritmu spočívá v normalizaci nalezených středů rukou podle vzájemné polohy bodu se souřadnicemi x = 0 a y = „střed šířky obrazovky“ a středu pozice hlavy. Tím se zamezí to, že by se nerozpoznalo stejné gesto prováděné v různých částech obrazovky.

Objekt třídy *MoveDetector* zpracovává body 4 až 10 v nastíněném algoritmu v metodě *processImageByMoveDetector*. Objekt třídy *ImageDetector* řídí celý algoritmus v metodě *processMoveDetector*.

Očekávaná funkčnost algoritmu byla zajištěna testování úspěšnosti algoritmu pomocí automatických testů (viz kapitola 6.3).

Filtrování získaných středů rukou

Výše popsaný algoritmus je založen na pohybu rukou. Byla tedy vložena nová funkčnost pro filtraci získaných středů rukou pro uživatele, který zařízení používá. Z praktického hlediska znakující osoba nutně musí pohnout rukama tak, aby se začala zaznamenávat poloha jejich rukou. Tedy až po prvním pohybu rukou může začít znakující osoba provádět gesto. Bohužel tímto se ale zanáší do výsledné sekvence středů rukou počáteční pozice rukou, které s daným gestem nesouvisí. Proto je možné tyto body filtrovat. Totiž ve výsledné sekvenci středu rukou se hledá místo, kde se ruce nepohybovaly - neměnily svou pozici, tj. začátek pozic rukou od startu provádění gesta. Všechny body ve výsledné sekvenci do tohoto místa

se odstraní a tím vznikne sekvence pouze snímající celé gesto, nikoliv něco navíc. Tato funkcionalita se nachází v metodě *getFinalNormalizeCentersOfHands* ve třídě *MoveDetector*.

5.3 Implementace automatických testů rozpoznávacího mechanismu

V této podkapitole bude nastíněna implementace použitých skriptů pro automatické testy. Samotný scénář automatických testů bude popsán v podkapitole 6.3.2.

Automatické testy skládají ze dvou Bash skriptu `runParseVideosByAnnotation.sh`, `runTestOfRecognizeProcess.sh` a programu `TestOfHMM.jar` programovacího jazyka Java.

Bash skript `runParseVideosByAnnotation.sh` se používá k automatickému rozdělení videí zaznamenávající figuranty znakující různé gesta na více menších videí, které obsahují právě jen jedno gesto znakujícího. Skript volá pro každé video program programovacího jazyka Python `createVideosByAnotation.py`, který má na vstupu právě cestu k danému videu. Zde je důležité zmínit, že formát názvu a cesty k video souboru musí mít stejnou strukturu jako formát názvu a cesty xml souboru obsahující anotace k výsledným menším videím. Anotace jednoho menšího videa obsahuje čas původního videa, kdy se dané gesto začíná provádět, a čas, kdy provádění gesta končí. Také je zde i informace o názvu anotace. V ukázce kódu 5.1 je naznačena struktura anotačního xml souboru.

```
<?xml version="1.0" encoding="UTF-8"?>
<TIER xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
    noNamespaceSchemaLocation="file:avatech-tier.xsd" columns="all_tiers">
  <span start="1.420" end="2.210">
    <v>nameOfFirstActor</v>
  </span>
  <span start="4.310" end="5.140">
    <v>nameOfSecondActor</v>
  </span>
  ...
</TIER>
```

Ukázka kódu 5.1: Struktura anotačního xml souboru

Program `createVideosByAnotation.py` využívá zmíněnou knihovnu Open CV, pro práci s videem a také knihovnu „The elementtree library“ pro práci s xml soubory. V celém programu je pouze jeden cyklus `while`, který prochází jednotlivé obrázky videa a ukládá tyto obrázky do dílčích videí podle času v anotacích.

Všechny testovací videa byla natáčena jedním zařízením (popsáno v kapitole 6.3.2). Ačkoliv kameraman držel zařízení tím způsobem, aby zástupný figurant (dále jen figurant) byl ve videu ve vzpřímené poloze, tak po stažení natočeného videa do PC byl figurant otočen o 270°. Proto navíc program `createVideosByAnotation.py` obrázky otáčí o těchto 270°, aby figuranti v těchto malých videích byly v původní vzpřímené poloze.

Bash skript `runTestOfRecognizeProcess.sh` má na starosti automaticky procházet menší videa získané skriptem `runParseVideosByAnnotation.sh`, zpracovat je a uložit pozice rukou do souborů s příponou csv. K tomu využívá program `basedOnMoveDetector.cpp` programovacího jazyka C++ obsahující rozpoznávací mechanismus popsaný v kapitole 5.2.3 v sekci „Implementace rozpoznávacího mechanismu“. Tedy vstupem do programu je cesta

k videu, dále adresář, kde se uloží výsledný csv soubor, a poslední parametr je určen pro zobrazení testovacích výpisů (1) či jejich ignorace (0).

Podobně jako minulý program i tento využívá knihovnu Open CV pro práci s videem. Navíc zde využívá **CascadeClassifier**, což je objekt pro detekci obličeje naimplementovaný v Open CV. Tento objekt načítá trénovací množinu obličejů ze souboru **lbpcascade_frontalface.xml** (tedy v adresáři, kde se spouští program s rozpoznávacím mechanismem musí být vložen i tento soubor). Dále je v programu hlavní cyklus `while`, který prochází jednotlivé obrázky vstupního videa, porovnává je mezi sebou a tím získává pohyb rukou a tedy pozici samotných rukou. Výsledné pozice normalizuje podle pozice hlavy a ukládá do stejnojmenného souboru s příponou csv, jako název vstupního videa. V ukázce kódu 5.2 lze vidět hlavičku csv souboru. Ukládají se pouze pozice levé a pravé ruky.

```
left, ,right
x ,y  ,x  ,y
...
```

Ukázka kódu 5.2: Hlavička csv souboru obsahujícího nalezené body pozic rukou

Jedná se o stejný princip, jako je v rozpoznávacím mechanismu aplikace Gesture Translator App uložený ve třídě *MoveDetector*. Nejprve byl vyvíjen program uložený v souboru `basedOnMoveDetector.cpp` a pak se provedl přepis rozpoznávací logiky do tříd *MoveDetector* aplikací Gesture Transalator App. Tento postup byl použit z důvodu objektivního testování rozpoznávacího algoritmu.

Posledním dílem automatických testů je program Java `TestOfHMM.jar`. Tento program se používá pro simulaci trénování HMM, použité ve výsledné aplikaci Gesture Translator Manager. Jediný rozdíl oproti aplikaci spočívá v zavedení třídy **TestOfHMM**, která obsahuje metodu *main*. V metodě *main* se volá metoda *fullfillModelByDirectoryWithGesturesSequences*, díky které se vytvoří model s gesty. Tato gesta obsahují informace získané o pozicích rukou z csv souborů reprezentující sekvence na trénování a testování daného gesta. V této metodě také trénujeme jednotlivé HMM pro daná gesta pouze trénovací množinou sekvencí. Na závěr celé metody *main* se volá metoda *testOfModel*, která spouští samotný test úspěšnosti testovacích sekvencí patřící k daným gestům. Vypisují se jednotlivé statistiky pro daná gesta, ale také statistika celkové úspěšnosti rozpoznání.

Daný `TestOfHMM.jar` se vytváří ze zdrojových souborů a sestavujícího souboru **build.xml**. Sestavující soubor **build.xml** je spouštěn pomocí Apache Ant[1]. V ukázce kódu 5.3 lze vidět, jakým příkazem se sestavuje `TestOfHMM.jar`.

Nakonec je třeba uvést, že dané automatické testy fungují v případě, když se zachová struktura adresářů, kde budou dané skripty, videa či soubory csv uloženy. Tato adresářová struktura je znázorněna a popsána v kapitole 6.3.

```
ant -DocvJarDir=path/to/dir/containing/opencv-244.jar -DocvLibDir=path/to
    /dir/containing/opencv_java244/native/library -DcsvDir=path/to/dir/
    containing/gestureDirs -DnumOfClusters=numberOfCluster -DnumOfStates=
    numberOfStatesInHMM
```

Ukázka kódu 5.3: Příkaz pro sestavení balíčku TestOfHMM.jar

[1] Jedná se o jeden z mnoha způsobů, jak sestavit java program s příponou *.jar*. Dostupné z: http://ant.apache.org/

Kapitola 6

Testování

Cílem kapitoly bude zhodnocení úspěšnosti návrhu a implementace obou aplikací. K tomuto budou sloužit dva různé typy testů. První z nich je zaměřen na testování aplikací v reálném prostředí, bude se testovat metrika **použitelnosti** aplikace, a druhý test hodnotí, jak jsou dané algoritmy použité v aplikacích **úspěšné**, tj. jak byly tyto algoritmy spolehlivé při konkrétních podmínkách zástupného prostředí.

V první části kapitoly se nastíní celkové zhodnocení aplikace i-CT Frameworku. V dalších částech kapitoly bude postupně rozepsáno, jak se testovaly rozpoznávací mechanismus a rozpoznávací logika HMM aplikace GestureTranslator, a na závěr se uvede celkové zhodnocení samotné aplikace Gesture Translator.

6.1 Zhodnocení aplikace i-CT Frameworku

Při testování i-CT Frameworku se využil první ze zmíněných typu testů, tj. testovala se metrika **použitelnosti** aplikace v reálném prostředí. K tomuto bylo využito prostředí Speciální základní školy Poděbrady. Pedagogové této školy mají již dlouholetou zkušenost s využíváním tabletů při výuce lidí s mentálním postižením. Konkrétně se tablety objevily v této škole už v roce 2011, kde krátce po uvedení tabletů v praxi, vznikla komunita **iSEN**, zaměřující se na vývoj aplikací pro osoby s mentálním postižením.

Tato škola, popřípadě komunita **iSEN**, je v oblasti výuky mentálně postižených známá ve světě. Nasvědčuje tomu pozvání komunity **iSEN** na mezinárodní událost **Apple Special Needs Summit, Prague** roku 2013, kdy se mimo zástupce firmy Apple této události zúčastnily také skupiny z celkem 17 zemí světa, kde 12 z těchto skupin byla ministerstva školství [3].

Testování použitelnosti aplikace i-CT Frameworku bylo provedeno minulý rok v období od 11.5. do 20.5.2016, kdy se aplikace odevzdávala v rámci diplomové práce Jana Kaliny. V této době autor této práce také dokončil práci na aplikace i-CT Frameworku a dále se zaměřoval na druhou aplikaci Gesture Translator, popsanou také v této práci. Testování aplikace i-CT Frameworku se zúčastnilo celkem 6 speciálních pedagogů.

6.1.1 Stanovení testů použitelnosti pro i-CT Framework

Pojem metriku **použitelnosti** lze definovat, jako míru jednoduchosti použití uživatelského rozhraní. Pan Jakob Nielsen, Ph.D., stanovil několik hledisek, jak se dívat na tuto metriku **použitelnosti**. Skládá se tedy z pohledů, jak snadno je pro uživatele činit základní úlohy při prvním setkáním s aplikací (**Naučitelnost**), jak rychle se uživatel zorientuje

v provádění jednoduchých úloh při obeznámení se s rozhraním (**Efektivita**), kolik chyb uživatelé dělají při použití rozhraní a jak snadno se z nich zotavují (**Chybovost**), jak moc je nenáročné provádět základní úlohy, když se uživatel k rozhraní dostane po delší době (**Zapamatovatelnost**), a jak je příjemné pro uživatele rozhraní používat (**Uspokojení**). [21]

Konkretní podoba tohoto testování byla navržena vedoucím práce pro její potřebu a s ohledem na užitou cílovou skupinu. Oslovení pedagogové měli za úkol vyplnit dotazník, kde ke každému z těchto hledisek měli vypsat hodnotu od 0 (nejhorší) až 5 (nejlepší) jen s výjimkou aspektu **Zapamatovatelnosti**, kde nejlepší hodnota bylo číslo 4 místo 5. Z těchto získaných hodnot se pak pomocí váženého průměru vypočítala celková metrika **použitelnosti**.

Aspekt	**Přidělená váha**
Naučitelnost	0,15
Efektivita	0,3
Chybovost	0,3
Zapamatovatelnost	0,15
Uspokojení	0,1

Tabulka 6.1: Přehled vah jednotlivých aspektů metriky použitelnosti

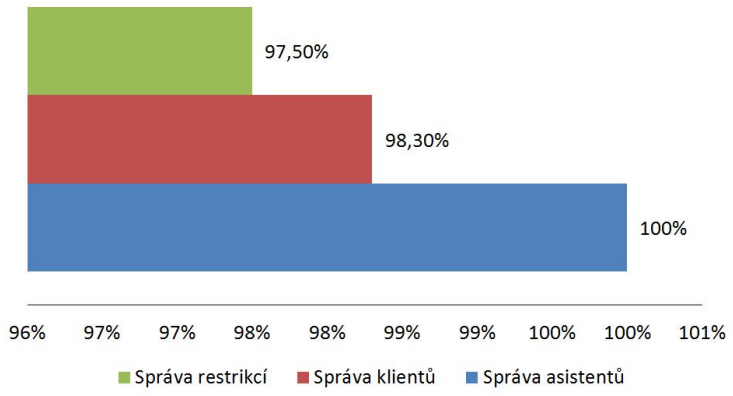
Jednotlivé váhy k daným aspektům **použitelnosti** lze přehledně vidět v tabulce 6.1. Ze všech nejmenší váhu má hledisko Uspokojení, což je zapřičiněno tím, že záleží na aktuálním naladění uživatele. Na druhou stranu největší hodnotu má hledisko Efektivita, která měří rychlost seznámení se s i-CT Frameworkem a jeho následného použití. Stejnou váhu lze zpozorovat i u hlediska Chybovosti, které značí, jak moc těžké je pro začínajícího uživatele vracet se z míst aplikace, kde uživatel zabloudil. Nakonec na pomezí vah jsou umístěny aspekty Naučitelnosti a Zapamatovatelnosti, kde se měří, jak rychle se nový uživatel s aplikací seznámil a jak rychle se uživatel zorientuje, když se k aplikaci dostane po „delší době “. (S ohledem na časový interval, kdy se i-CT Framework testoval, byl za „delší dobu“ pokládán následující pracovní den.)

6.1.2 Výsledky testů aplikace i-CT Frameworku

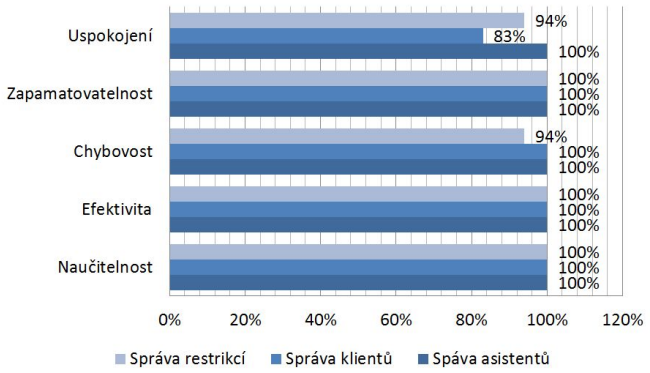
Testování aplikace i-CT Frameworku probíhalo pouze na platformě iOS, která se nejčastěji používá pro aplikace určené pro výuku osob s mentálním postižením. Dále se u aplikace i-CT Frameworku hodnotila metrika **použitelnosti** těchto jejích částí:

- Správa klientů - práce s účty klientů a jejich profilů (tj. správa záznamů zdravotního, sociálního a vzdělávacího profilu)
- Správa asistentů - editace vlastního účtu asistenta, popřípadě editace účtů podřízených asistentů
- Správa restrikcí - přidělování jen několika aplikací k daným klientům a nastavování doby spuštění aplikace u konkrétních klientů

Výsledné hodnocení použitelnosti v procentech lze spatřit v grafu na obrázku 6.1. Hodnocení jednotlivých aspektů použitelnosti dílčích částí i-CT Frameworku lze zpozorovat na obrázku 6.2.



Obrázek 6.1: Výsledné hodnocení použitelnosti i-CT Frameworku v procentech



Obrázek 6.2: Detail výsledného hodnocení použitelnosti i-CT Frameworku v procentech

Výsledné grafy ukazují, že celkové hodnocení i-CT Frameworku podle metriky **použitelnosti** je velmi dobré. Je tedy nutno podotknout, že využití návrhových principů Počítačové terapie má kladné důsledky na vývoj takto specializovaných aplikací.

6.2 Testování rozpoznávací logiky HMM aplikace Gesture Translator Manager

V této práci se pro rozpoznávací logiku v podobě HMM nevyužila žádná ze stávajících knihoven řešící podobný problém, ale vlastní implementace zmíněné v kapitole 5.2.2 v sekci „Implementace rozpoznávací logiky v Gesture Tranlator Manager“. Testování správnosti vlastní implementace probíhala porovnáním několika sad sekvencí při trénování HMM s výstupy z knihovny *Jahmm*[1] vytvořené autorem Jean-Marcem Francois v programovacím jazyce Java a z knihovny *Hidden Markov Model (HMM) Toolbox*[2] autora Kevina Murphyho v programovacím jazyku Matlab. Pro HMM bylo nastaveno 7 skrytých stavů, 8 symbolů abecedy generovaného řetězce a pro maximum iterací Baum-Welsch algoritmu bylo nastaveno 50. Výsledky všech zmíněných HMM byly stejné, tedy v porovnání všech hodnot matic A, B, π. Rozdíly začaly vznikat až na hodnotách okolo v řádu 10 na -100, což je přijatelné.

[1] Dostupné zdrojové kódy z: `https://github.com/KommuSoft/jahmm`.
[2] Dostupné z: `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`.

6.3 Testování rozpoznávacího mechanismu aplikace GestureTranslator

Cílem tohoto testování bylo zjistit, jak se chová použitý rozpoznávací proces v odlišných prostředích a při znakování různých gest. Díky tohoto testování se mohl postupně rozpoznávací mechanismus vylepšovat až na stávající verzi.

Způsob testování tohoto rozpoznávacího mechanismu byl proveden pomocí automatických testů. Tento přístup přináší výhodu, že při použití více videí lze vidět, zda-li rozpoznávací mechanismus funguje i pro obecnější použití a ne jen pro jeden omezený konkrétní případ (např. jedno jediné prostředí, apod.).

Po konzultaci s vedoucím práce pro testování bylo vybráno těchto deset gest: radostný, prohlížeč, kytara, dům, míč, talíř, velký, malý, život a strom. Ukázky jednotlivých znaků je možno vidět v online slovníku znakové řeči na portálu `http://ruce.cz/slovnik`.

Vybraná gesta reprezentují základní slova v užívání běžného života a dále se jedná o gesta českého znakového jazyka. Při znakování těchto gest se vždy využívají obě ruce. Výběr gesta strom bylo užito z důvodu, aby se otestovala správnost provádění gesta, u kterého dochází k přecházení jedné ruky do výskytu regionu druhé ruky.

Stanovení podmínek testovacích videí

Pro nahrávání videí bylo předem stanoveno několik podmínek, aby testy mohly být co nejvíce objektivní, ale také, aby nedocházelo k velkým odlišnostem prováděných gest. Mezi tyto podmínky patří:

- Znakující figuranti by měli být jak muži, tak ženy.
- Figurant zná, jak se gesto provádí.
- Figurant snímaný kamerou by měl stát uprostřed snímku.
- Pozadí za figurantem by mělo být jednobarevné, např. bílá stěna.
- Figurant by měl stát 80 cm od stěny a 180 cm od kamery.
- Scéna by se měla natáčet v takovém prostředí, kde by se neměly vyskytovat ostré stíny na těle ani na pozadí stěny.

Stanovení prostředí pro automatické testy

Pro automatické testy je potřeba vytvořit danou adresářovou strukturu, aby se testy mohly správně spouštět. V ukázce kódu 6.1 lze vidět jak tato struktura vypadá. V adresáři **ROOT_DIR** se nachází výše zmíněné Bash skripty `runTestOfRecognizeProcess.sh` a `runParseVideosByAnnotation.sh`, ale také zkompilovaný program `basedOnMoveDetector.cpp`, program `createVideosByAnotation.py` a soubor **lbpcascade_frontalface.xml** s trénovacími množinami pro rozpoznávání obličejů. Dále je v adresáři **ROOT_DIR** adresář **HMM**, kde se nachází soubor **build.xml** pro program Apache Ant a zdrojové kódy HMM logiky jsou uloženy v adresáři **HMM/src**. Popis k jednotlivým skriptům či programům je v kapitole 5.3.

```
ROOT_DIR
    HMM
        src
    gesture_1
        videa
        csv
    gesture_2
        videa
        csv
...
```

Ukázka kódu 6.1: Struktura adresářů automatických testů

Adresáře **gesture_1** až **gesture_n** se nacházejí v adresáři **ROOT_DIR** a vždy platí pro ně tyto zásady:

- Adresář **gesture_x** obsahuje soubor s videm ve formátu **gesture_x_E[_V].mp4**, kde **gesture_x** je název gesta (např. strom,...), **E** je prostředí, kde se video točilo (0 pro první prostředí, 1 pro druhé prostředí, atd.) a **V** je volitelný parametr, který určuje verzi videa pro konkrétní znak v konkrétním prostředí. Také v tomto adresáři je nutno mít soubor **gesture_x_E[_V].xml** s anotacemi k danému videu.
- V adresáři **gesture_x/videa** se nacházejí videa, které obsahují provedení jen jednoho gesta. Formát názvu těchto videí je **gest_ID_il_E_per_NAME_rep_R.avi**, kde **ID** je číslo označující identifikátor gesta, **E** je prostředí, kde se video točilo (viz výše), **NAME** je jméno herce získané z názvu anotace a **R** číslo opakování gesta daného herce.
- V adresáři **gesture_x/csv** se nacházejí csv soubory, ve kterých jsou uloženy pozice ruk k jednotlivým videím z adresáře **gesture_x/videa**. Formát názvu csv souboru je **gest_ID_il_E_per_NAME_rep_R.csv** a odpovídá názvu konkrétního videa, ze kterého byl vytvořen. Navíc v tomto adresáři se také nacházejí soubory **gest_ID_il_E_per_NAME_rep_R.jpg**, ve kterých jsou zakresleny nalezené body z konkrétního csv souboru. Tyto obrázky jsou určené pro přehled výsledků rozpoznávacího mechanismu a usnadňují tak rychlejší nalezení chybně rozpoznaných pozic rukou konkrétního videa.

6.3.1 Předpoklady výsledků automatických testů rozpoznávacího mechanismu

Z důvodu efektivity práce a samotného vývoje rozpoznávacího mechanismu se zavedl předpoklad, že pro výsledky počátečních automatických testů (prvních iterací) daného vyvíjeného algoritmu se bude předpokládat průměrná spodní mez úspěšnosti 50% z výsledné rozpoznávací logiky HMM, která se bude trénovat a testovat z právě vytvořených sekvencí rozpoznávacího mechanismu. Pro tento předpoklad bylo zavedeno rozšíření, které usnadňuje získání správné cesty vývoje tím způsobem, že pro zmíněnou pravděpodobnost úspěšného rozpoznání byly použity sekvence pěti různých gest. Předpoklad byl stanoven hlavně proto, aby se ze začátku cesta vývoje rozpoznávacího mechanismu, která nevedla ke správnému

řešení, v čas odstranila a vývoj se soustředil pouze na řešení, které mělo perspektivu. Celý předpoklad a jeho rozšíření bylo stanoveno až po konzultaci s vedoucím práce.

6.3.2 Postup provedení automatických testů

V této kapitole bude objasněn postup spouštění automatických testů. Postup se skládal ze dvou hlavních částí a to získání dostatečného počtu testovacích videí a následné testování rozpoznávacího mechanismu právě na těchto získaných videích.

Ke správnému otestování rozpoznávacího mechanismu bylo potřeba získat dostatečné množství sekvencí pro trénování HMM a následného jejího testování. Tedy celková sada sekvencí musela být dostatečně velká. Proto bylo vytvořeno přibližně 30 až 35 videí provedení jednoho gesta, celkově 300 až 350 videí.

Pro vytvoření těchto videí bylo osloveno celkem 9 figurantů, z toho 4 ženy a 5 mužů. Většina z těchto figurantů měla věk 20 až 26 let, ale dva z těchto figurantů byly staršího věku a to konkrétně jeden muž měl 57 let a jedna žena 51 let.

Dále natáčení videí byla použita 2 prostředí:

- Místnost, kdy byl večer a venku nesvítilo slunce. Tedy veškeré osvětlení bylo umělé (světlo z lustru).
- Místnost, kdy byl slunečný den a venku svítilo slunce. Tedy v místnosti se nesvítilo žádným umělým osvětlením.

Při natáčení veškerých videí byly splněny všechny podmínky popsané v části 6.3. Zařízení na které se videa natáčely bylo „Xiaomi Redmi 2“. Jedná se o zástupce běžného mobilního zařízení s procesorem „Qualcomm Snapdragon 410 (4x Cortex-A53, 1,2 GHz, 28nm)“, 2 GB operační pamětí, s rozlišením 8 Mpx zadní kamery, kde je průměrně 31 snímků za sekundu při snímání videa.

Postup získání testovacích videí

1. Pro zjednodušení se snímala větší videa, kde figuranti postupně střídali a vždy několikrát znakovali jedno gesto vícekrát.
2. Po získání těchto větších videí byly pomocí programu „ELAN Linguistic Annotator“ vytvořeny xml anotační soubory pro jednotlivá videa.
3. Poté byly videa a anotační xml soubory přesunuty do připravené adresářové struktury popsané v části 6.3.
4. Nakonec byl spuštěn Bash skript `runParseVideosByAnnotation.sh` (popsán v kapitole 5.3) a tím se vytvořil zmíněný počet testovacích videí.

Postup testování rozpoznávacího mechanismu

Jednotlivé iterace vývoje konkrétních algoritmů rozpoznávacího mechanismu probíhaly tímto způsobem:

1. V programu `basedOnMoveDetector.cpp` se provedli konkrétní optimalizační změny rozpoznávacího mechanismu a program se přeložil.

2. Spustil se Bash skript `runTestOfRecognizeProcess.sh`, který vytvořil csv soubory s pozicemi rukou k daným videím.

3. Pětkrát za sebou se spustil balíček `TestOfHMM.jar`. Vypočítala se průměrná úspěšnost rozpoznávacího mechanismu z těchto 5-ti spuštění. Průměrné výsledky úspěšnosti rozpoznávání se zaznamenali a porovnaly s předchozí iterací. V případě zlepšení, se změny zanechaly, jinak se zahazovali. Navíc u nových algoritmů rozpoznávacího mechanismu se testovalo, jestli byl splněný počáteční předpoklad popsán výše 6.3.1. Pokud se vývoj daného algoritmu dostal do bodu, kdy stále nedocházelo k výraznému zlepšení tohoto algoritmu, tak se přestalo u daného algoritmu pokračovat ve vývoji, jinak se zpravidla pokračovalo bodem 1.

6.3.3 Výsledky automatických testů finálního rozpoznávacího mechanismu

Při testování pomocí automatických testů se využil druhý ze zmíněných typu testů v úvodu kapitoly 6, tj. testovala se metrika **úspěšnosti** rozpoznávacího mechanismu, který se pak využil v aplikacích Gesture Translator App. Automatické testování finálního rozpoznávacího mechanismu proběhlo s videi pouze s jedním z figurantů, jedním mužem. To z toho důvodu, že jeden z cílů aplikace Gesture Translator je zaměřit se na překlad znakového jazyka v prostředí osob se speciálními vzdělávacími prostředky, kde se očekává, že každá osoba pro význam stejného gesta, provádí gesto individuálním způsobem (viz kapitola 3.5).

Nastavení HMM pro finální testování rozpoznávacího mechanismu

Pro automatické testování finálního algoritmu je také potřeba zadefinovat způsob nastavení HMM. Ze samotné definice HMM je možné měnit tyto údaje:

- počet skrytých stavů v HMM
- počet symbolů abecedy generovaného řetězce

Nakonec je však také možné měnit **maximální počet iterací**, které se nastavují v Baum-Welsch algoritmu pro trénování HMM (viz kapitola 5.2.2 v sekci „Implementace rozpoznávací logiky v Gesture Tranlator Manager“).

Zde jsou vypsány všechny rozsahy zmíněných parametrů, které se využily při závěrečném testování rozpoznávacího mechanismu:

- počet skrytých stavů v HMM - 5, 10, 15
- počet symbolů abecedy generovaného řetězce - 7, 15, 20, 30, 50
- maximální počet iterací - 50, 80, 100, 200, 300

Tedy vniká zhruba 75 možných kombinací. Vhledem k tomu, že samotný výpočet trénování HMM trval dlouho, všechny kombinace vyzkoušené nebyly. Nejprve se totiž pro trénování HMM testoval vliv počtu skrytých stavů a počet symbolů abecedy generovaného řetězce (u těchto testů byla nastaven maximální počet iterací na 100). Poté se vybraly dvě kombinace s nejúspěšnějším rozpoznáváním a u těchto kombinací se dále měnil parametr maximálního počtu iterací. Nakonec se vybrala kombinace všech tří parametrů, která měla nejlepší úspěšnost rozpoznávání.

počet stavů	počet symbolů	maximální počet iterací	průměrná úspěšnost rozpoznání (v %)
5	7	100	41,3043472
5	15	100	49,5652164
5	20	100	51,304348
5	30	100	56,3043484
5	50	100	57,173912
10	7	100	41,9565224
10	15	100	58,0434776
10	**20**	**100**	**62,1739124**
10	30	100	55,6521724
10	50	100	59,7826064
15	7	100	44,7826088
15	15	100	50,4347828
15	20	100	56,5217392
15	**30**	**100**	**61,6956516**
15	50	100	60,652174

Tabulka 6.2: Výsledky pro výběr kandidátů do druhého testování

Získané data z automatických testů finálního rozpoznávacího mechanismu

Vzhledem k tomu, že tento test byl směrodatný a určoval úspěšnost celého finálního rozpoznávacího mechanismu výsledných aplikací Gesture Translator App, byl proveden v rozsahu nastavení parametrů HMM, který je uveden v předešlé kapitole. Tedy se celkem provedlo 15 testů, kde se zkoumal vliv počtu skrytých stavů HMM a počtu symbolů abecedy generovaného řetězce. V tabulce 6.2 lze přehledně vidět výsledky tohoto testování. Dále jsou v tabulce vyznačení dva kandidáti s nejlepší úspěšnosti a jsou vybráni pro test vlivu parametru maximálního počtu iterací. V další tabulce 6.3 vidíme zhodnocení testů, kde se měnil již jen parametr maximálního počtu iterací u vybraných kandidátů.

počet stavů	počet symbolů	maximální počet iterací	průměrná úspěšnost rozpoznání (v %)
10	20	50	55,6521736
10	20	80	53,6956508
10	**20**	**100**	**62,1739124**
10	20	200	55,21739
10	20	300	58,2608696
15	30	50	57,608692
15	30	80	54,9999976
15	30	100	61,6956516
15	30	200	60,869566
15	30	300	51,7391288

Tabulka 6.3: Testování parametru maximálního počtu iterací u zvolených kandidátů

Z tabulky 6.3 lze zpozorovat, že nejvhodnější kombinace má úspěšnost rozpoznávání 62,1739124%, kde se u HMM nastaví 10 skrytých stavů, 20 symbolů abecedy generova-

ného řetězce a u Baum-Welsch algoritmu je nastaveno 100 u parametru maximálního počtu iterací. Toto nastavení je použito i v aplikaci Gesture Translator Manager, ve které je rozpoznávací logika HMM implementovaná (viz kapitola 5.2.2 v sekci „Implementace rozpoznávací logiky v Gesture Tranlator Manager“).

6.3.4 Zhodnocení výsledků finálních automatických testů

Z výše popsaných tabulek je tedy zřejmé, že při ideálním nastavení HMM a parametru maximálního počtu iterací u Baum-Welsch algoritmu při použití videí jednoho figuranta je úspěšnost rozpoznání 62,17391245%. I když tato hodnota není příliš vysoká je pro rozsah této diplomové práce přijatelná.

6.4 Zhodnocení aplikací Gesture Translator App a Gesture Translator Manager

Sada aplikací Gesture Transaltor App a Gesture Translator Manager byla vyvinuta s hlavním cílem vytvořit překladač znakového jazyka v prostředí osob se speciálními vzdělávacími prostředky, tedy se zaměřit na konkrétní znakující osobu, u které se očekává, že gesta prováděná touto osobou mohou být velmi individuální od obecného vzoru gesta (viz kapitola 3.5).

Vyvinuté aplikace Gesture Translator Manager a Gesture Translator App s mechanismem rozpoznávání, u kterého se otestovala správná funkčnost pomocí automatických testů, byly také nasazené na několik zařízeních operačního systému Android a iOS, kde se potvrdila funkcionalita vycházející z automatických testů.

V tabulce 6.4 je možné vidět porovnání vyvinuté sady aplikací Gesture Transaltor s existujícími aplikacemi, popsaných v kapitole 3.4.

	MotionSavvy: UNI	Mimix Sign Language Translator**	sada aplikací Gesture Translator
Cena	11976 Kč*	96 Kč*	zdarma
Multiplatformnost	Android	Android	Android/iOS
Speciální hardware	ano	ne	ne
Znakový jazyk	ASL, SEE	ASL	mimo jiné i ČZJ
Překlad znakového jazyka	ano	ne	ano (ale jen po jednotlivých znacích)
Prostředí lidí se specifickými vzdělávacími potřebami	ne	ne	ano

* Přibližná cena v českých korunách po převodu z Amerického dolaru (při kurzu $1:24 Kč).

** Tato aplikace se uvádí z důvodu popsaného v kapitole 3.4.1

Tabulka 6.4: Srovnání výhod a nevýhod stávajících aplikací s vyvinutou sadou aplikací Gesture Transaltor

Vzhledem k cíli sadě aplikací Gesture Tranlator Manager a Gesture Translator App popsaný rozpoznávací mechanismus (viz kapitola 5.2.3) se soustřeďuje pouze na pohyb rukou, který je základní očekávanou komponentou provádění gest v prostředí, pro které je aplikace vyvíjena. Další komponenty českého znakového jazyka, jako je tvar rukou, poloha

rukou k tělu (vzdálenost rukou od těla či orientace dlaně a prstů) se můžou u jednoho konkrétního uživatele odlišovat při vyjádření stejného gesta, proto se na ně algoritmus rozpoznávacího mechanismu aplikace nezaměřuje.

Dále pak aplikace Gesture Translator App byla zamýšlena pro běžná mobilních zařízení s využitím standardní kamery zařízení, proto by ani nebylo možné získání prostorových informací, jako vzdálenost ruky od těla (uvedené výše), např. v porovnání s řešením *MotionSavvy:UNI*, kde je zaveden speciální hardware a prostor před uživatelem je při překladu zpracováván.

Nicméně v rámci práce byla také snaha vytvořit takové rozšíření, kdy by aplikace byly používány bez i-CT Frameworku, kde nemusí být přítomna individuální gesta pro daného uživatele. V tomto případě je zde na druhou stranu vhodné do rozpoznávacího mechanismu co nejvíce komponent českého znakového jazyka zahrnout. Toto ale nebyl cíl této práce, proto se předmětem užití tohoto rozšíření text práce dál nebude zabývat. Avšak toto rozšíření je vhodnou motivací pro další vývoj této práce a to i při užití napříč různými uživateli (bez jejich individualizovaných gest), dá ale se také předpokládat, že pro dobré výsledky bude třeba upravit stávající algoritmy pro tento případ užití.

Kapitola 7

Závěr

Cílem této diplomové práce bylo seznámit se s požadavky návrhových principů počítačové terapie, s problematikou speciálních vzdělávacích potřeb u osob s mentálním postižením, a také s požadavky na návrh a vývoj aplikací z pohledu interakce člověka s počítačem. Dalším cílem bylo prostudovat stávající aplikace zabývající se překladem znakové řeči na operačních systémech Android a iOS.

Výsledkem jsou aplikace i-CT Framework (který byl vyvíjen ve spolupráci s Janem Kalinou), a dále Gesture Translator Manager a Gesture Translator App, díky kterým je možné překládat znakový jazyk osob se speciálními vzdělávacími potřebami. Aplikace i-CT Framework se zaměřuje na správu osob s mentálním postižením: jde především o nástroj pro asistenty, kteří mohou jak spravovat, tak následně spouštět výukové aplikace svým klientům. V neposlední řadě se též jedná o bezpečnostní prostředek, který brání nedovolenému přístupu osob s mentálním postižením k citlivým datům výukových aplikací. Aplikace byla otestována v reálném prostředí osob s mentálním postižením z pohledu metriky použitelnosti (viz kapitola 6.1.2). Díky příznivým výsledkům těchto testů bylo shledáno, že aplikace i-CT Framework je pro toto prostředí a využití přínosná. Celkově bylo potvrzeno, že vývoj vycházející z návrhových principů počítačové terapie je vhodným prostředkem pro aplikace pro osoby se specifickými vzdělávacími potřebami.

Aplikace Gesture Translator Manager je určena pro správu gest a sekvencí reprezentujících tato gesta. Gesta je zde možné ze sekvencí natrénovat pomocí skrytých markovských modelů, a tím tak zaručit jejich správné rozpoznávání. Druhá aplikace Gesture Translator App je zaměřena na rozpoznávání polohy rukou z kamery v zařízení pomocí knihovny OpenCV. Aplikace Gesture Translator Manager, která vzájemně komunikuje s druhou aplikací Gesture Translator App, je navíc považována za jednu z výukových aplikací i-CT Frameworku, a lze ji tedy bezpečně používat z i-CT Frameworku pro překlad znakového jazyka při práci s lidmi se speciálními vzdělávacími potřebami. Směrodatné pro aplikace Gesture Translator Manager a Gesture Translator App bylo testování formou automatických testů, kdy byl testován rozpoznávací mechanismus zabudovaný v těchto aplikacích. Tyto automatické testy proběhly s relativně pozitivními výsledky, a lze tedy říci, že v rámci rozsahu práce byly požadavky na tyto aplikace splněny.

Všechny aplikace této práce jsou nyní k dispozici jako open source pro operační systémy Android a iOS pod licencí GNU GPL verze 3, což byl jeden z cílů této práce, který vychází z návrhových principů počítačové terapie.

Případné rozšíření aplikací by mohlo být provedeno formou zobecnění rozpoznávacího mechanismu aplikace Gesture Translator App pro širší cílovou skupinu, jako jsou lidé bez speciálních vzdělávacích potřeb. To by například bylo možné získáváním dalších infor-

mací z obrazu kamery mobilního zařízení, nejen tedy pohyb rukou, ale i další komponenty znaků, jako natočení rukou či jejich tvar při provádění gesta.

Literatura

[1] *Android Studio* [online]. [Online; navštíveno 20.05.2017].
URL https://developer.android.com/studio/index.html

[2] *Xcode 8* [online]. [Online; navštíveno 20.05.2017].
URL https://developer.apple.com/xcode/

[3] *Apple special needs summit Praha* [online]. 2017, [Online; navštíveno 20.05.2017].
URL http://www.i-sen.cz/clanky/apple-special-needs-summit

[4] *Clustering* [online]. 2017, [Online; navštíveno 31.01.2017].
URL http://docs.opencv.org/2.4/modules/core/doc/clustering.html#kmeans

[5] *Deliver native Android, iOS, and Windows apps, using existing skills, teams, and code* [online]. 2017, [Online; navštíveno 31.01.2017].
URL https://www.xamarin.com/platform

[6] *Getting Started with the NDK* [online]. 2017, [Online; navštíveno 31.01.2017].
URL https://developer.android.com/ndk/guides/index.html

[7] *Introduction to Oracle Mobile Application Framework: Introduction to Mobile Application Framework* [online]. 2017, [Online; navštíveno 20.05.2017].
URL http://docs.oracle.com/middleware/maf213/mobile/develop-maf/maf-about.htm#ADFMF1152

[8] *iSEN články* [online]. 2017, [Online; navštíveno 31.01.2017].
URL http://www.i-sen.cz/rubriky/clanky

[9] *Localizing MAF Applications: What You May Need to Know About XLIFF Files for iOS Applications* [online]. 2017, [Online; navštíveno 20.05.2017].
URL https://docs.oracle.com/middleware/maf210/mobile/develop/maf-apps-localize.htm#ADFMF293

[10] *OpenCV4Android SDK* [online]. 2017, [Online; navštíveno 31.01.2017].
URL http://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/O4A_SDK.html#o4a-sdk

[11] *Transform your enterprise with the power of mobile* [online]. 2017, [Online; navštíveno 31.01.2017].
URL http://www.redhat.com/en/technologies/mobile

[12] *Xamarin.Forms* [online]. 2017, [Online; navštíveno 20.05.2017].
URL https://www.xamarin.com/forms

[13] *Xamarin is now a member of the Visual Studio family.* [online]. 2017, [Online; navštíveno 31.01.2017].
URL https://www.xamarin.com/compare-visual-studio

[14] Nusirwan Anwar bin Abdul Rahman, K. C. W.; See, J.: *RGB-H-CbCr Skin Colour Model for Human Face Detection.* Diplomová práce, Multimedia University, 2014.
URL http://pesona.mmu.edu.my/~johnsee/research/papers/files/rgbhcbcr_m2usic06.pdf

[15] Bednařík, M.: *Péče o hluchoněmé děti.* Ústřední péče o hluchoněmé, 1940.

[16] Bouguet, J.-Y.: Pyramidal Implementation of the Lucas Kanade Feature Tracker.
URL http://vision.ssu.ac.kr/LecData2015-2/grad_cv/Features/KLTracker%EB%85%BC%EB%AC%B8%20-%20Optical%20Flow.pdf

[17] Cheng, Y.: Mean Shift, Mode Seeking, and Clustering. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, ročník 17, č. 8, 1995: str. 790–799.

[18] Fiala, J.; Kočí, R.: Computer as Therapy : in role of alternative and augmentative communication. In *Proceedings of 4th International Conference on Advanced in Computing & Emerging E-learning Technology (ICACET 2014)*, *1*, ročník 18, Singapore, 2014, s. 34–42, iSSN 2091-1610.

[19] Fiala, J.; Kočí, R.: Počítačová terapie jako koncept nové formy terapie pro osoby s mentálním postižením: teorie i praxe. *Journal of Technology and Information Education*, ročník 6, č. 1, 2014: s. 89–103, ISSN 1803-537X.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10718

[20] Fiala, J.; Zendulka, J.: Mentally challenged as design principles and models for their applications. *Applied Computer Science*, ročník 12, č. 4, 2016: s. 28–48, ISSN 1895-3735.
URL http://www.fit.vutbr.cz/research/view_pub.php.cs?id=11129

[21] Jakob Nielsen, P.: *Usability 101: Introduction to Usability* [online]. 2017, [Online; navštíveno 20.05.2017].
URL https://www.nngroup.com/articles/usability-101-introduction-to-usability/

[22] Kalina, J.: *Vývoj i-CT frameworku a jeho aplikace pro komunikaci typu ANO/NE.* Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2015.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=18385

[23] Meca, V.: *Aplikace pro výuku znaku do řeči pro osoby s mentálním postižením.* Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2013.
URL http://www.fit.vutbr.cz/study/DP/BP.php?id=5517

[24] Mellado, I.: *Getting started with Computer Vision on iPhone* [online]. 2017, [Online; navštíveno 31.01.2017].
URL http://www.ignaciomellado.es/blog/Getting-started-with-Computer-Vision-on-iPhone#related-code

[25] Moriyah, N.: *Primer to Kalman filtering: a physicist perspective*. New York: Nova Science Publishers, 2011, iSBN 978-1-61668-311-5.

[26] Papík, R.: *Strategie vyhledávání informací a elektronické informační zdroje*. Brno: Tribun EU, první vydání, 2011, ISBN 978-80-7399-338-2.

[27] Park, E.; Sehdev, N.; Frogoso, R.: *American Sign LanguageTranslator* [online]. 2017, [Online; navštíveno 20.05.2017].
URL http://www-cs.engr.ccny.cuny.edu/~csjie/cap/f16_des_pres/SignLanguageTranslator.pdf

[28] Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, ročník 77, č. 2, 1989: s. 257–286.

[29] Taft, D. K.: *Microsoft Makes Xamarin free in Visual Studio, Open-Sources SDK* [online]. 2016, [Online; navštíveno 20.05.2017].
URL http://www.eweek.com/developer/microsoft-makes-xamarin-free-in-visual-studio-open-sources-sdk.html

[30] Tůma, J.: *Aplikace pro YES/NO alternativní komunikaci pro osoby s mentálním postižením*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2013.
URL http://www.fit.vutbr.cz/study/DP/BP.php?id=4681

[31] Ujbányai, M.: *Programujeme pro Android*, ročník 1. Praha: Grada, 2012, ISBN 978-80-247-3995-3.

[32] Vejtasa, O.: *Aplikace pro alternativní a augmentativní komunikaci pro osoby s mentálním postižením*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2013.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=12342

[33] Vrbová, R.: *Metodika práce se žákem s narušenou komunikační schopností*. Olomouc: Univerzita Palackého v Olomouci, první vydání, 2012, ISBN 978-80-244-3312-7, 126 s.

[34] Zivkovic, Z.: *Improved Adaptive Gaussian Mixture Model for Background Subtraction*. Diplomová práce, Intelligent and Autonomous Systems Group, 2004.

[35] Ústav zdravotnických informací a statistiky ČR: *MENTÁLNÍ RETARDACE (F70–F79)* [online]. 2014, [Online; navštíveno 20.05.2017].
URL http://www.uzis.cz/cz/mkn/F70-F79.html

[36] Římalová, L. S.: *Bádání o jazycích a literaturách*. Praha: Univerzita Karlova, 2002, ISBN 80-7308-029-x.

[37] Škodová, E.; Jedlička, I.: *Klinická logopedie*. Opera linguae bohemicae studentium, Portál, druhé vydání, 2007, ISBN 978-80-7367-340-6.

[38] Švarcová, I.: *Mentální retardace*. Portál, druhé vydání, 2011, ISBN 978-80-7367-889-0, 24–32 s.

Přílohy

Příloha A

Obsah CD

Na přiloženém CD nachází zdrojové soubory aplikace včetně technické zprávy a všech souboru README.txt (u každé aplikace je přiložen jeden z nich).

```
/xmecav00
  /src
    /iCTFramework
      README.txt
      ...
    /GestureTranslatorManager
      README.txt
      ...
    /GestureTranslatorApp
      /android
        README.txt
        ...
      /ios
        README.txt
        ...
  /doc
    DIP_Meca.pdf
    /src - latex
```

Ukázka kódu A.1: Obsah adresáře na přiloženém CD

- `xmecav00/src/iCTFramework` - zdrojové kódy aplikace iCT-Framework
- `xmecav00/src/GestureTranslatorManager` - zdrojové kódy aplikace Gesture Translator Manager
- `xmecav00/src/GestureTranslatorApp/android` - zdrojové kódy aplikace Gesture Translator App pro android
- `xmecav00/src/GestureTranslatorApp/android` - zdrojové kódy aplikace Gesture Translator App pro iOS

Příloha B

Manuál použití aplikací Gesture Transaltor Manager a Gesture Translator App

V této příloze bude popsán způsob použití aplikací Gesture Translator Manager a Gesture Translator App.

B.1 Případ překladu

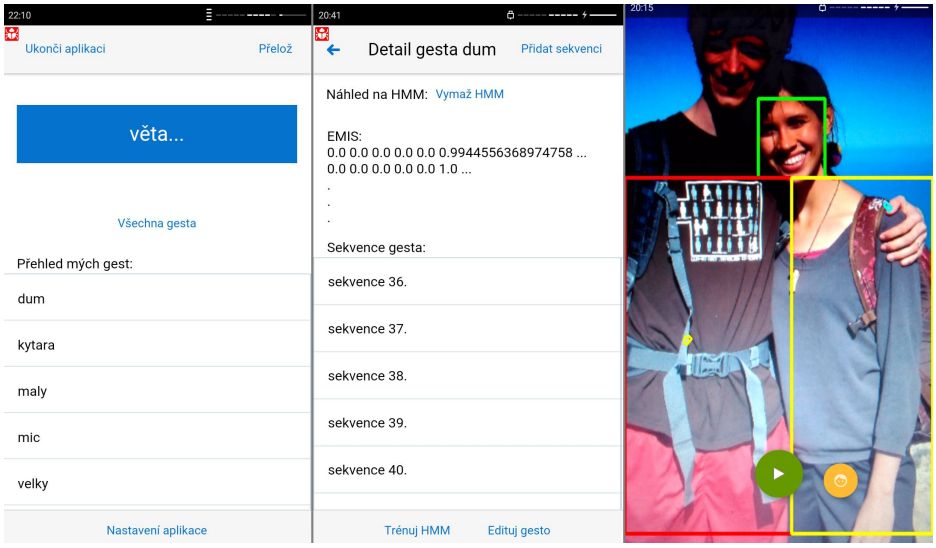
1. Po spuštění aplikace Gesture Translator Manager se objeví základní obrazovka aplikace a jdou vidět všechny gesta, která se nacházejí v aplikaci. Je-li spuštěná aplikace pomocí iCT-Frameworku je možné vidět seznam aktuálních gest, které jsou přiřazeny k danému uživateli (viz obrázek B.1).

2. Dále je pak pro překlad nutné kliknout na tlačítko **přelož** umístěné v pravém horním rohu. Touto činností se vyvolá aplikace Gesture Translator App, ve které se bude zaznamenávat sekvence gesta.

3. Poté je třeba kliknout na zelené tlačítko přehraj. Aplikace začíná zaznamenávat sekvenci gesta. Avšak pro samotné zaznamenávání je nutné najít hlavu osoby, která se snímá. Až po nalezení této hlavy se začíná zaznamenávat sekvence. Nalezení hlavy se však může již provést před samotným záznamem sekvence (viz obrázek B.1 napravo) a proto, je možné aktuální nastavení hlavy vynulovat pomocí tlačítka inicializace hlavy.

4. Pro skončení zaznamenávající sekvence stiskněte tlačítko rozpoznat (pause), poté se uživateli zobrazí všechny zaznamenané body rukou, ale už v normalizované podobě. Uživatel může vybrat jestli danou sekvenci rozpoznávat či provést záznam sekvence ještě jednou. Při kladném potvrzení rozpoznávání sekvence je zpět uživatel přesunut do aplikace Gesture Translator Manager, kde se po chvíli na obrazovce objeví název provedeného gesta. Pokud se ale daná sekvence nerozpozná, uživateli se zobrazí chybové hlášení **Špatně zadané gesto, prosím opakujte.**.

5. Vymazání věty s názvy rozpoznaných gest je možné kliknutím na větu a následným potvrzením.



Obrázek B.1: Obrazovka při spuštění aplikace Gesture Translator Manager.

B.2 Správa gest

V případě, že je přihlášen asistent v aplikaci Gesture Translator Manager v režimu správy aplikace či přehlášený uživatel se přihlásil bez iCT-Frameworku a navíc jsou mu udělena práva pro editaci gest, je možno editovat stávající gesta aplikace.

Pro přechod do části aplikace, kde se dané gesta dají editovat slouží tlačítko **všechna gesta** na úvodní obrazovce. Pokud toto tlačítko není vidět, uživatel nemá dostatečná práva pro tuto operaci. Pak lze provádět operace:

- Přidej nové gesto:
  1. zvol tlačítko **přidej gesto**
  2. po vyplnění formuláře s názvem gesta a následným potvrzením tlačítkem **ulož** dané gesto ulož.
- Přidej několik gest ČJZ do aplikace ze souborů. Tato operace ale zabere více než 10 minut:
  1. zvol tlačítko **načti gesta ze souborů** a vyčkej přidání několika gest ČJZ do aplikace.
- Smaž gesto:
  1. posuň doprava daný řádek s názvem gesta
  2. zvol tlačítko **smazat**
- Edituj název gesta:
  1. zvol ze seznamu gesto pro editaci
  2. stiskni tlačítko **edituj gesto**
  3. napiš nový název gesta
  4. ulož

- Trénuj gesto gest:
  1. zvol ze seznamu gesto
  2. stiskni tlačítko **trénuj gesto** (minimum sekvencí pro trénink gesta 20)
  3. Po chvíli se objeví náhled jedné z matic natrénované HMM (obrázek B.1 uprostřed)

B.3 Správa sekvencí u daného gesta

V případě, že je přihlášen asistent v aplikaci Gesture Translator Manager v režimu správy aplikace či přehlášený uživatel se přihlásil bez iCT-Frameworku a navíc jsou mu udělena práva pro editaci gest, je možno editovat sekvence u gest aplikace

Pro přechod do části aplikace, kde se dané sekvence dají editovat slouží tlačítko **všechna gesta** na úvodní obrazovce. Po vybrání gesta (stisk názvu daného gesta) lze editovat dané sekvence gesta:

- Přidej novou sekvenci gesta:
  1. zvol tlačítko **přidej sekvenci**, otevře se aplikace Gesture Translator App.
  2. Poté je třeba kliknout na zelené tlačítko přehraj. Aplikace začíná zaznamenávat sekvenci gesta. Avšak pro samotné zaznamenávání je nutné najít hlavu osoby, která se snímá. Až po nalezení této hlavy se začíná zaznamenávat sekvence. Nalezení hlavy se však může již provést před samotným záznamem sekvence (viz obrázek B.1 napravo) a proto, je možné aktuální nastavení hlavy vynulovat pomocí tlačítka inicializace hlavy.
  3. Pro skončení zaznamenávající sekvence stiskněte tlačítko uložit (pause), poté se uživateli zobrazí všechny zaznamenané body rukou, ale už v normalizované podobě. Uživatel může vybrat jestli danou sekvenci chce uložit či provést záznam sekvence ještě jednou. Při kladném potvrzení uložení sekvence je zpět uživatel přesunut do aplikace Gesture Translator Manager, kde se sekvence uloží. Nová sekvence je vidět na konci seznamu všech sekvencí gesta.
- Smaž sekvenci:
  1. posuň doprava daný řádek s danou sekvencí
  2. zvol tlačítko **smazat**
- Přehraj sekvenci:
  1. zvol sekvenci ze seznamu sekvencí gesta, otevře se aplikace Gesture Translator App.
  2. Pro začátek přehrávání sekvence zvol tlačítko play (přehraj).
  3. Zde se může přehrávání sekvence pozastavit (tlačítko stop) a znovu pokračovat (tlačítko play) v přehrávání.
  4. V případě platformy iOS po přehrání sekvence se objeví výzva k uživateli, zda-li chce přehrát sekvenci znovu.
  5. Pro návrat zpět do aplikace Gesture Translator App je nutné stisknout tlačítko **zpět**.

B.4 Zbylé operace

V případě, že uživatel může nastavit aplikaci, se klikne na tlačítko **nastavení aplikace** a zde uživatel může uživatel nastavit, jestli chce filtrovat sekvence při rozpoznávání. Asistent přihlášený přes iCT-Framework v režimu správy může nastavit, zda-li nepřihlášený uživatel může editovat data aplikace (gesta, sekvence, nastavení aplikace).

V neposlední řadě je možné při přihlášení asistenta v režimu správy přes iCT-Framework nastavit konkrétní gesta uživateli, pro kterého připravuje prostředí aplikace a následným uložením změn. Tato operace se provádí zmačknutím tlačítka **nastavení uživatele**.

Anonymní uživatel může také zaregistrovat danou aplikaci Gesture Translator od iCT-Frameworku, pokud klikne na modrou oblast s textem **Registruj aplikaci do iCT-Frameworku**.