VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

UŽIVATELSKÉ ROZHRANÍ S VYUŽITÍM MOBILNÍHO ZAŘÍZENÍ A KAMERY

USER INTERFACE USING MOBILE DEVICE AND CAMERA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DALIBOR JELÍNEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. PAVEL NAJMAN,**
SUPERVISOR

BRNO 2018

Abstrakt

Tato diplomová práce se zabývá vývojem uživatelského rozhraní, které umožní ovládat mobilní zařízení bez použití rukou za pomocí kamery. Práce zahrnuje rešerši existujících řešení bezdotykového ovládání počítačových zařízení a technologií počítačového vidění relevantních k tomuto tématu. Výstupem praktické části je aplikace pro systém Android sloužící jako webový prohlížeč, který lze ovládat výhradně pohyby obličeje a hlasem.

Abstract

This Bachelor thesis deals with development of user interface which will allow to control mobile device using camera and without use of hands. Thesis includes research on existing non-contact user interface sollutions and relevat computer vision techlologies. The output of the practical part is Android application which serves as a web browser that can be controlled exclusively by facial movements and voice.

Klíčová slova

uživatelské rozhraní, Dlib, Google Vision, OpenFace, detekce obličeje, Android, kamera, počítačové vidění, bezkontaktní ovládání, rozpoznávání gest

Keywords

user interface, Dlib, Google Vision, OpenFace, face detection, Android, camera, computer vision, touchless interface, gesture recognition

Citace

JELÍNEK, Dalibor. *Uživatelské rozhraní s využitím mobilního zařízení a kamery*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Pavel Najman,

Uživatelské rozhraní s využitím mobilního zařízení a kamery

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Pavla Najmana a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Dalibor Jelínek
17. května 2018

Poděkování

Rád bych využil této příležitosti k poděkování původnímu vedoucímu této práce panu profesoru Pavlovi Zemčíkovi za zajímavé téma a podnětné nasměrování, mému vedoucímu panu inženýru Pavlovi Najmanovi za věcný a vstřícný přístup na všech konzultacích a mé přítelkyni Adéle a rodině za morální i materiální podporu během celého studia. Velmi si toho vážím.

Obsah

Kapitola 1

Úvod

S alternativními přístupy k uživatelskému rozhraní bez použití fyzického kontaktu se zařízením se v posledních letech setkáváme stále častěji. To je pochopitelné zejména ze dvou důvodů. Na rozdíl od klasických rozhraní tyto moderní technologie často vyžadují značný výpočetní výkon a také kvalitní senzorické výstupy. S postupujícím vývojem je uspokojení těchto požadavků stále snazší.

Druhým důvodem je paralelně rostoucí oblast uplatnění těchto technologií. Lze jmenovat například snahu rozšířit herní zážitek prostřednictvím pohybového senzoru Kinect, minimalizovat nezbytná periferní zařízení, kde jako vhodný příklad může posloužit bezpilotní letoun DJI Spark, který lze řídit bez vysílačky pouze pomocí gest, či fotoaparáty, které pořídí skupinovou fotografii až poté, co se všichni účastníci usmívají. Za velmi důležitou považuji také snahu poskytnout způsob ovládání různých zařízení i lidem s pohybovým omezením, pro něž jsou běžná uživatelská rozhraní nepoužitelná.

Tato práce se zabývá vývojem bezdotykového rozhraní pro mobilní zařízení. Několik řešení bezkontaktního rozhraní již existuje a jejich analýze se věnuje druhá kapitola, nicméně tato řešení často vyžadují buď externí a nákladná zařízení, nebo nejsou dostatečně spolehlivá.

Cílem této práce je návrh a implementace uživatelského rozhraní, které umožní ovládat mobilní zařízení bez fyzického kontaktu s tímto zařízením, tedy i lidem postiženým pohybovým omezením horních končetin. Práce si neklade za cíl kompletně nahradit univerzální uživatelské rozhraní dotykových obrazovek, ale vytvořit uživatelsky přívětivé rozhraní navržené pro konkretní aplikaci, a to bez nutnosti využívat jakákoliv externí hardwarová rozšíření současných běžných tabletů, netbooků či chytrých telefonů.

Výstupem práce je aplikace „Touchless browser“ sloužící jako webový prohlížeč pro OS Android, který lze ovládat pouze pohyby hlavy a mimikou. Pomocí sady gest navržené specificky pro tento účel a simulací kurzoru myši je možné vyhledávat, zobrazovat a procházet webový obsah. Aplikace mimo to disponuje funkcí rozpoznávání řeči, což umožňuje efektivní práci s interaktivními prvky, jako jsou vstupní textová pole nebo rychlé zadávání vyhledávacích dotazů.

Kapitola 2

Shrnutí dosavadního stavu

Tato kapitola v první části poskytuje příklady relevantních existující řešení zabývajících se ovládáním elektronických zařízení za pomocí kamery, nebo obecně bez použití rukou. Dále se zabývá dostupnými nástroji pro rozpoznávání obličeje použitelnými na platformě Android a v poslední části poskytuje obecný pohled na některé aspekty vývoje aplikací pro Android související s tématem práce.

2.1 Existující řešení bezkontaktních uživatelských rozhraní

2.1.1 Tobii Eye Tracking

Společnost Tobii[1] je švédskou firmou založenou roku 2001, která se zabývá eye trackingem, neboli detekcí směru pohledu uživatele. Toho lze využít pro různé výzkumné či statistické účely, ale také tato technologie poskytuje dobrý základ pro bezkontaktní uživatelské rozhraní, neboť směr pohledu může nahradit polohovací zařízení jako jsou myš nebo touchpad.

K rozpoznávání směru pohledu používá Tobii technologii pupil centre corneal reflection (PCCR). Systém spočívá v osvětlení očí zdrojem světla, který vytvoří dobře viditelné odlesky na rohovce a ty jsou pak snímány pomocí kamery. Ze vzájemné polohy odlesků na rohovce a zornici lze pak vypočítat vektor pohledu[2].

Jako světelný zdroj je v tomto případě použitý infračervený zářič, který promítá na rohovku odrazové vzory a pomocí pokročilých algoritmů s využitím fyziologického 3D modelu oka je odhadován cíl pohledu s vysokou přesností. Systém navíc provádí veškeré výpočty uvnitř externího modulu, díky čemuž není čerpán výpočetní výkon počítače.

Společnost nabízí celou řadu nástrojů poskytujících uživatelské rozhraní založené na eye trackingu. Kompletní řešení ve většině případu sestává z hardwarového zařízení v podobě lišty, která se nejčastěji umisťuje pod monitor (viz obrázek 2.1). V té je zabudovaný zdroj infračerveného záření a dvojice kamer s vysokým rozlišením. Softwarovou část pak tvoří aplikace poskytující dva různé přístupy k ovládání zařízení.

Prvním z nich je Gaze Selection. Tento systém funguje tak, že na obrazovku přidá lištu s běžně používanými příkazy na současných zařízeních jako jsou dvojklik, rolování nebo drag & drop. Uživatel poté postupuje takovým způsobem, že nejprve vybere pohledem na odpovídající ikonku na liště požadovaný příkaz a poté namíří svůj pohled na pozici na obrazovce do míst, kde chce příkaz provést. Vzhledem k tomu, že ovládací prvky běžných

[1] Tobbii https://www.tobii.com/group/about/

[2] Tobii eye tracker - princip https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/how-do-tobii-eye-trackers-work/

Obrázek 2.1: Hardwarová lišta s infrazářičem a dvěma kamerami

grafických uživatelských rozhraní jsou často miniaturní, Gaze Selection neprovádí příkaz bezprostředně po ustálení pohledu konkretním směrem, ale v tento moment pouze začne kontinuálně přibližovat cílený výřez obrazovky do určité míry určitou rychlostí, během čehož je možné cíl příkazu blíže specifikovat pohledem do určité části výřezu. Rychlost a míra přiblížení je otázkou uživatelského nastaveni.

Druhou variantou je Mouse Emulation. Jak už název napovídá, tento nástroj poskytuje plnou kontrolu nad běžným kurzorem myši, přičemž případné příkazy jsou vyvolány upřeným pohledem do místa, kde je vyvolání příkazu požadováno po určitou dobu. Výběr typu příkazu je opět řešen prvkem grafického uživatelského rozhraní, nicméně tentokrát se nejedná o lištu u kraje obrazovky, ale plovoucí blok, jehož pozici může uživatel snadno měnit opět pohledem.

První z přístupů je vhodný pro běžné používání a díky dvoukrokovému provádění příkazů je minimalizována pravděpodobnost jejich nechtěného provedení. Mouse Emulation je dle výrobce učený spíše pro použití například u grafických nástrojů a je údajně možné dosáhnout stejné přesnosti jako s hardwarovou myší, což ovšem vyžaduje to určitý tréning.

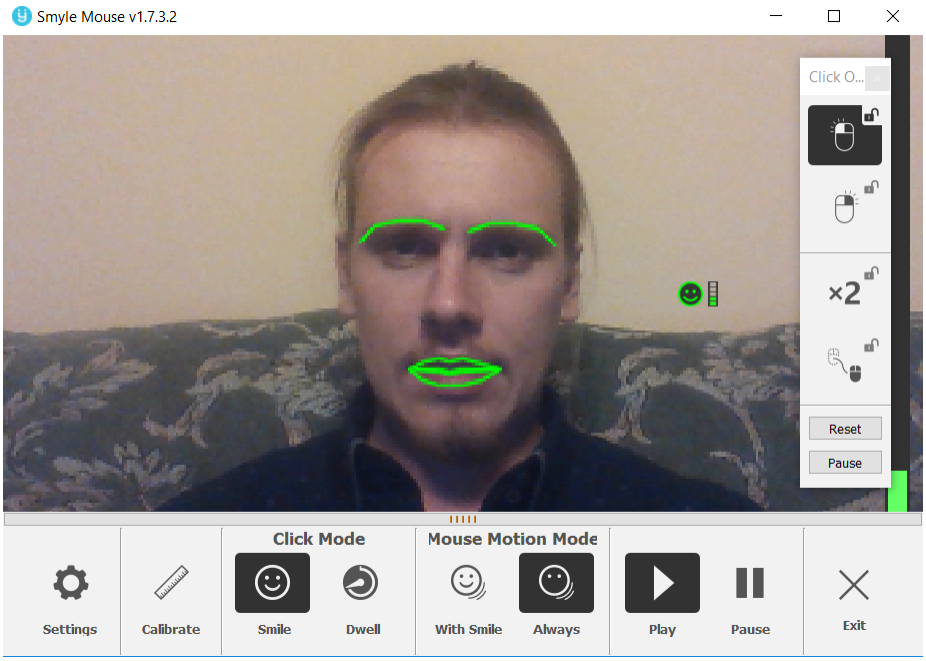
2.1.2 Smyle Mouse

Cílem této aplikace je nahradit počítačovou myš. Oproti předchozímu příkladu je Smyle Mouse[3] plně softwarové řešení a jedná se o poměrně jednoduchou aplikaci, která využívá rozpoznávání polohy hlavy a tvaru úst. Pohyb kurzoru se odvíjí od polohy hlavy vůči kameře a provedení příkazu (kliknutí) se realizuje simulovaným úsměvem. K dipozici jsou také nástroje pro dvojklik, rolování nebo drag & drop.

Aplikace nabízí dva přístupy k ovládání kurzoru. První z nich aktivuje jeho pohyb, kdykoliv se obličej uživatele pohne rychleji než je uvedeno v nastavení. V tomto režimu je možné k provedení příkazu jako je kliknutí použít úsměv.

Druhou alternativou je režim, při kterém je nutné aktivovat pohyb kurzoru úsměvem. Je přitom možné definovat, po jakou dobu úsměv musí trvat, aby došlo k „odparkování“ kurzoru. Pokud je tato doba trvání úsměvu překročena, aniž by uživatel pohnul hlavou, je

[3]Smyle Mouse https://smylemouse.com/index.php



Obrázek 2.2: Uživatelské rozhraní aplikace Smyle mouse

spuštěna druhá časomíra, která po jejím dokončení aktivuje režim drag & drop. Aplikace rozpoznává i „intenzitu“ úsměvu a je tedy možné nastavit při jaké intenzitě dojde k aktivaci. Zároveň tento přístup ale zamezuje použití úsměvu například ke kliknutí. Aplikace tedy poskytuje alternativní režim i pro tuto část ovládání. V tomto režimu je příkaz proveden vždy, když je kurzor po určitou dobu v nečinnosti.

Pokud chce uživatel jako příkaz definovat například dvojklik místo výchozího kliknutí, nebo mezi těmito režimy dynamicky přepínat, je možné si zobrazit grafickou lištu s podporovanými příkazy (viz obrázek 2.2 po pravé straně). Příkaz se poté vybírá kliknutím na ikonu příkazu. Celý reakční mechanismus tak sestává ze sady nastavitelných časomír detekce úsměvu a pohybu obličeje.

2.1.3 SmartNav

SmartNav je dalším technologicky odlišným řešením téhož problému. Hlavním úkolem tohoto systému je poskytnout způsob určování polohy kurzoru, který nevyžaduje použití rukou. Produkt je založený na zařízení, které připomíná webkameru a upevňuje se nejčastěji na vrchní část monitoru. Toto zařízení má v sobě zdroj infračerveného světla a CMOS senzor. Princip spočívá v tom, že uživatel umístí na některou část těla či oděvu drobnou reflexní samolepku, která odráží infračervené světlo zpět a to je zachycené senzorem. Výstup senzoru pak slouží k výpočtu odpovídajících pohybů kurzoru, jejichž citlivost je zvlášť nastavitelná v obou osách[4].

Samolepky jsou součástí balení a je možné je umístit na hlavu, ruce, brýle, nebo i kšiltovku, kterou má výrobce v nabídce rozšíření stejně jako speciální reflexní prsteny.

Kromě samotného polohování kurzoru je nicméně velmi žádoucí vlastností i provádění příkazů, jako je například kliknutí. SmartNav tento problém řeší nabídkou různých variant

[4]SmartNav http://www.naturalpoint.com/smartnav/products/howitworks.html

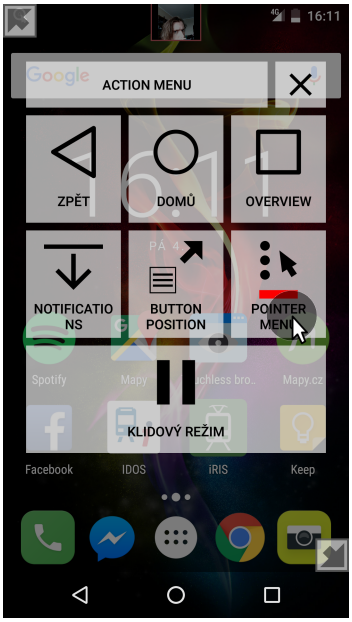
rozšíření. V nabídce je externí ruční tlačítko, pedál pro jednu nohu, dvojice pedálů umožňující definovat příkazy i pro různé kombinace stlačení nebo software rozpoznávající hlasové instrukce, které si může uživatel sám nadefinovat.

2.1.4 EVA Facial Mouse

EVA Facial Mouse je prvním a jediným zástupcem řešení pro mobilní zařízení fungující na platformě Android. Jedná se o aplikaci s otevřeným kódem pro OS Android kterou vyvinula společnost CREA Computer Systems za podpory španělské nadace Vodafone a získala celou řadu ocenění[5].

Aplikace využívá videa ze zabudované přední kamery k rozpoznávání pohybů hlavy. Po instalaci je uživatel proveden interaktivním návodem a nastavením, při kterém dojde ke kalibraci a představení základních funkcí. Následně se aplikace objeví v nabídce „Usnadnění“ v nastavení operačního systému.

Po spuštění se v rozích obrazovky zobrazí poloprůhledná tlačítka, která umožňují simulovat potažení prstem po dotykové obrazovce, které je v uživatelském rozhraní OS Andorid často používané například k přepínání ploch. V pravém dolním rohu je také kontextové tlačítko umožňující zobrazit menu akcí, které je zachycené na obrázku 2.3. K dispozici je standardní trojice tlačítek spodní navigační lišty (zpět, domů, přehled spuštěných aplikací), zobrazení systémových upozornění, možnost přesunout tlačítko akcí do protějšího rohu a také aktivovat režim „Pointer menu“ (viz níže) nebo klidový režim.



Obrázek 2.3: Uživatelské rozhraní aplikace Eva facial mouse

Příkaz kliknutí je v tomto systému vyvolán vždy, jakmile je kurzor po definovanou dobu v nečinnosti (dwell-click). Výjimku tvoří klidový režim, který lze aktivovat z menu akcí. V klidovém režimu je možné kliknout pouze na tlačítka uživatelského rozhraní této aplikace.

[5]EVA Facial Mouse https://github.com/cmauri/eva_facial_mouse

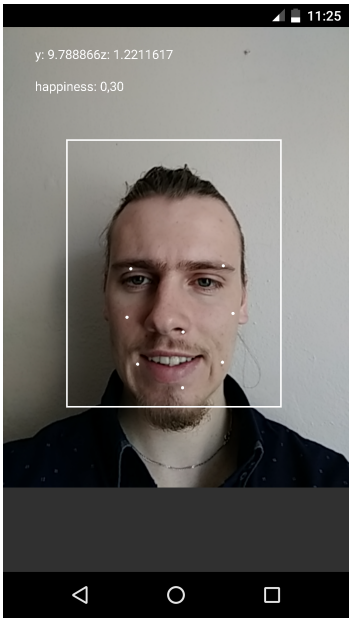
Protože v uživatelském rozhraní OS Android není výjimkou využití delšího stisku, je možné aktivovat režim „Pointer menu“. V tomto režimu je při každém kliknutí vyvolána kontextová nabídka s možností zvolit krátký nebo dlouhý stisk.

2.2 Dostupné technologie pro rozpoznávání obličeje v OS Android

V současné době existuje celá řada řešení, která se obecně zabývají tímto problémem. Každé řešení má ovšem svá specifika, která mají často zásadní vliv na jejich použitelnost na konkrétní problém. Pro účely využití informací o obličeji k ovládání uživatelského rozhraní jsou zcela kritickými parametry rychlost (téměř real-time), spolehlivost a detekce landmarků (klíčových bodů na obličeji) s velkou přesností nebo odhad rotace hlavy ve všech osách. Dalším omezujícím kriteriem je použitelnost na platformě Android. Byť Android nevylučuje použití knihoven napsaných v C++ díky nástroji NDK, je nezbytné pro tuto knihovnu i její závislosti vytvořit JNI (Java Native Interface), což nemusí byt vždy úplně snadné. Následující sekce se věnuje relevantním řešením pro účely této práce.

2.2.1 Google Mobile Vision API

Společnost Google se dlouhodobě zabývá počítačovým viděním. Kromě své cloud služby nyní navíc poskytuje speciálně pro mobilní zařízení i službu Mobile Vision API, která je dostupná i bez připojení k síti. Jedná se o API poskytující framework pro rozpoznávání objektu ve fotografiích či videu. V současnosti poskytuje nástroje pro detekci obličeje, QR či jiných podobných kódů a také ORC čtečku textů psaných latinkou[6].



Obrázek 2.4: Příklad použití Google Vision API

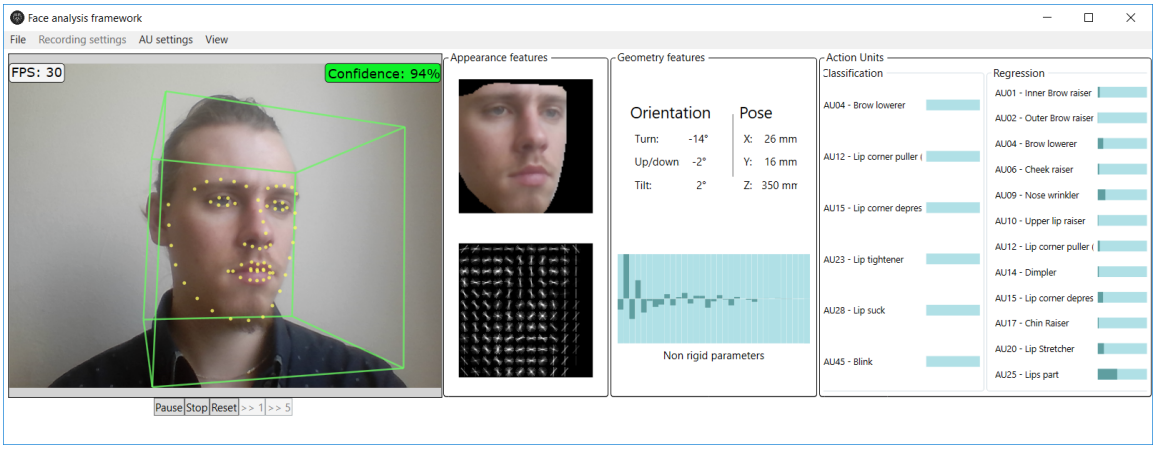
Vzhledem k tématu práce budou následující řádky věnovány pouze detekci obličeje. Ta poskytuje celou řadu užitečných informaci. Po úspěšném rozpoznání obličeje uživatel získá

[6]Google Mobile Vision https://developers.google.com/vision/

informace o pozici obličeje v rámci obrazu, jeho rozměrech a také jeho náklonu nebo orientaci. Ty je však bohužel nyní možné detekovat pouze v osách Y a Z. Pokud je obličej namířen čelem ke kameře, je možné detekovat osmici klíčových bodů umístěných na obou očích, kořenu nosu, na krajích a uprostřed úst a na tvářích. Obličej je možné detekovat i z profilu nebo pod úhlem. V takové situaci se však počet klíčových bodů snižuje. Zajímavou funkcí je též klasifikace určitých vlastností obličeje. API poskytuje funkce vracející informace o „míře“ úsměvu či otevřenosti očí. API Mobile Vision od Google je k dispozici též pro iOS, nicméně některé vlastnosti, jako je například detekce otevřených očí, jsou omezeny.

2.2.2 OpenFace

OpenFace je nástroj s otevřeným kódem vyvinutý za účelem počítačového vidění a strojového učení. Je určený zejména pro vývoj interaktivních aplikací založených na analyzování obličeje a jeho chovaní. Jedná se o první open source nástroj schopný detekovat landmarky (klíčkové body obličeje), odhadnout náklon a natočení hlavy ve všech osách, směr pohledu a také informace o takzvaných akčních jednotkách obličeje[7]. Algoritmus počítačového vidění, který je jádrem OpenFace, představuje „state-of-the-art“ výsledků ve všech výše zmíněných disciplínách.



Obrázek 2.5: Příklad použití Google Vision API

Tento nástroj navíc poskytuje výsledky takřka v reálném čase a jako vstupní data postačí fotografie v nízkém rozlišení či výstup z běžné webové kamery.

OpenFace využívá pro detekci landmarků CLNF (Conditional Local Neural Fields). Jedná se o instanci CLM (Constrainde Local Model) využívající pokročilých optimalizačních funkcí [1].

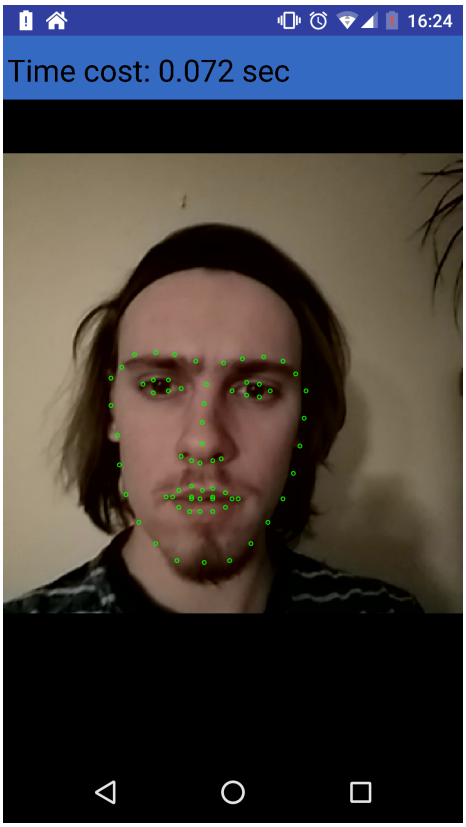
2.2.3 Dlib

Dlib[8] je moderní volně dostupná knihovna zahrnující širokou škálu algoritmů pro strojové učení, numerické početní úlohy nebo zpracování obrazu. Je napsaná v jazyce C++, pro který je primárně určena, nicméně existuje i rozhraní pro Python nebo Javu. Její kód není

[7] Akční jednotky popisují stav určitých částí obličeje, které mají vliv na celkový výraz. Jedná se například o otevření úst či mrknutí oka.

[8] Dlib http://dlib.net/

závislý na platformě. Je tedy možné ji kompilovat jak pro Windows, Linux, tak i Android [3].



Obrázek 2.6: Testování knihovny Dlib

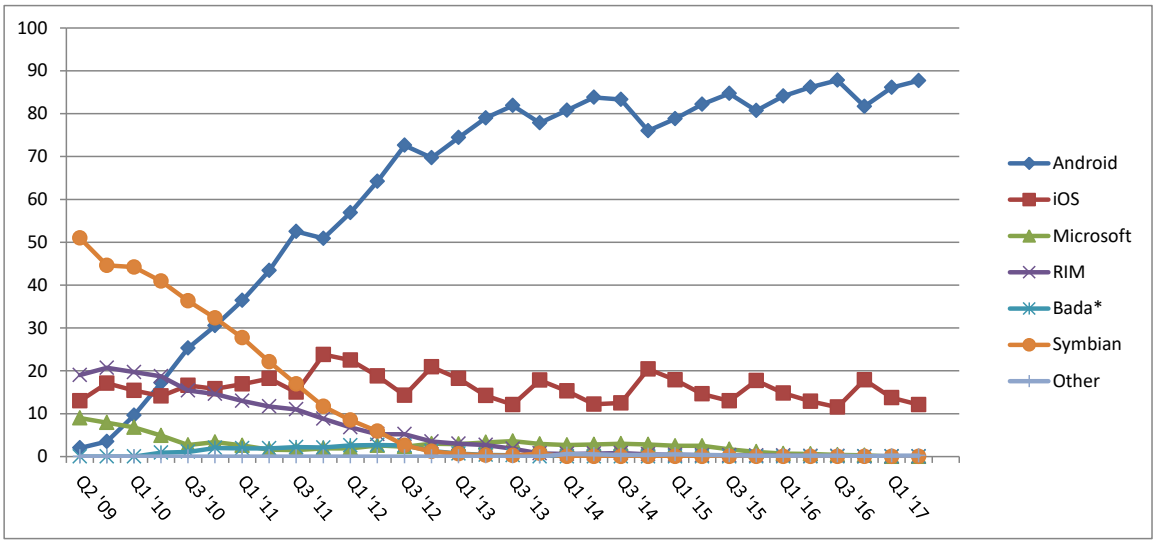
Dlib používá k detekci obličeje tzv. histogram orientovaných gradientů. Jedná se o popis objektů pomocí jednoduchých příznaků. Princip lze zjednodušeně popsat tak, že obraz je rozdělen na segmenty, pro něž jsou vypočteny vektory gradientů v rozsahu 0 - 180° rozdělené dle směru do devíti tříd neboli binů (tedy po 20°). Obraz lze pak popsat konkatenací histogramů všech segmentů. Ukázku je možné vidět na obrázku 2.5.

Výhodou teto metody je, že vykazuje velmi dobré výsledky v popisu kontur obličeje a není citlivá na množství světla nebo drobné odchylky [4].

Ke knihovně Dlib je k dispozici několik natrénovaných modelů[9]. „Nevýkonnější“ z nich je schopen detekovat 68 landmarků a je trénovaný na datové sadě W-300[10]. Zejména kvůli této datové sadě je však omezený pouze na nekomerční účely. Výsledek tohoto detektoru je možné vidět na obrázku 2.6. Součástí Dlib jsou též nástroje na vytváření a testování těchto datasetů.

[9] Natrénované modely pro Dlib https://github.com/davisking/dlib-models

[10] Datová sada W-300 https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/



Obrázek 2.7: Historický vývoj podílu mobilních OS na trhu[11]

2.3 Vývoj pro platformu Android

Následující sekce poskytuje obecný úvod do programování pro platformu Android a popisuje některá specifika a zajímavé oblasti tohoto vývoje, jejichž znalost byla využita při vytváření této práce.

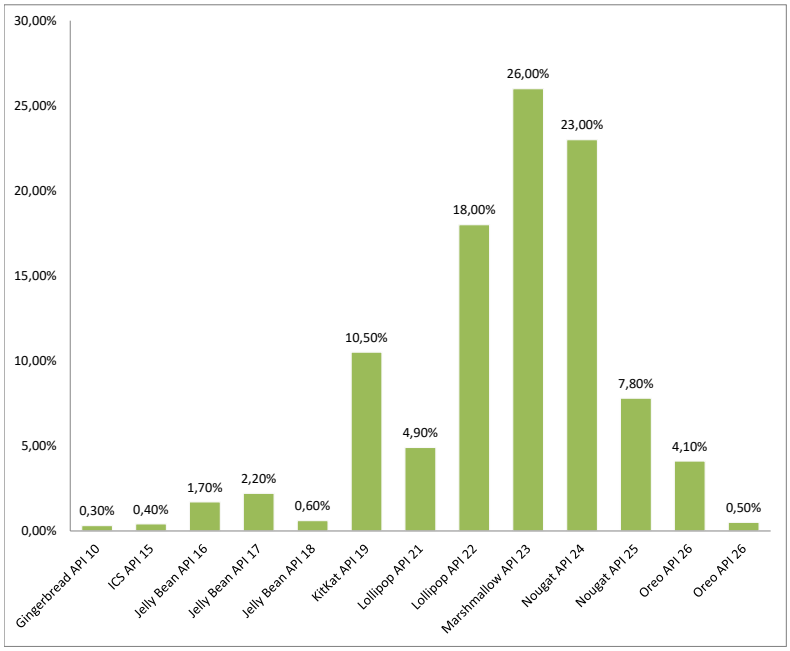
2.3.1 Operační systém Android

Android je operační systém založený na open source platformě vyvíjený společností Google a určený primárně pro mobilní zařízení. Vývoj pro Android byl původně vázán na jazyk Java, nicméně v poslední době se těší stále větší popularitě jazyk Kotlin vyvinutý společností JetBrains. Na konferenci Google IO roku 2017 byl tak uveden jako oficiálně podporovaný jazyk pro vývoj aplikací pro platformu Android[2]. Mezi charakteristické rysy tohoto OS patří podpora multitaskingu, přičemž každá aplikace běží ve vlastním virtuálním stroji, což umožňuje běh aplikací na pozadí. Díky otevřenému kódu je obvyklé, že si výrobci upravují prostředí ke svým potřebám[3]. Systém založený na linuxovém jádře byl poprvé představen v roce 2003 a první telefon s tímto OS byl uveden na trh koncem roku 2008. Od té doby si vybudoval dominantní místo na globálním trhu[11], jak je možná vidět z grafu 2.7.

2.3.2 Verzování

Za dobu svého působení bylo vydáno 8 verzí, přičemž každá verze je pojmenována po určité pochutině a zároveň disponuje číselným označením a označením podporované verze API. Mezi jednotlivými verzemi pak dochází k méně významným aktualizacím, které zpravidla poskytnou nové API. Při vývoji aplikace pak programátor musí zvolit, pro jaké API chce aplikaci vyvíjet a kolik API nazpět chce podporovat. Od toho se pak odvíjí jaké systémové funkce bude mít k dispozici. Dokumentace uvádí, že za optimální je považována podpora

[11] Dlouhodobá statistika podílů OS na trhu https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/



Obrázek 2.8: Rozdělení aktuálně používaných verzí OS Android[12]

90 % zařízení na trhu a zároveň poskytuje aktuální přehled procentuálního zastoupení používaných verzí[12]. Nejnovější verze nese název Oreo 8.1 přičemž odpovídající API má označení 27. Jeho používanost je však aktuálně kolem 0,5 %. Z grafu 2.8 lze pak odhadnout, že zpětná kompatibilita by měla být zajištěna alespoň 5 verzí nazpět.

2.3.3 Systém oprávnění přístupu

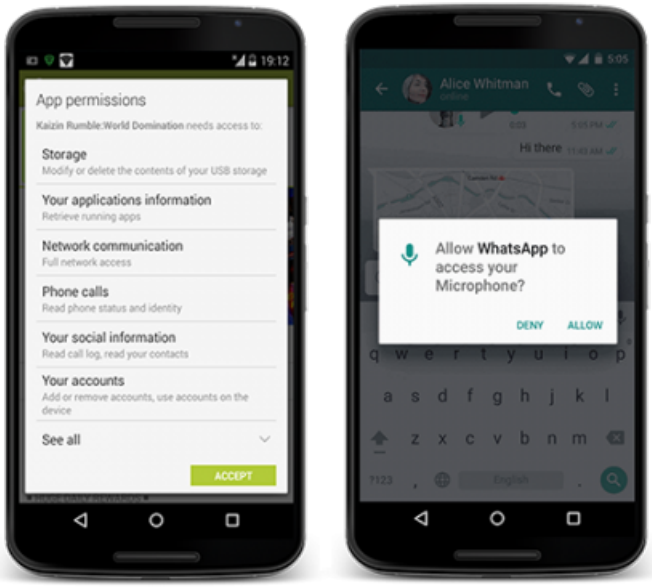
Z důvodu ochrany soukromí uživatele má Android systém pro správu tzv. oprávnění. Díky tomu je již před instalací aplikace uživatel informován k jakým datům (např. kontakty, SMS nebo SD karta) či systémovým funkcím (kamera, přístup k internetu...) bude mít aplikace přístup. V současné chvíli jsou oprávnění rozdělena do tří hlavních kategorií.

První kategorii je „Normal permissions“. Tato kategorie pokrývá oblasti, v nichž aplikace vyžaduje externí zdroje, nicméně jejich využití nepředstavuje pro uživatele velké riziko. Jedná se například o oprávnění přístupu k informacím o stavu k bluetooth nebo wifi, oprávnění využít vibrace telefonu, ale i přístup k internetu. Tato oprávnění jsou přidělena aplikaci automaticky během instalace.

Následující kategorie nese označení Signature permissions. Oprávnění z této kategorie jsou stejně jako u té předchozí přidělována v době instalace. Rozdíl je v tom, že jsou přidělena pouze v případě, že je aplikace podepsána stejným certifikátem jako aplikace, která toto oprávnění definovala. Tato kategorie má smysl především z toho důvodu, aby omezila přístup vývojářům aplikací třetích stran k některým kritickým systémovým službám, ale také proto, aby jim umožnila využívat tento systém k definování a spravování vlastních oprávnění.

Poslední kategorií je Dangerous permissions, která pokrývá oblasti vyžadující soukromá data uživatele a operace, které by mohly mít na tato data dopad nebo by mohly ovlivnit ostatní aplikace. Tato oprávnění musí uživatel po instalaci aplikace explicitně potvrdit.

[12] Andorid distribution dashboard https://developer.android.com/about/dashboards/

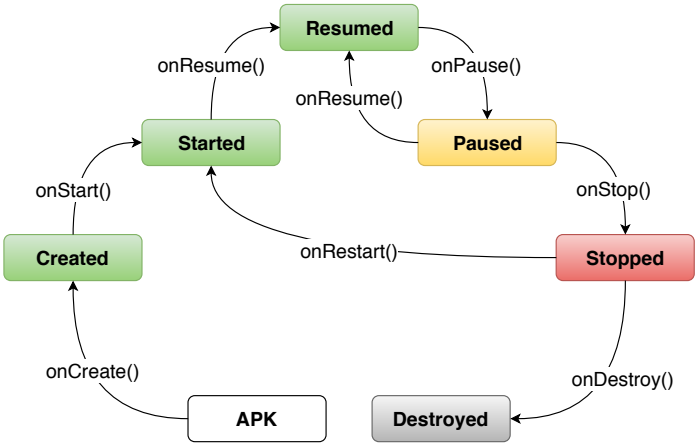


Obrázek 2.9: Srovnání požadavků na kritická oprávnění před verzí 6.0 (vlevo) a po ní.

V nižších verzích než 6.0 (API 23) byl uživateli při prvním startu aplikace předložen seznam kritických oprávnění a ten je mohl hromadně povolit či zamítnout. Od verze 6.0 doznal tento přístup ve snaze vyhnout se dlouhým seznamům oprávnění, o jejichž významu nemá uživatel v době instalace tušení, jistých změn. Aktuální systém je navržený tak, aby měl uživatel co nejlepší přehled o tom, jaké oprávnění uděluje a z jakého důvodu. Systém zároveň počítá i s variantou, že některá oprávnění nejsou pro běh aplikace nezbytná. Uživatel tak může aplikaci používat i přesto, že některé oprávnění zamítl. Toho je docíleno tak, že Android nyní umožňuje zasílat uživateli požadavky na oprávnění za běhu aplikace a poskytuje rozhraní pro reakci na rozhodnutí uživatele. Pokud vývojář vytváří například aplikaci na úpravu obrázků, jejíž součástí je možnost pořídit fotografii, kterou lze následně upravovat, měl by zaslat požadavek na oprávnění k užití kamery až ve chvíli, kdy se uživatel rozhodne spustit funkci snímání. Tím je zajištěno, že uživatel je dotazován až za situace, kdy má pro něj požadavek smysl a zároveň poskytuje možnost odmítnout a přesto dál používat aplikaci k upravě existujících obrázků.

Oprávnění jsou navíc dělena do skupin za účelem odstínit uživatele od příliš technických detailů. Například pokud aplikace vyžaduje povolení ke čtení kontaktů, Android vyzve uživatele obecně k udělení přístupu ke kontaktům. Pokud později aplikace žádá povolení k zápisu do kontaktů, je přiděleno automaticky, neboť čtení a zápis jsou ve stejné skupině. Dokumentace nicméně uvádí, že by vývojáři na tento efekt neměli spoléhat, neboť členství oprávnění k určitým skupinám se muže napříč verzemi lišit.

Systém oprávnění je také využit pro vyhodnocení podporovaných zařízení při distribuci. Pokud například aplikace vyžaduje oprávnění na Bluetooth, v distribučním nástroji Google Play se nezobrazí na zařízeních, která technologii Bluetooth nepodporují. Pokud se jedná o oprávnění, které není nezbytné pro běh aplikace, je nutné to uvést u příslušného oprávnění v souboru manifestu.



Obrázek 2.10: Životní cyklus aktivity

2.3.4 Základní elementy aplikace pro OS Android

Aktivity a Fragmenty

Aktivita reprezentuje jednu „obrazovku“ s uživatelským rozhraním. Zdrojový kód aktivity sestává ze dvou části. První částí je třída aktivity, která popisuje její logickou část. Druhou částí je pak XML soubor definující vzhled této obrazovky, respektive prvky uživatelského rozhraní a celkový layout (návrh). Aplikace se obvykle skládá z více aktivit, které lze za běhu vytvářet a přepínat mezi sebou. Každá aktivita má svůj životní cyklus (viz obrázek 2.10).

Pokud se jedná o komplexnější aktivitu, která se skládá z více panelů nebo složitějších ovládacích prvků, je možné využít takzvaných fragmentů. Jedná se o samostatné bloky podobné aktivitám s vlastním životním cyklem. Fragmenty mohou být přiřazeny k rodičovské aktivitě, a tak s ní nebo s ostatními fragmenty komunikovat. Výhodou fragmentů je, že je možné tutéž třídu fragmentu přiřadit k různým aktivitám, nebo k jedné aktivitě přiřadit více instancí téhož fragmentu. Myšlenka fragmentů tak značně napomáhá opakovatelnosti kódu a konzistenci uživatelského rozhraní.

Služby

O Službě v souvislosti s OS Android mluvíme v případě dlouho trvajících procesů obvykle běžících na pozadí. Jde o nástroj pomocí něhož mohou aplikace spustit proces nezávisle na tom, zda je daná aplikace na popředí. Z pravidla tedy tyto procesy běží i v době, kdy se zařízením uživatel nepracuje. Zároveň poskytují rozhraní které umožňuje navázat se službou komunikaci z jiného procesu. Služby mohou sloužit například k síťové komunikaci, provádět vstupní či výstupní souborové operace nebo například přehrávat hudbu.

Služby lze rozdělit do tří kategorií na základě způsobu interakce. První skupinou jsou služby běžící zcela na pozadí, o jejichž činnosti uživatel není informován, nicméně mohou zasílat upozornění. Vhodným případem může být proces údržby vlastních dat aplikace či stahování aktuálních dat za sítě.

Další skupinou jsou služby na popředí. O běhu těchto služeb musí být uživatel informován v oblasti upozornění. Jedná se například o již zmíněné přehrávání hudby, či stahování větších objemů dat, jako jsou balíčky kartografických map pro aplikaci navigace. V takovém

případě je po dobu běhu zobrazena v oblasti upozornění položka informující o běhu služby a ve většině případů též o jejím průběhu.

Poslední skupinou jsou navazované služby. Navazované služby poskytují rozhraní typu klient-server, které umožňuje komponentám nebo aplikacím zasílat na službu dotazy, přijímat odpovědi a to napříč procesy. Jedná se tedy o nástroj pro „meziprocesovou“ komunikaci (IPC).[5]

Vzhledem k tomu, že běh těchto služeb konzumuje omezené zdroje zařízení, jako je operační paměť nebo čas procesoru, muže mít jejich přílišné užívání negativní dopad na UX (user experience), zvláště u operací náročných na výkon, jako je hraní her či přehrávání videa. Proto byly zavedeny od verze 8.0 (API 26) určitá omezení pro služby na pozadí a jejich místo by ve většině případů měl nahradit plánovač úloh.[13]

Přijímače všesměrového vysílání

Tzv. „broadcast receivers“ jsou komponenty které umožňují reagovat na události, které zasílá systém všem aplikacím. Jedná se například o informaci o nízkém stavu baterie, nebo o tom že display byl zhasnut. Všesměrová vysílání mohou používat také aplikace. Toho lze využít například když chce aplikace informovat ostatní aplikace o tom, že data jsou stažena a nyní je mohou ostatní procesy nebo aplikace využívat. Tato všesměrová hlášení sama o sobě nijak neinteragují s uživatelským rozhraním, nicméně aplikace, která vysílání přijme může uživatele o události informovat. Obvykle je všesměrové vysílání bránou ke komunikaci s jinými aplikacemi. Implementace tohoto přijímače je provedena jako podtřída třídy BoradcastReciever a vysílání je doručeno jako Intent objekt, což je třída poskytující prostředky pro popis určitého záměru, jako je například přepnutí aktivity.

2.3.5 Práce s kamerou

OS Android poskytuje několik způsobů, jak pořídit snímek prostřednictvím kamery. Nejjednodušší variantou je využít třídu `Intent` s akcí `MediaStore.ACTION_IMAGE_CAPTURE`. Po zavolání metody `startActivityForResult()` s touto třídou v parametru je systém informován, že má spustit aktivitu pro snímání fotografie. V praxi je na základě tohoto volání spuštěna výchozí aplikace fotoaparátu.

Poté, co je snímek pořízen, zavolá aktivita fotoaparátu metodu `onActivityResult()`, které v parametru předá vyfocený snímek ve formátu BMP. Proto je nutné tuto metodu přepsat kódem který zajistí zpracování výsledku.

Mnohem sofistikovanějším nástrojem je Camera2 API, které je k dispozici od verze Amdroid API 21 a poskytuje sadu tříd pro komplexní práci s kamerou. Velmi podstatnou částí je třída `CameraManager`, která poskytuje přístup k informacím o dostupných snímačích (typicky přední a zadní kamera) a jejich vlastnostech (např. podpora automatického ostření, rozlišení...) a také umožňuje vytvořit spojení s vybraným zařízením. Jakmile je spojení navázáno, díky třídě `CameraDevice.StateCallback` je možné na tuto událost reagovat metodou `onOpened()`, která předává v argumentu instanci třídy `CameraDevice`. Pro tuto instanci je již možné vytvořit pomocí metody `createCaptureRequest()` Builder. Ten slouží k přiřazení cíle - tedy instatnce třídy `Surface`, která je připravena k zobrazení. Nyní je možné zahájit snímání pomocí metody `createCaptureSession()`. Oproti předchozí metodě je třeba k dosažení výsledku následovat několik kroků, které zahrnují znatelně větší

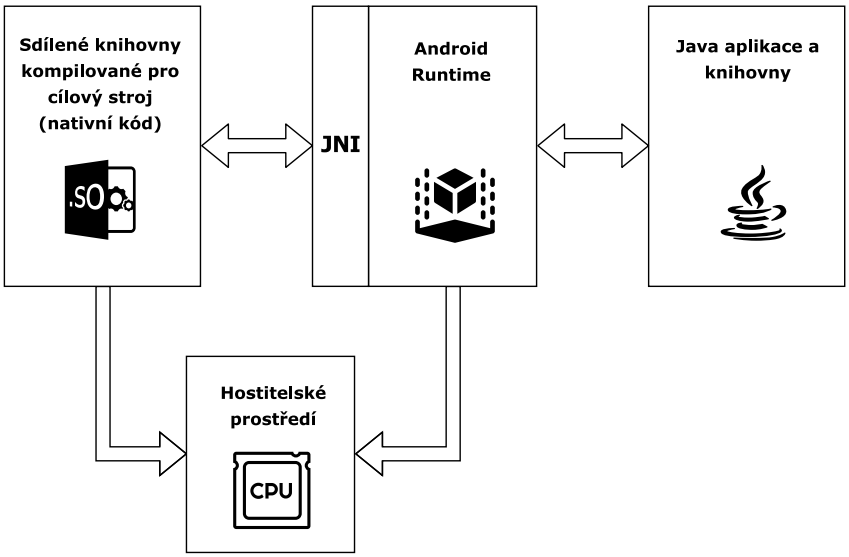
[13] Omezení služeb na pozadí https://developer.android.com/about/versions/oreo/background

množství kódu, nicméně právě díky tomu je možné například zobrazit náhled či obraz dále zpracovávat.

2.3.6 Použití C++ knihoven v prostředí Android

Operační systém Android používá od verze 5.0 Lollipop k běhu svých aplikací tzv. Android Runtime (ART), který se dá považovat za obdobu Java Virtual Machine (JVM) a poskytuje vhodné prostředí pro většinu aplikací napsaných v Jazyce Java či Kotlin. Existují však nejméně dvě situace, kdy je užitečné toto prostředí obejít. První z nich nastává ve chvíli, kdy je vyžadována co nejnižší latence a co nejvyšší výkon (například v případě videoher). Druhá pak nastává, pokud vývojář chce použít již existující kód (např. knihovnu), který je ovšem napsaný v jiném jazyce.

Pro tyto případy je k dispozici tzv. Native Development Kit(NDK). Jedná se o balíček nástrojů sloužících k použití C nebo C++ kódu na platformě Android. Klíčovou roli v této oblasti hraje speciální rozhraní známé z platformy Java - Java Native Interface (JNI), které je třeba pro používání C/C++ kódu v prostředí Android implementovat. Poté NDK umožní kompilovat kód do nativních knihoven a přibalit je k výsledné aplikaci.[6]



Obrázek 2.11: Diagram komunikace pomoci JNI

Kapitola 3

Zhodnocení současného stavu a plán práce

Tato kapitola v první části srovnává dostupné nástroje v oblasti bezkontaktního ovládání počítačových zařízení, přináší jejich hodnocení na základě několika relevantních kriterií a diskutuje prostředky. Druhá část se zabývá výběrem vhodné technologie pro implementaci bezdotykového uživatelského rozhraní. Poslední část na základě předchozího hodnocení detailně specifikuje cíle práce.

3.1 Srovnání existujících bezdotykových uživatelských rozhraní

Vzhledem k tomu, že většinu ovládacích prvků poskytují současné operační systémy prostřednictvím obrazovek, je jedním z hlavních úkolů uživatelského rozhraní zajistit efektivní způsob, jak předat zařízení informaci o tom, který ovládací prvek chce uživatel použít. Nejpřímočařejším způsobem je použití dotykové obrazovky, které je rozšířené zejména u přenosných počítačů nebo chytrých telefonů. U desktopů je naprostým standardem počítačová myš a hojně rozšířené jsou i další indikátory polohy jako touchpad nebo trackball. Všechna tato zařízení ovšem nelze použít bez pomoci rukou. Tento nelehký úkol řeší mimo jiné čtveřice aplikací popsaná v sekci 2.1. Nyní se pokusím zhodnotit, do jaké míry se jim to daří.

Tobii PCEye Plus

Prvním zástupcem je Tobii PCEye Plus. V rámci této čtveřice se jedná o nejsofistikovanější a také nejnákladnější řešení. Jako jediný využívá k navádění kurzoru skutečný směr pohledu a nikoliv natočení obličeje. To poskytuje větší komfort, rychlost i přesnost. Nevýhodou tohoto přístupu jsou nadstandardní hardwarové požadavky a z nich plynoucí i vyšší náklady. Systém navíc disponuje praktickým a přívětivým grafickým uživatelským rozhraním, které umožňuje provádět efektivně akce jako jsou klik, dvojklik, nebo rolování. Velkou předností je i podpora zadávání textu pomocí speciální klávesnice nebo pomocí rozpoznávání řeči. Systém je určený pro platformu MS Windows.

Klady	Zápory
rychlost a přesnost	cena 63 400,00 Kč[14]
propracované GUI	(verze mini 45 600,00 Kč)
ergonomie	nezbytný externí hardware
rozpoznávání řeči	

Tabulka 3.1: Hodnocení Tobii PCEye Plus

SmartNav

SmartNav je dalším zástupcem, který vyžaduje ke svému fungování externí zařízení. V tomto případě se jedná o infra kameru a reflexní bod umístěný buď na těle, nebo na nějaké části oblečení. Tento přístup má pozitivní vliv na přesnost oproti použití běžné kamery a také umožňuje k indikaci polohy použít libovolnou pohyblivou část těla. Systém poskytuje jen velmi omezené softwarové prostředky pro efektivní používání a provádění akcí jako je klik nebo rolování. SmartNav jde spíše cestou hardwarových doplňků a tak zmíněný problém řeší nabídkou pedálů, které slouží jako tlačítka myši. Tyto pedály nejsou součástí balení.

Klady	Zápory
přesnost	nezbytný externí hardware
možnost doplňků	nepohodlné ovládání
	zadávání textu pouze pomocí softwarové klávesnice

Tabulka 3.2: Hodnocení SmartNav

SmyleMouse

SmyleMouse je aplikací, která si vystačí s běžnou webkamerou, kterou mimo navádění kurzoru na základě polohy obličeje využívá i k rozpoznání úsměvu, což lze využít jako příkaz. Tato myšlenka je hlavním přínosem tohoto řešení do světa bezkontaktních uživatelských rozhraní, neboť většina ostatních systému využívá jako indikátor kliknutí časovač nečinnosti kurzoru (po nastaveném časovém úseku, kdy se kurzor nehýbe, systém provede klik na pozici kurzoru), což často vede k nevyžádaným kliknutím. SmyleMouse umožňuje úsměv využít například k aktivaci pohybu kurzoru nebo k události kliknutí. Technologie bohužel není příliš přesná a obsažení celé plochy monitoru vyžaduje poměrně silné pohyby hlavou. Sytém je určen pro platformu Windows.

[14] Ceník českého dodavatele kompenzačních pomůcek http://www.spektra.eu/soubory/ceniky/MC.CZK.pdf?2017-10-19

Klady	Zápory
plně softwarové řešení	zadávání textu pouze pomocí softwarové klávesnice
událost (klik) na gesto	nepohodlné ovládání
	nepřesný

Tabulka 3.3: Hodnocení Smyle Mouse

EVA facial mouse

EVA Facial Mouse je jediným zástupcem pro platformu Android. Uživatelské rozhraní, které poskytuje, je velice podobné předchozímu řešení. Na základě polohy obličeje pohybuje kurzorem po obrazovce a ve chvíli, kdy se kurzor zastaví na určitou dobu, provede kliknutí. Jedná se o jediné nekomerční řešení.

Klady	Zápory
plně softwarové řešení	zadávání textu pouze pomocí softwarové klávesnice
zdarma - otevřený kód	nepohodlné ovládání
	nepřesný

Tabulka 3.4: Hodnocení EVA facial mouse

3.2 Specifikace výsledné aplikace

Z výše uvedeného srovnání vyplývá, že uživatelé s postižením nebo absencí rukou musí při záměru interagovat s elektronickým zařízením volit mezi velmi nákladným, nebo velmi nepohodlným a neefektivním řešením. V oblasti mobilních zařízení (vezmeme-li navíc v úvahu, že 90 % mobilních zařízení na trhu funguje pod OS Android (viz graf 2.7) je pak situace ještě horší. Z aktuální nabídky lze také odhadnout, že vytvořit komplexní bezdotykové uživatelské rozhraní pro současná zařízení je s dostupnými technologiemi nelehký úkol.

S ohledem na zadání práce a z výše uvedených důvodů jsem se rozhodl vytvořit uživatelské rozhraní, které umožní ovládat zařízení s operačním systémem Android za pomocí pohybů hlavy a úst a které bude splňovat následující kriteria:

- Systém si vystačí s běžným hardwarovým vybavením současných zařízení
- Systém vyžaduje co nejmenší pohyb uživatele
- Systém umožňuje efektivní zadávání textů
- Sytém poskytuje dostatečnou spolehlivost rozpoznávání gest
- Sytém poskytuje co nejvyšší pohodlí používání

Vzhledem k důrazu na efektivitu použití výsledné aplikace jsem se rozhodl nevytvářet komplexní uživatelské rozhraní nahrazující dotykovou obrazovku, ale optimalizované řešení pro specifický účel. Volba padla na internetový prohlížeč, jelikož je tento nástroj dostatečně univerzální a využívaný, avšak nemá tak komplexní požadavky na uživatelské rozhraní.

Cílem této práce je tedy vytvořit webový prohlížeč pro mobilní platformu Android, který lze efektivně a pohodlně využívat bez použití rukou.

Kapitola 4

Vývoj obličejem ovládaného webového prohlížeče

Následující kapitola rozebírá jednotlivé kroky, které provázely vývoj aplikace, která má sloužit jako internetový prohlížeč, jehož použití je možné bez užití rukou a poskytne tak způsob lidem s pohybovým postižením interaktivně využívat webových služeb na zařízeních s operačním systémem Android.

4.1 Výběr vhodné technologie pro rozpoznávání obličeje

Sekce 2.2 popisuje dostupné technologie, které jsou schopné z videa extrahovat informace o obličeji. Pro účely uživatelského rozhraní je třeba získat dostatečné množství informaci s dostatečnou spolehlivostí a v dostatečné rychlosti. Tato sekce popisuje, jak tyto požadavky splňují vybrané technologie, které jsou dostupné pro OS Android.

Google Mobile Vision Největší výhoda Google Mobile Vision je, že se jedná o API vytvořené přímo „na tělo“ mobilním zařízením a stejně jako OS Android je vyvíjené pod hlavičkou společnosti Google. Z toho plyne velice snadné použití, výborná optimalizace a tím pádem i rychlost. API funguje velmi spolehlivě v úloze detekce obličeje v obraze a to i z profilu. Poskytuje také informace o orientaci obličeje a o tom, zda se tvář usmívá či nikoliv. Z nějakého důvodu jsou ovšem k dispozici pouze informace o orientaci obličeje v osách Y a Z. To je velká škoda a je možné, že v následujících aktualizacích se této funkce dočkáme. Nicméně v současné době API bohužel neposkytuje dostatek informací k vytvoření efektivního uživatelského rozhraní a situaci nezlepší ani 8 landmarků, jejichž detekce není bohužel příliš přesná ani spolehlivá. API je zkrátka navržené k jinému účelu, ale v budoucnu by mohlo poskytnout velmi zajímavou alternativu pro tento úkol.

OpenFace OpenFace je špičkovým nástrojem v oblasti rozpoznávání obličeje. Z vybraných možností poskytuje největší množství informací. Díky informacím o akčních jednotkách obličeje, náklonu a orientaci hlavy ve všech osách, detekcí 68 landmarků, a dokonce i odhadem směru pohledu takřka v reálném čase představuje nejvšestranější nástroj. Jeho použití bohužel vyloučila náročnost implementace této technologie v prostředí Android. Ta je způsobena mimo jiné tím, že OpenFace využívá knihovny třetích stran. Jelikož se jedná o OpenSource projekt, není tato možnost vyloučena a v komunitě vývojářů se o ní v současné

době vedou diskuze[1]. Nepodařilo se však najít použitelný výsledek a samotná implementace daleko přesahuje rozsah této práce. Kdyby byla ovšem cílem práce jiná platforma, jistě by padla volba na OpenFace.

Dlib Dlib je dalším projektem s otevřeným kódem, který poskytuje funkce umožňující s velkou přesností a rychlostí detekovat až 68 landmarků. Stejně jako v případě OpenFace se jedná o software s otevřeným kódem a navíc je navržený s důrazem na platformní nezávislost a vysokou přenositelnost kódu. Tyto atributy daly vzniknout projektu dlib-android[2] od taiwanského vývojáře TzuTa Lina. Další z jeho repositářů navíc poskytuje ukázkovou aplikaci, na které proběhlo testování technologie a po jejím úspěšném zvolení byly některé třídy z této aplikace použity jako základ pro vyvíjený prohlížeč.

4.2 Rozpoznávání obličeje

Prvním úkolem bylo získat tok videa z kamery a na jednotlivé snímky aplikovat detektor landmarků. Ke komunikaci se zařízením kamery bylo využito Camera2 API. Pomocí něj je bezprostředně po startu aplikace provedeno nastavení a inicializace zařízení kamery. Tomuto API je věnována sekce 2.3.5, proto tuto část nebudu více rozvádět.

O zpracování výsledných jednotlivých snímků se stará třída `onGetImageListener`, jež implementuje rozhraní `OnImageAvailableListener`. Prvním jeho úkolem je převést obrazová data do vhodného formátu pro detektor. Standardem pro náhled videa v prostředí Android je, zejména z důvodu komprese dat, formát YUV420. Pro aplikování detektoru je ovšem vhodným formátem RGB8888. Díky všestrannosti knihovny DLib je však možné ji mimo samotné detekce obličeje použít i k tomuto účelu.

Po převedení obrazových dat (bitmapy) do příslušného formátu systém provede samotnou detekci. Vzhledem k tomu, že detektor, který poskytuje knihovna Dlib, je schopen detekovat i více než jednu tvář v jednom snímku, je výsledkem `List<VisionDetRet>`, přičemž třída `VisionDetRet` reprezentuje jeden detekovaný obličej. Pro účely této aplikace je však tato funkcionalita irelevantní, a tak použitý algoritmus bere v potaz pouze první detekovaný obličej. Jeho nejpodstatnější složkou je `ArrayList<Point>`. Jedná se o souhrn souřadnic všech detekovaných landmarků.

Posledním úkolem třídy `onGetImageListener` je vytvořit náhledový obrázek, který slouží jako uživatelský náhled. Obrázek sestává ze získaného snímku kamery a vrstvy, která vykresluje jednotlivé landmarky na svých pozicích v podobě jejich pořadových čísel. Tyto jsou v rámečku, který definuje oblast, na které detektor vyhodnotil výskyt obličeje.

Tento obrázek je společně s daty týkající se landmarků prostřednictvím metody `onLandmarksAvailible` předán hlavní a jediné aktivitě této aplikace.

4.3 Rozpoznávání gest

4.3.1 Definování gest

Na první pohled převrácená posloupnost kroků vývoje, kdy předchází výběr technologie rozpoznávání specifikaci toho, co bude rozpoznávat, není zvolena náhodou. Během vyhledávání a testování rozpoznávacích algoritmů bylo zjištěno, že různé algoritmy reagují na

[1] OpenFace Android integration issue https://github.com/TadasBaltrusaitis/OpenFace/issues/9
[2] Dlib-android https://github.com/tzutalin/dlib-android

stejné podněty s různou citlivostí a přesností. Strategie výběru technologie byla zvolena s cílem získat řešení, které poskytne co nejvíce informací za nejkratší čas s uspokojivou přesností. Až po výběru knihovny a jejím testování v praxi bylo přistoupeno k definovaní gest s důrazem na pohodlí uživatele a spolehlivost v kombinaci právě s konkrétní zvolenou rozpoznávací technologií.

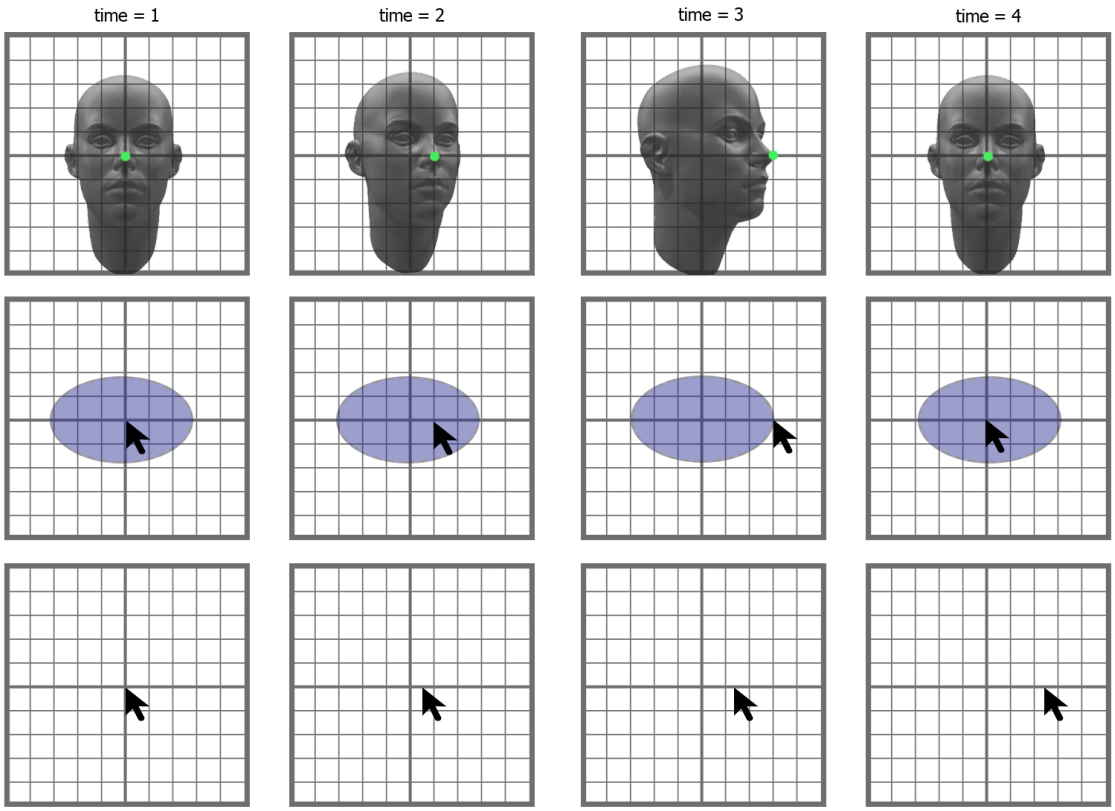
Během testování bylo například vyloučeno původně zamýšlené gesto zdviženého obočí. Bylo totiž zjištěno, že u osob s brýlemi nebo se světlým obočím algoritmus vykazuje neuspokojivou spolehlivost. Stejně tak bylo nutno upustit od záměru použít „špulení rtů“, případně roztažení koutů úst od sebe pro zoom-in, respektive zoom-out v okně prohlížeče.

Indikátor polohy

Nejkomplexnějším ovládacím mechanismem, který bylo třeba vytvořit, byl indikátor polohy. Tedy systém který umožní uživateli pohybovat kurzorem. Všechna testovaná řešení, která byla popsána v kapitole 2.1 využívající pohybů hlavy, vycházela ze systému podobnému počítačové myši. Tedy každý pohyb sledované části těla (nejčastěji nosu) promítnutý do dvourozměrné souřadnicové sítě vyvolal odpovídající, případně mírně upravený pohyb kurzoru na zobrazovací jednotce. Jedná se o systém poskytující možnost velmi rychlé změny pozice kurzoru. Problém ovšem představuje skutečnost, že jemnými pohyby hlavy, které jsou pohodlné i po delší době používání, uživatel obsáhne pouze malou část plochy obrazovky. Tato část má navíc, na rozdíl od obrazovky, spíše oválný tvar. Při testování ostatních řešení se několikrát stalo, že ve snaze dosáhnout do rohu obrazovky bylo nutno mířit nosem do takové polohy, že se obrazovka dostala mimo přirozený zorný úhel. Řešením může být zvýšení citlivosti, což má ovšem za následek zvýšení počtu nevyžádaných pohybů kurzorů a také se stane mnohem obtížnější trefit ovládací prvky o menších rozměrech. Za těchto podmínek se ovšem skutečně nedá mluvit o pohodlném uživatelském rozhraní.

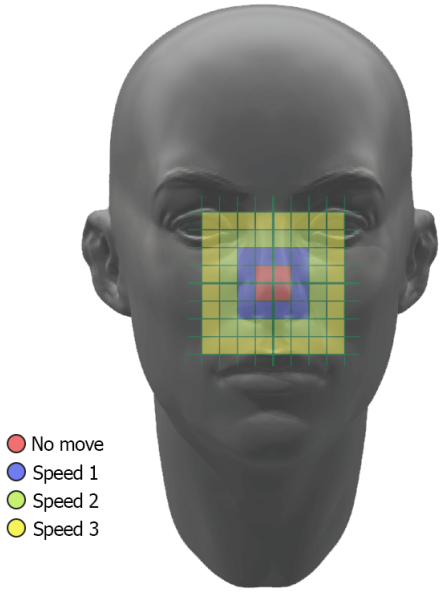
Z těchto důvodů jsem se rozhodl použít systém, který se dá přirovnat spíše než k myši k track pointu nebo joysticku. Systém sleduje natočení obličeje v osách X a Y a na základě těchto údajů vypočítá směr a rychlost pohybu kurzoru. V případě, že je obličej v přirozené poloze, jsou tyto parametry na nule. Tento stav znázorňuje obrázek 4.2.

V případě, že se obličej natočí například mírně vpravo, systém začne pomalu pohybovat kurzorem směrem doprava. Pokud se obličej natočí více, pohyb se zrychlí. Když poté uživatel náhle vrátí hlavu do původní přirozené polohy, kurzor se zastaví a zůstane na své pozici. Tento scénář na obou srovnávaných systémech ilustruje obrázek 4.1. První řada na obrázku představuje pohyb obličeje v čase, druhá řada simuluje chování běžné pro většinu existujících řešení a třetí řada pak prezentuje chování vyvíjené aplikace. Modře vyznačená oblast zobrazuje přibližný tvar plochy dostupné v rámci „komfortní zóny“ pohybu hlavy.



Obrázek 4.1: Simulace chování různých metod navádění kurzoru

Systém založený na detekci natočení obličeje je oproti absolutnímu určování pozice pomalejší, nicméně poskytuje větší pohodlí rovnoměrně na celé ploše obrazovky a také není tolik náchylný k chybám vznikajícím případným posunem snímacího zařízení oproti obličeji (v případě absolutního určování polohy musí být zařízení po celou dobu používání ve stejné poloze). Za výhodu tohoto přístupu lze také považovat, že nevyžaduje kalibraci.



Obrázek 4.2: Ilustrace závislosti rychlosti pohybu kurzoru na poloze hlavy

Gesto pro kliknutí

Je zřejmé, že samotná indikace polohy bez provedení akce v daném místě by byla bezvýznamná. Současné systémy se s problémem vypořádávají různými způsoby. Častým řešením je tzv. dwell-clicker. Jedná se o způsob, kdy je vybraný příkaz (nejčastěji kliknutí) proveden tehdy, když je kurzor po určitou dobu v nečinnosti. Tento systém sice plní svůj účel, nicméně vede ke značnému diskomfortu, a to zejména v situaci, kdy je jako indikátor polohy použita hlava uživatele. Pokud totiž uživatel právě nechce provést klik, má dvě možnosti. První možností je navigovat kurzor do místa, kde klik nemá žádný efekt, a nechat jej kliknout. Pak si ovšem musí dát pozor, aby neudělal příliš velký pohyb hlavou, neboť by pak došlo k „odparkování kurozru“ a proces by bylo nutné opakovat. Práh rychlosti pohybu nutné k „odparkování kurozru“ bývá sice nastavitelný, nicméně příliš vysoká hodnota na druhé straně vede k nutnosti nepříjemně prudkých pohybů v situaci, kdy je „odparkování kurozru“ žádoucí. Druhou možností je uvést dwell-clicker do neaktivního režimu. Komplikovanost přechodu do neaktivního režimu a zpět se liší, ale obecně jsou to úkony, které uživatele zdržují. Použití externího hardware k tomuto účelu je automaticky vyloučeno vzhledem ke stanoveným cílům a tak zbývá využití landmarků, které máme k dispozici z rozpoznávacích algoritmů.

Cílem bylo nalézt gesto (pohyb obličejové akční jednotky), které by bylo pro uživatele přirozené, neboť jej bude používat velmi často, ale zároveň jednoznačně rozpoznatelné s ohledem na přesnost detekce landmarků. Po krátkém experimentování byla vybrána oblast úst, jelikož se jedná o nejpohyblivější část obličeje a mimické svaly v této oblasti jsou nejlépe trénované. S ohledem na toleranci nepřesnosti algoritmu byl nakonec zvolen úsměv s podmínkou oddělení rtů. Tato podmínka byla přidána až dodatečně, neboť bez ní detekce gesta nevykazovala dostatečnou spolehlivost. To představuje určitý zásah do pohodlí uživatele, a to především proto, že aplikace díky tomu provádí nevyžádaná kliknutí, pokud uživatel během používání aplikace mluví. Obecně se dá ale říci, že pokud se uživatel zdrží

hovoru, počet nevyžádaných kliknutí tato úprava sníží a také díky tomu nemusí být úsměv „tolik široký“.

Optimalizační gesta

Ve chvíli, kdy aplikace umí určit polohu na obrazovce a na tomto místě lze provést kliknutí, je možné realizovat takřka jakýkoliv příkaz jednoduše přidáním dalšího grafického ovládacího prvku na obrazovku. Je třeba si ale uvědomit, že s dostupnými prostředky bude navádění kurzoru buď nepřesné nebo nebo nedostatečně rychlé. Dalším faktorem je, že zabírat místo na obrazovce také nelze do nekonečna a cílem je zachovat co největší efektivní plochu pro samotný webový prohlížeč.

Existuje tedy celá řada motivací k maximálnímu využití potenciálu gest na místo softwarových tlačítek. Kriteriem pro nasazení takového gesta bylo, aby jeho použití bylo pohodlnější a rychlejší, než navigování kurzoru na softwarové tlačítko. Taková gesta nakonec byla nalezena právě dvě. Jedná se o náklon celé hlavy doleva a stejný náklon doprava. Tato dvě gesta sice nejsou tak pohodlná, ale pro „nepříliš časté“ příkazy jsou jistou optimalizací.

Na náklon vlevo byl navázán příkaz prohlížeče „kork zpět“ a na náklon vpravo reaguje aplikace aktivací hlasového rozpoznávání, po jehož provedení automaticky začne zobrazovat výsledky pro zvolený výraz.

4.3.2 Detekce gest

Tato část popisuje proces detekce gest z hlediska algoritmů, které detekci provádí, popisuje řešené problémy a blíže specifikuje charakteristické vlastnosti a optimalizace přesnosti těchto algoritmů.

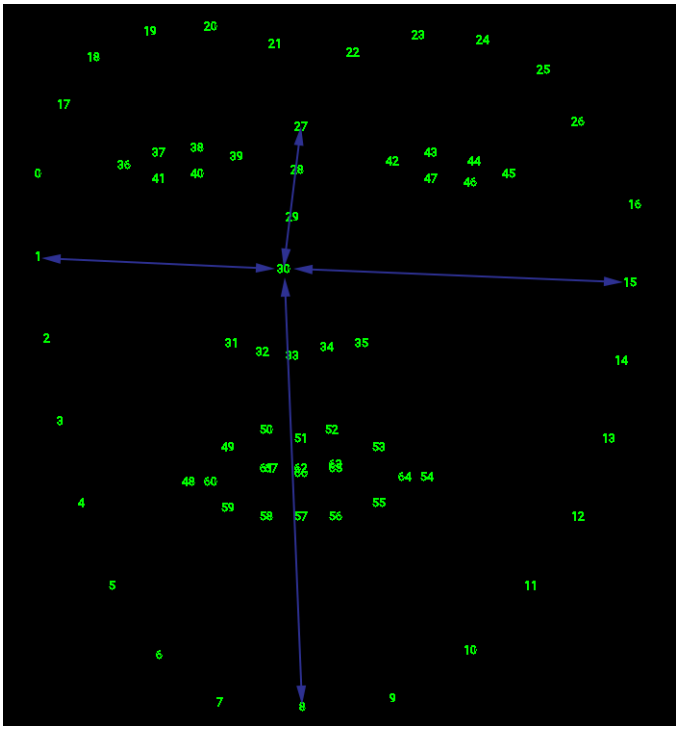
Výpočet směru a rychlosti kurzoru

Jak již bylo uvedeno v předchozí sekci, aplikace využívá k určení směru a rychlosti pohybu kurzoru vychýlení směru pohledu oproti přímému pohledu vpřed v osách X a Y. Vzhledem k tomu, že použitý detektor z knihovny Dlib bohužel neposkytuje informace o orientaci obličeje ani v jedné ose, je třeba tyto parametry odhadnout na základě pozic landmarků. K výpočtu byla využita pětice landmarku, jejichž relativní vzdálenost se při natáčení obličeje nejvíce měnila.

Pro osu X se jedná se o bod 30, který je umístěný na špičce nosu, bod 1, který reprezentuje levý okraj rozpoznaného obličeje a bod 15, který reprezentuje pravý okraj obličeje.

Základní myšlenka vychází ze symetrie obličeje. Výpočet tedy probíhá tak, že se nejprve spočítá vzdálenost bodu špičky nosu od bodu levého okraje. Stejným způsobem je vypočítána vzdálenost špičky nosu od pravého okraje a na závěr jsou tyto dvě hodnoty od sebe odečteny. Pokud jsou obě vzdálenosti stejné, znamená to, že obličej jen namířen přímo proti kameře a rozdíl obou hodnot je tedy roven nule. Je zřejmé, že záporná hodnota této proměnné indikuje, že je obličej namířen doleva, a kladná znamená pravý opak.

Pro osu Y je postup velmi podobný, nicméně zde už nelze využít osové souměrnosti. Je ale možné vypozorovat, že při přirozené poloze obličeje vzdálenost bodu 8, který reprezentuje nejnižší bod brady od špičky nosu, je zhruba dvojnásobkem vzdálenosti špičky nosu od jeho kořenu. Pokud tedy normalizujeme tuto vzdálenost násobením dvěma, získáme stejný vztah jako pro osu X. Obě hodnoty jsou na závěr násobeny koeficientem rychlosti.



Obrázek 4.3: Ilustrace měřených hodnot za účelem pohybu kurzoru

Pro zamezení nevyžádaných pohybů kurzoru a zvýšení přesnosti je ovšem nezbytné definovat práh, od kterého pohyb obličeje začne mít vliv na pohyb kurzoru. Jinými slovy tento práh definuje rozsah pohybu obličeje, který nebude vyhodnocen jako impuls k pohybu kurzoru a také poskytne low-pass filtr pro nepřesnosti detektoru landmarků. Je nasnadě vyřešit tuto situaci podmínkou na minimální absolutní hodnotu odchylky. Toto řešení ovšem vytváří postranní efekt, který způsobuje, že jakmile je práh překročen, rychlost skokově naroste o velikost prahu. Proto je po překročení tohoto koeficientu jeho hodnotu od výsledku nutno odečíst. Pohyb však může být i v záporném směru. Proto jsou součástí výpočtu proměnné `Hpositive` a `Vpositive`, které nabývají hodnot 1 nebo -1 na základě toho, ve kterém směru byl práh překročen, a slouží k tomu, aby byl koeficient v případě pohybu v kladném směru odečten a v záporném směru přičten.

Nyní máme k dispozici hodnoty `deltaH` a `deltaV`, jejichž hodnoty určují rychlost a jejich znaménka směr. Následuje zjištění aktuálních souřadnic kurzoru a přičtením `deltaH` a `deltaV` jsou vypočteny souřadnice nové.

Posun kurzoru a podpora rolování

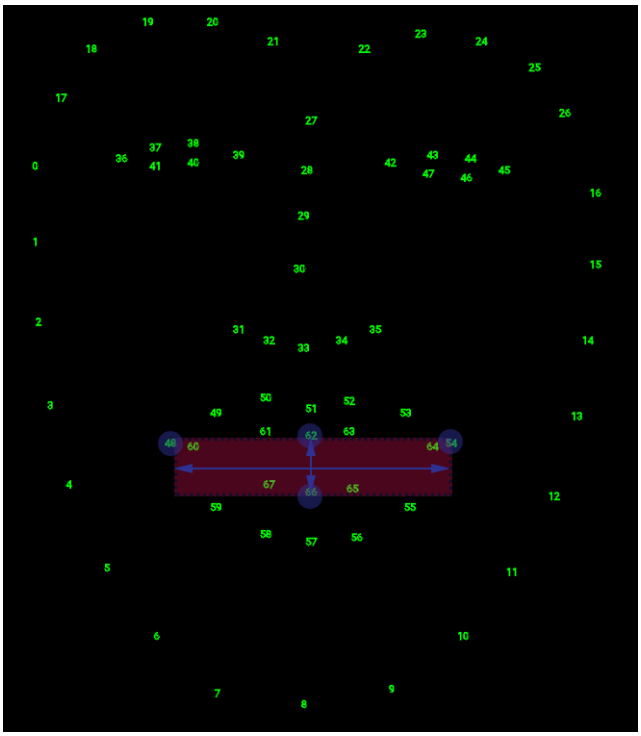
Díky vypočítaným novým souřadnicím můžeme nyní posunout kurzor na novou pozici. Vzhledem k tomu, že málokdy se však celý obsah webové stránky vejde na obrazovku, je nezbytné vyřešit podporu rolování. Problém je řešen následujícím způsobem.

Systém využívá konstanty `H_SCROLL_THRESHOLD` a `V_SCROLL_THRESHOLD` pro definování tzv. rolovacího rámečku. `H_SCROLL_THRESHOLD` specifikuje výšku vrchní a spodní části rolovacího rámečku a `V_SCROLL_THRESHOLD` pak levý a pravý okraj. Pokud se vypočtené nové souřadnice kurzoru nachází mimo plochu rolovacího rámečku, je kurzor na tyto souřadnice přemístěn za pomocí animace. Pokud však leží nové souřadnice na ploše rolovacího rámečku,

systém nejprve zjišťuje, zda je možné v daném směru někam rolovat, a pokud ano, kurzor zůstává na místě a stránka je rolována v požadovaném směru. Až ve chvíli, kdy není kam rolovat, umožní systém kurzoru „vstoupit“ na plochu rolovacího rámečku. Tento přístup se snaží maximalizovat rychlost používání aplikace v případě webových stránek přesahujících rozsah obrazovky. Pokud uživatel potřebuje informace které se nachází pod úrovní zobrazené části, bez rolovacího rámečku by musel přejet kurzorem celou obrazovku a ve chvíli, kdy by chtěl rolovat zpět nahoru, by musel čekat, než se kurzor dostane zpět k horní hraně obrazovky.

Detekce úsměvu

Jak už bylo naznačeno výše, k příkazu kliknutí je určen úsměv s podmínkou oddělení rtů. Pokud se vyjádřím více technicky, pak se jedná o detekci minimální plochy, která je definována koutky úst a spodním okrajem horního rtu a horním okrajem spodního rtu. K tomuto účelu byly vybrány odpovídající landmarky 48, 54, 62. Při zavřených ústech se tato plocha v rámci tolerance nepřesnosti detektoru landmarků blíží 0. Pokud jsou však koutky dostatečně od sebe, stačí „mírné“ oddělení rtů od sebe a gesto je rozpoznáno. Každý uživatel si tak může najít svůj rty vymezený co nejpohodlnější tvar, který vytvoří dostatečnou plochu.

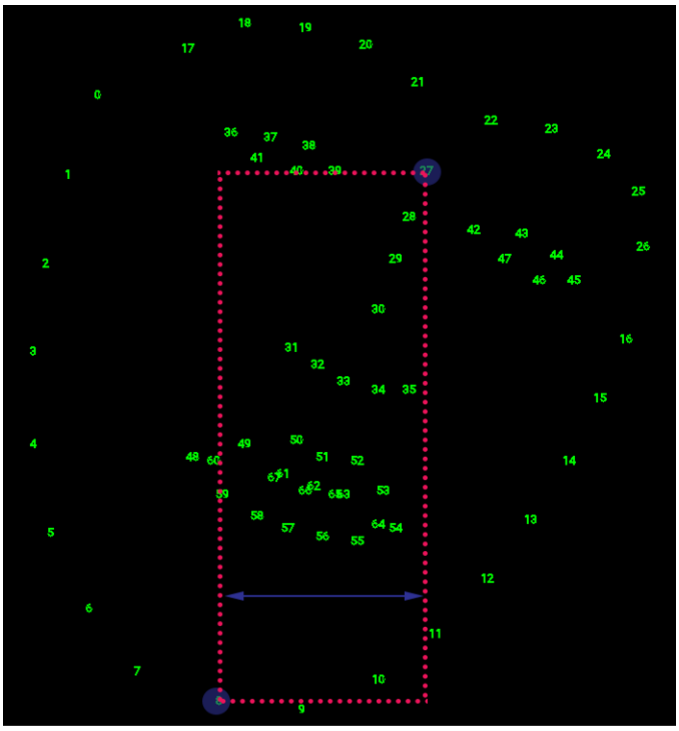


Obrázek 4.4: Ilustrace závislosti rychlosti pohybu kurzoru na poloze hlavy

Detekce naklonění hlavy do stran

Na závěr zbývají dvě nejméně pohodlné, za to z hlediska detekce nejsnazší a velice spolehlivá gesta. Jedná se o zjištění do jaké míry a na jakou stranu je hlava uživatele nakloněna. K tomuto účelu od sebe stačí odečíxt X složky souřadnic určujících pozici kořene nosu a středu brady. Pokud jsou tyto body nad sebou jejich souřadnice na X ose jsou stejné a jejich rozdíl

tedy dává 0. Záporná hodnota reprezentuje naklonění doleva, kladná pak náklon doprava. Po překročení určité úrovně na obou stranách je pak vyvolán příslušný příkaz.



Obrázek 4.5: Ilustrace závislosti rychlosti pohybu kurzoru na poloze hlavy

4.4 Rozhraní pro zadávání textu

Při hodnocení existujících řešení byla jedním z kriterií i schopnost efektivního zadávání textu. Vyjma nejnákladnějšího řešení byla ve všech ostatních případech k dispozici pouze softwarová klávesnice. Pohodlnost a rychlost navádění kurzoru pomocí pohybů hlavy na drobná tlačítka klávesnice a následného usmívání se na každé z nich, nebo ještě hůře - čekání na automatické kliknutí, si každý jistě dovede představit.

Vývoj pro platformu Android má v tomto ohledu velkou výhodu. U většiny zařízení s tímto OS je podporován převod řeči na text implicitně. Stačí ho tedy jen použít. Nejjednodušší způsob, jak získat text, se velmi podobá postup k pořízení fotografie, který je popsán v části 2.3.5. Stejně jako v tomto případě stačí vytvořit `Intent`, což je třída, která určitým způsobem popisuje, jaká operace má být provedena. V tomto případě se jedná o operaci `RecognizerIntent.ACTION_RECOGNIZE_SPEECH`. Poté stačí zavolat metodu `startActivityForResult` a v metodě `onActivityResult()` zpracovat výsledek.

Bohužel ani v tomto případě není nejjednodušší postup použitelný. Záměrem bylo takové chování aplikace, aby se při kliknutí do libovolného vstupního pole (inputbox) automaticky místo softwarové klávesnice spustil systém převodu řeči na text. Bylo tedy nezbytné najít způsob, jak reagovat na událost kliknutí do vstupního pole a především uchovat informaci o původci této události, aby bylo možné po úspěšném rozpoznání text vložit na příslušné místo. Řešení nakonec poskytlo rozhraní určené pro vstup z klávesnice. Ve chvíli, kdy Android detekuje focus ve vstupním poli, vytvoří spojení s tímto textovým polem pomocí třídy `InputConnection`. Zahájení tohoto spojení lze odchytit pomocí přepsání metody

`onCreateInputConnection()` v rámci WebView. Díky tomu je možné uložit si na toto spojení referenci a vyvolat rozpoznávání. Metoda popsaná na začátku této sekce ovšem není optimální, neboť událost vytvoření vstupního připojení lze odchytit jen v rámci třídy která jej vyvolala a na výsledek spouštět jinou aktivitu a čekat na její výsledek je možné jen v rámci aktivity. Z toho důvodu bylo nutné vytvořit vlastní instanci třídy `SpeechRecognizer`, které bylo poté možno přiřadit i vlastní `RecognitionListener()`. Díky tomu byla získána plná kontrola nad jednotlivými kroky rozpoznávání a tím pádem byla možnost poskytnout uživateli lepší zpětnou vazbu o průběhu rozpoznávání.

Drobnou vadou na kráse jest podpora ručního uzavření vstupního připojení (`InputConnection.closeConnection()`) až od API verze 24. Z toho plyne, že není možné v případě neúspěšného rozpoznávání odebrat focus právě aktivnímu vstupnímu poli. To má za následek, že pokud chce uživatel opakovat hlasové zadávání, musí nejprve kliknout jinam a poté znovu do původního textového pole. Odříznout za tuto cenu v současnosti nejpočetnější skupinu uživatelů API 23 a nižších určitě nestojí.

Posledním problémem, který bylo nutné v této souvislosti nutné vyřešit, bylo nevyžádané klikání během mluvení na mikrofon (například vyslovení písmene I přesně splňuje požadavky na gesto určené ke kliknutí). Z toho důvodu bylo nezbytné po dobu diktování deaktivovat možnost kliknout.

4.5 Zhodnocení výsledků a diskuze možných rozšíření

4.5.1 Stanovené cíle

Ze zadání této práce bylo cílem vytvořit takové uživatelské rozhraní, které umožní uživateli ovládat mobilní zařízení pouze pomocí kamery, která snímá jeho obličej. Na základě průzkumu trhu byl jako vhodná aplikace demonstrující výsledek zvolen webový prohlížeč pro platformu Android a s ohledem na množnosti a nedostatky byla stanovena kritéria pro vývoj této aplikace. Tato kritéria byla stanovena s důrazem na to, aby vzniknuvší aplikace umožnila osobám s pohybovým omezením rukou co nepohodlněji a nejefektivněji využívat služeb internetu, aniž by to pro ně představovalo vysokou finanční zátěž. Z průzkumu 2.1 totiž vyplývá, že takové řešení na současném trhu chybí.

4.5.2 Dosažené cíle

Výsledkem je funkční aplikace TouchlessBrowser pro platformu Android, kterou lze používat plně bez pomoci rukou. Prohlížeč disponuje uživatelským rozhraním zahrnujícím navádění kurzoru na základě orientace obličeje, možnost kliknutí v reakci na simulovaný úsměv a dvojici gest v podobě náklonu hlavy sloužící k optimalizaci často prováděných úkonu. Velkou výhodou výsledné aplikace je podpora hlasového zadávání textu, která výrazně zrychluje běžnou práci s aplikací a rozšiřuje její možnosti.

4.5.3 Výsledky testování a návrhy pro následující vývoj

Na seriozní formát testování v rámci této práce už bohužel nezbyl čas, nicméně ve snaze alespoň okrajově ověřit funkčnost na uživatelích bylo aplikováno několik náhodně vybraných případů užití pro úzkou skupinu testerů. Vzhledem k formátu testování výsledky nelze považovat za objektivní, ani je řádně strukturovat. Přesto však poskytují obstojné vodítko pro návrh úprav při budoucím vývoji.

Na základě testování byly vyvozeny tyto závěry:

problém: Pohodlí i efektivita používání aplikace fatálně závisí na dobrém pochopení, jak se systémem zacházet. Častou chybou bylo nevhodné namíření kamery. Další problém představovala snaha zrychlit pohyb kurzoru natočením hlavy do takové míry, že obličej již nebylo možné rozpoznat. Po vhodné instruktáži a chvíli trénování však aplikace poskytovala relativně pohodlné a především díky rozpoznávání hlasu v porovnání s podobnými metodami nesrovnatelně rychlejší prostředí.

řešení: Řešením by mohl být interaktivní průvodce, který by při prvním spuštění aplikace objasnil správné používání.

problém: Uživatelé se cítili zmateni při načítání samotné aplikace, případně při načítání stránek.

řešení: Pro oba tyto procesy by měly být přidány progressbary.

problém: Přesnost rozpoznávacího algoritmu závisí na různých parametrech. Mimo jiné se jedná například o směr zdroje světla nebo vzdálenost obličeje od kamery.

řešení: V první verzi aplikace jsou všechny citlivosti zejména z časových důvodů implementovány jako konstantní hodnoty. Pro další verzi se však počítá s uživatelským rozhraním umožňujícím tyto konstanty za běhu programu upravovat. V plánu je též automatická kompenzace vzdálenosti obličeje od kamery a podpora landscape modu (aby zařízení mohlo být v poloze „na šířku“).

Kapitola 5

Závěr

Cílem práce bylo navrhnout a implementovat uživatelské rozhraní umožňující ovládat mobilní zařízení na základě dat z kamery a demonstrovat jeho funkčnost na vhodné aplikaci. Výsledkem práce je aplikace pro OS Android poskytující plnohodnotné prostředí webového prohlížeče ovladatelné zcela bez použití rukou. Mimo to je implementována funkce pro zadávání textu hlasem, což má v porovnání s konkurencí znatelný vliv na efektivitu používání. Aplikace v současné verzi splňuje zadání a při správném použití vykazuje uspokojivé výsledky z hlediska efektivity a komfortu ovládání. Funkce aplikace jsou též demonstrovány prostřednictvím prezentačního vida, které vzniklo nad rámec této práce[1]

Během testování bylo zjištěno několik nedostatků, po jejichž odstranění by aplikace mohla poskytnout zajímavou alternativu handikepovaným v záměru využívat služeb internetu. To bylo jednou z hlavních motivací pro výběr tohoto tématu a věřím, že při pokračování ve vývoji by výsledek mohl přesáhnout význam bakalářské práce. Práci hodnotím jako velmi přínosnou i pro mou osobu, neboť jsem si osvojil principy programování pro platformu Android a získal jsem mnoho zajímavých informaci z oblasti počítačového vidění, což je velmi aktuální téma.

[1] Prezentační video https://youtu.be/-ibkSepD5Cg

Literatura

[1] Baltrusaitis, T.; Robinson, P.; Morency, L.-P.: OpenFace: An open source facial behavior analysis toolkit. IEEE, March 2016, ISBN 978-1-5090-0641-0, s. 1–10.

[2] DANILOV, A.: *Analýza programovacího jazyka Kotlin, jeho srovnání s jazykem Java 8 a praktické využití při vývoji na platformě Android [online]*. Diplomová práce, Vysoká škola ekonomická v Praze, Praha, 2017 [cit. 2018-05-14].

[3] Hubl, L.: *Webová galerie s rozpoznáváním osob*. Diplomová práce, Vysoké učení technické v Brně. Fakulta informačních technologií, 2016.

[4] Rekha, N.; Kurian, D. M.: Face Detection in Real Time Based on HOG. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, ročník 3, č. 4, 2014.

[5] Wertheim, M.: *Zpracování obrazu v systému Android - odečet hodnoty plynoměru*. Diplomová práce, Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, 2016.

[6] Žampach, J.: *Volání nativního kódu z javovské aplikace pomocí JNI*. Diplomová práce, Univerzita Pardubice, 2011.

Příloha A

Manuál k aplikaci Touchless browser

Aplikace Touchless browser složí jako internetový prohlížeč, který lze ovládat pouze pomocí pohybů hlavy a úst. Před prvním spuštěním si prosím pozorně přečtěte tyto instrukce.

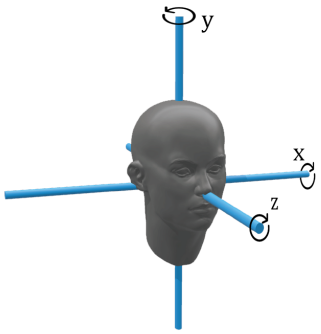
Po spuštění aplikace prosím vyčkejte, dokud nezmizí dialogové okno, které informuje o načítání rozpoznávacího systému. Při prvním spuštění aplikace to může trvat až 1 minutu.

První spuštění

Pro zajištění ideální přesnosti prosím umístěte před spuštěním aplikace zařízení tak, aby kamera mířila přímo proti obličeji. (Přesněji řečeno tak, aby normála roviny senzoru přední kamery vycházející z jeho středu mířila kolmo doprostřed snímaného obličeje.) Kamera by měla ve vzdálenosti mez 20 - 30 cm. Dále prosím mějte na paměti, že jedním z ovládacích prvků je tvar úst. Je proto vhodné během ovládaní minimalizovat hovor, nebo jinak zabránit rozpoznávání.

Navádění kurzoru

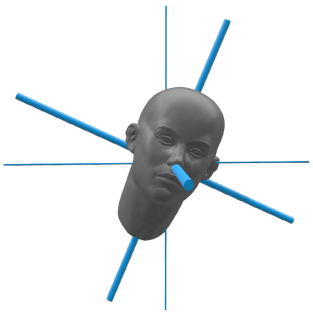
Po načtení systému je možné začít pohybovat kurzorem po obrazovce jemnými pohyby hlavy. **Směr natočení obličeje** v osách X a Y určuje směr pohybu kurzoru. Míra natočení se pak odráží na jeho rychlosti. Není při tom třeba dělat žádné velké pohyby hlavou. Naopak při příliš velkém natočení ztrácí systém schopnost rozpoznat obličej a kurzor se zastaví.
Chcete-li se přesvědčit, jak je obličej rozpoznán, případně zda je zařízení kamery ve správné poloze, je možné zobrazit náhled kliknutím na ikonku s obličejem v pravém dolní rohu.



Kliknutí

Kliknutí lze provést roztažením koutků úst od sebe a mírným odhalením zubů. Tvar úst, na který aplikace reaguje kliknutím v místě kurzoru tedy připomíná **úsměv**.

Hlasové rozpoznávání



Kdykoliv je příkaz kliknutí vyvolán nad vstupním políčkem, je automaticky aktivováno hlasové rozpoznáváni. Je tedy možné začít diktovat text. Ten je po dokončení fráze automaticky vložen do zvoleného políčka. Hlasové zadávání je také možné využít pro rychlé vyhledávání. To lze aktivovat **náklonem hlavy vpravo** podle osy Z.
Během hlasového zadávání je deaktivována možnost kliknutí, není tedy třeba se obávat nevyžádaných kliknutí během diktování.

Krok zpět

Náklonem hlavy vlevo podle osy Z je možné realizovat krok zpět v historii vyhledávače.