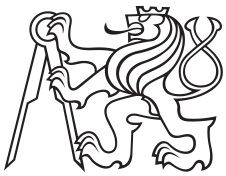
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Implementace lupy pro zrakově postižené uživatele

Michal Blažek
Program: Kybernetika a robotika
Obor: Systémy a řízení

březen 2015
Vedoucí práce: Ing. Daniel Novák, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Michal Blažek**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Implementace lupy pro zrakově postižené uživatele**

Pokyny pro vypracování:

1. Seznamte se s technologiemi pro přibližování textu, detekcí textu a převedením textu do hlasového výstupu.
2. Navrhněte algoritmus pro přibližování textu včetně intuitivního ovládání.
3. Implementujte algoritmus pro rozpoznávání textu a jeho ozvučení.
4. Výslednou aplikaci otestujte na vzorku 5 uživatelů.

Seznam odborné literatury:

[1] Al Copolillo Tony Gentry , Low Vision Intervention: Decision Making for Acquiring and Integrating Assistive Technology, International Handbook of Occupational Therapy Interventions 2015, pp 323-338, 2014
[2] R. Kline, E.P.Glinert, Improving GUI accessibility for people with low vision, Proceeding CHI '95 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Pages 114-121, 1995

Vedoucí: Ing. Daniel Novák, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 20. 2. 2015

Poděkování / Prohlášení

Rád bych poděkoval vedoucímu své práce Ing. Danielu Novákovi, Ph.D. za výborné vedení práce. Dále bych rád poděkoval své rodině, která mě podporovala během celého studia vysoké školy, a svým přátelům.
Díky patří také všem, kteří se zúčastnili testování aplikace, zejména pak pánům Michalu Jelínkovi a Martinu Procházkovi.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 10. 5. 2015

.....

Abstrakt / Abstract

Tato práce se zabývá zvětšováním textu pro potřeby slabozrakých. Seznamuje čtenáře se základní problematikou. Dále se zabývá detekcí a převodem textu z obrazu do mluveného slova. Výstupem této práce je aplikace pro chytré telefony s operačním systémem Android, která pomocí velmi jednoduchého ovládání umožní slabozrakým uživatelům přiblížit text a případně si ho nechat i přečíst. Součástí práce je také návrh stojánku pro telefon, který zjednoduší uživateli práci s aplikací. Tento stojánek je možné vytisknout na 3D tiskárně.

Klíčová slova: bakalářská práce, android, lupa, kamera, přibližování textu, detekce textu, hlasový výstup, slabozrací uživatelé, OCR

This thesis deals with the enlargement of text for the purposes of purblind people. It introduces the basic problems to the reader. It also deals with detection and conversion of text from image into spoken word. The outcome of this work is an application for smartphones with Android operating system which through a very simple operation allows visually impaired users to zoom text and possibly have it read to them. The work also includes a design for a phone stand that simplifies user's work with the application. This stand can be printed on a 3D print.

Keywords: bachelor work, android, magnifying glass, camera, zooming text, text detection, voice output, low vision users, OCR

Title translation: Magnifier Glass Implementation for Visually Impaired Users

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Lupa je optický systém používaný pro zvětšení pozorovaného objektu. Standardně je tvořena spojnou čočkou. S přibývajícím věkem se zrak mnoha lidí zhoršuje, někteří lidé se s vadou zraku rodí. Tito lidé používají princip lupy každý den a díky ní jsou schopni přečíst si například noviny nebo studijní a pracovní materiály.

V dnešní digitální době není překvapivé, že klasickou skleněnou lupu nahrazuje elektrické zařízení, jež kamerou snímá obraz a přenáší jej na displej. V češtině se pro toto zařízení používá název kamerová lupa. Výhodou těchto řešení je například možnost měnit hodnotu přiblížení či následné speciální zpracování obrazu ve snaze zvýraznit uživateli čtený text.

Úkolem práce bude vytvořit aplikaci pro dnes tak rychle se rozšiřující chytré telefony, která by zastávala stejnou činnost jako kamerová lupa, a její ovládání přizpůsobit pro zrakově postižené uživatele.

1.1 Cíl práce

V práci budu vytvářet aplikaci pro telefony s operačním systémem Android. Android je operační systém založený na jádře Linuxu, který je vyvíjen firmou Google a dostupný jako open source. Podíl Androidu mezi operačními systémy chytrých telefonů je v současné době asi 75 % [1]) .

Tato aplikace bude sloužit hlavně pro potřeby uživatelů s vadou zraku, na což bude brán ohled při navrhování ovládání. Z tohoto důvodu bude ovládání celé aplikace ozvučené tak, aby uživateli poskytovalo zpětnou vazbu o dění na displeji telefonu. Snahou bude vyrovnat se dnes prodávaným speciálním zařízením zvaným „Přenosné kamerové lupy“ a umožnit tak toto zařízení nahradit chytrým telefonem.

Obrázek 1.1. Moje aplikace Lupa na telefonu Samsung Galaxy SII

[1]) `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`

Druhou stěžejní částí samotné aplikace bude rozpoznání textu v obrazu pomocí technologie OCR. Tuto možnost dnes prodávané přenosné kamerové lupy zpravidla neobsahují. Přínosem pro zrakově postižené uživatele by měla být možnost nechat si aplikací přečíst například vizitky, obaly potravin nebo novinové sloupky.

Kapitola 2
Technologie pro přibližování textu

Před samotným vývojem vlastní aplikace jsem provedl průzkum, abych zjistil, jaké možnosti má v současné době zrakově postižený uživatel. Zvlášť jsem se zaměřil na kamerové lupy, speciální zařízení sloužící právě jen pro zvětšení, případně barevné zvýraznění textu a zvláště na softwarové řešení této problematiky pro mobilní telefony.

2.1 Kamerové lupy pro nevidomé

Existuje několik zařízení, která jsou určena pro přiblížení textu pro nevidomé. Souhrnně se označují jako kamerové lupy. Tyto kamerové lupy můžeme rozdělit podle způsobu jejich použití na Přenosné kamerové lupy, Stolní kamerové lupy a Kamerové lupy připojitelné k počítači nebo televizi. [1]

2.1.1 Přenosné kamerové lupy

Z názvu vyplývá, že se jedná o zařízení, která může mít uživatel stále při sobě v kapse či batohu. Tato idea je podpořena tím, že tato zařízení mají vestavěný akumulátor. Rozměry se pohybují od velmi malých zařízení určených pro čtení kdekoli v terénu, až po zařízení větších rozměrů, která jsou sice méně mobilní, avšak poskytují větší komfort při čtení.

Všechna tato zařízení se vyznačují jednoduchou obsluhou, práce s nimi však vyžaduje jemnou motoriku a systematickou orientaci v textu. Tato zařízení poskytují několikanásobné zvětšení, typicky až 10x, ale mnohdy i více. Velmi často jsou vybaveny tlačítkem sloužícím k pozastavení textu. Obvykle také obsahují různé režimy čtení, například černobílý (pozitivní-negativní) nebo barevné režimy, kde je písmo obarveno jinou barvou, než jakou má ve skutečnosti, a pozadí je obaveno barvou k této kontrastní. Některé z těchto kamerových lup umožňují také ukládání zastaveného obrazu. Většinou je omezen počet ukládaných obrazů na jednotky.

Obrázek 2.1. Přenosná kamerová lupa Maggie Pro [1]

Nejčastější využití je při čtení krátkých textů, v obchodech, při studiu či v zaměstnání. Ceny těchto zařízení se pohybují od 6000 Kč do více než 50000 Kč [2].

Zástupcem z této kategorie je například Maggie Pro (viz obr. 2.1). Tato kamerová lupa je zajímavá havně svými rozměry a cenou. Její velikost odpovídá velikosti kreditní karty, obrazovka má úhlopříčku 3" (7,6 cm) a výdrž na jedno nabití je asi 3 hodiny. Zajímavostí je, že k tomuto zařízení výrobce dodává i stojánek na čtení, díky kterému je lupa stále ve stejné výšce nad textem bez nutnosti toho, aby ji uživatel musel držet v ruce.

2.1.2 Stolní kamerové lupy

U těchto zařízení se nepředpokládá, že by je uživatel někam přemísťoval. Obvykle má doma či v zaměstnání vyhrazené místo, kde je lupa umístěna a využívána. Jedná se o ideální řešení pro delší čtení, které by mělo splňovat nejnáročnější požadavky uživatelů. Existují v několika provedeních, ovšem měly by se dát přizpůsobit podle požadavků uživatele. Vše je samozřejmě zohledněno v ceně, která začíná tam, kde končila cena přenosných lup. Pohybujeme se tedy v cenách vyšších než 50000 Kč.

Obrázek 2.2. Stolní kamerová lupa ClearView+ [1]

2.1.3 Kamerové lupy připojitelné k počítači nebo televizi

Tyto lupy se dají ještě rozdělit na kamerové lupy v provedení myši do ruky a kamerové lupy skládací. Kamerové lupy v provedení myši do ruky skutečně velmi připomínají počítačovou myš. Uživatel jimi přejíždí po knize či dokumentu a ten je přenášen do speciálního softwaru v počítači. Tento software se pak stará o další zpracování obrazu, jako je například zvýraznění písma změnou barev či ukládání načteného obsahu. V kombinaci s notebookem se jedná o přenosně mobilní řešení. Cana těchto lup je okolo 5000 Kč, ovšem software sloužící pro kompletní zpracování obrazu stojí dalších 10000 Kč.

Obrázek 2.3. Kamerová lupa v provedení myš do ruky Bierley ColorMouse [1]

Skládací lupy jsou vhodné nejen pro čtení, ale i psaní, a dokonce je možné je využít i na snímání okolního prostoru, například na školní tabuli. Svou konstrukcí připomínají některé stolní lampy, kde je místo samotného světla umístěna kamera. Pohyb lupy může být manuální, nebo pomocí elektromotorů řízených povely z připojeného počítače. Před touto kamerou mohou být ještě clony či čočky. Zpracování obrazu se provádí, stejně jako u provedení myši do ruky, až v připojeném počítači. Cena těchto lup je od 30000 Kč výše.

Obrázek 2.4. Skládací kamerová lupa CLEARNOTE 18X ZOOM [1]

2.2 Lupa jako aplikace pro chytré telefony

Existuje celá řada aplikací pro chytré telefony, které dostaneme po zadání výrazu „lupa“ do internetového vyhledávače. Většina z nich přináší spíše předělané ovládání fotoaparátu telefonu. Aplikace většinou poskytují uživateli podmnožinu těchto funkcí:

- přiblížení textu
  - standardně se přibližuje, dokud je jeden pixel displeje roven jednomu nebo více pixelům ze snímače. Má-li telefon fotoaparát s rozlišením 8MPx (3264 x 2448 Px) a displej rozlišení 800 x 480 Px, znamená to, že obraz bude zvětšen maximálně 4 krát oproti obrazu, který je snímán.
  - některé aplikace umí přiblížit obraz více, ovšem za cenu rozmazání obrazu
- rozsvícení blesku
- inverze barev
- zastavení obrazu
- uložení zastaveného obrazu
- otočení obrazu.

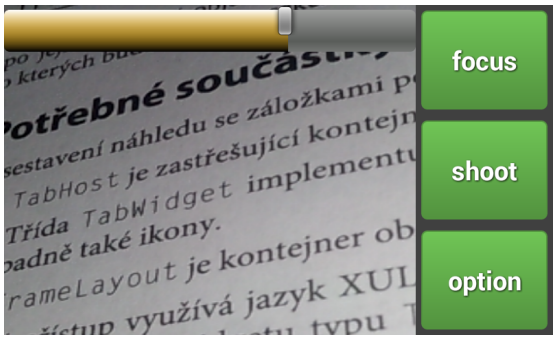
2.2.1 Aplikace Silver magnifier

Aplikace je volně dostupná z webového obchodu společnosti Google, Google Play[1]) . Již z popisu aplikace je patrné, že je inspirovaná kamerovou lupou MaggiePro (viz kapitola 2.1). Uživatel může obraz přibližovat, ovšem pouze standardně do rovnosti jednoho pixelu snímače a displeje. Dále je mu umožněno rozsvítit blesk a pořídit fotografii.

[1]) `https://play.google.com/store/apps/details?id=com.nenara.camera2`

Ovládání je řešeno pomocí velkého táhla, sloužícího pro nastavení hodnoty přiblížení a tří velkých tlačítek přičemž, pod jedním z nich je ještě rozbalovací menu. V aplikaci je také možnost ozvučení, bohužel mně z neznámých důvodů nefungovalo, a tak jsem jej nemohl otestovat.

Každopádně se jedná o aplikaci, jejíž ovládání je alespoň částečně přizpůsobeno zrakově postiženým uživatelům, a to právě díky velkým ovládacím prvkům.



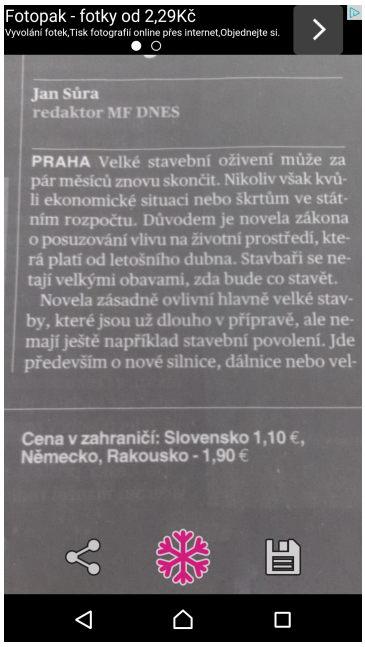
Obrázek 2.5. Prostředi aplikace Silver magnifier

2.3 Aplikace zvětšovací sklo

I tato aplikace je volně dostupná na Google Play [1]). Samozřejmě umožňuje přibližovat obraz, opět ovšem pouze standardně do rovnosti jednoho pixelu snímače a displeje. Uživatel si může obraz zastavit a následně uložit. K dispozici je také možnost zobrazení v negativních barvách, úprava vyvážení bílé barvy a zhasnutí či rozsvícení blesku.

Ovládání je ovšem řešeno standardními prvky a není tedy přizpůsobeno potřebám slabozrakých či nevidomých uživatelů.

Obrázek 2.6. Standardní režim



Obrázek 2.7. Inverzní režim

[1]) `https://play.google.com/store/apps/details?id=mmapps.mobile.magnifier`

Kapitola 3
Detekce a rozpoznání textu

Získání textu z obrázku lze obecně rozdělit do dvou fází. V první je třeba nejprve nalézt množinu oblastí, v níž se text pravděpodobně vyskytuje (tzv. lokalizace). Následným úkolem je z těchto malých oblastí, které již budou obsahovat pouze text, tento text rozpoznat.

3.1 Detekce textu

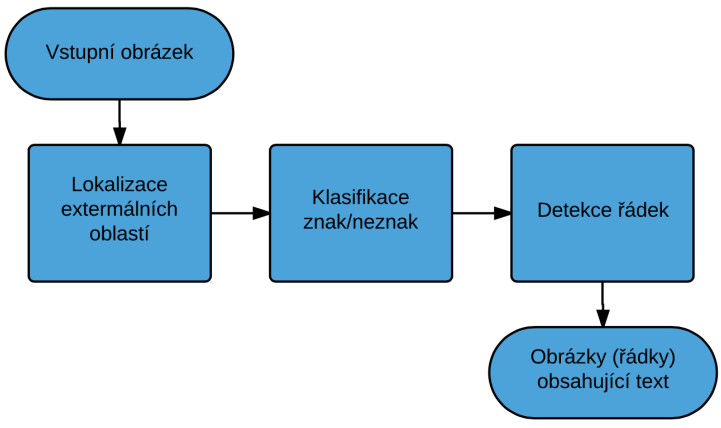
Jednou ze speciálních kategorií hledání objektů v obraze je právě detekce textu v obraze. Se zvyšujícím se zájmem o strojové porozumění objektu je rozpoznání rovinného textu v obraze často řešeným problémem. Tuto problematiku můžeme rozdělit na několik částí, jež jsou níže seřazeny podle úrovně obecnosti:

- detekce statického textu na jednobarevném pozadí
- detekce statického textu na různobarevném pozadí
- detekce pohyblivého textu ze statického obrazu, kde je známé umístění (například detekce SPZ)
- detekce rovinného textu v reálných scénách
- úplně obecná detekce textu za libovolných okolních podmínek.

V dnešní době se většina implementací a publikací zabývá rozpoznáváním textu v reálných scénách[3].

3.1.1 Metoda podle Neuman a Matas

Tato metoda předpokládá, že jednotlivá písmena jsou extermálními oblastmi a že řádky textu mají dané geometrické vlastnosti. Jak je vidět z vývojového diagramu (viz obr. 3.1), v obrázku je nutné nejprve najít extermální oblasti. Jedná se o obasti tvořené všemi pixely, které jsou tmavší než daná prahová hodnota. Protože obrázky běžně ukládáme pomocí barevného prostoru RGB, je vhodné převést si vstupní obrázek do stupňů šedi.

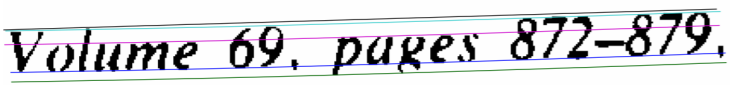


Obrázek 3.1. Vývojový diagram detekce řádků textu v obraze

Jako další krok je třeba rozpoznat extermální oblasti, v nichž by se mohl znak vyskytovat. K tomu slouží klasifikátor, pomocí něhož oblast ohodnotíme. Podle [4] lze použít klasifikátor AdaBoost.

Sestavení písmen do jednotlivých řádků vychází z předpokladu, že horní a dolní okraje písma sedí na dvou dotažnicích (nahoře i dole). Metoda vyhledá jednotlivé trojice písmen, jež tomuto modelu odpovídají, a poté tuto množinu prořeže podle následujících podmínek [4] [5]:

- vzdálenost písmen normalizovaná jejich výškou
- vzájemný poměr výšek
- vzájemné překrývání
- úhel svíraný těžišti písmen.

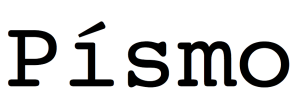


Obrázek 3.2. Dotažnice v textu [6]

3.2 Optické rozpoznávání znaků

OCR (z anglického Optical Character Recognition) je metoda převádějící text z tištěné podoby do podoby elektronické. Rozpoznávání textu je 20 až 25 krát rychlejší než ruční přepisování. S elektronickou verzí textu lze již následně běžně pracovat.

Strojový převod z obrázku na text se provádí v několika krocích. Prvním krokem je segmentace řádek (viz kapitola 3.1.1). V následující fázi se řádek rozdělí na jednotlivá slova a ta se rozdělí na samostatné znaky. Pokud je text psán fontem s neproporciálními písmeny (např. Courier, viz obr. 3.3), je segmentace znaků jednoduchá, každý znak má v tomto případě konstantní šířku. Častěji se ovšem používají fonty proporciální, protože jsou lépe čitelné lidským okem. U těchto fontů dochází k převisu části písmen a k dotýkání znaků (viz obr. 3.4). V roce 1966 došlo v USA ke standardizaci tzv. písma OCR-A. Jedná se o písmo, které je upraveno a zjednodušeno právě pro snadné strojové čtení, ale zároveň také pro čtení okem (viz obr. 3.5). V posledním kroku je ke každému tvaru přiřazen nejvíce odpovídající znak [7].



Obrázek 3.3. neproporcionální font Courtier

Obrázek 3.4. proporcionální font Helvetica

Obrázek 3.5. font speciálně navržen pro strojové čtení OCR-A

Úspěšnost převodu tištěného textu na text elektronický je přímo úměrná kvalitě obrazové předlohy. Text získaný samotným převodem je zpravidla vždy nutné podrobit důkladné jazykové korektuře, aby se eliminoval výskyt špatně rozeznaných znaků.

3.2.1 Tesseract OCR

OCR Tesserct Engine [6] byl vyvinut v laboratořích společnoti HP v letech 1985 - 1995. Poté se jeho vývoj částečně uspal až do roku 2006. Od tohoto roku patří Tessaract pod Google a vývoj nadále pokračuje pod licencí Apache 2.0.

Volba Tesseractu pro převod textu v obrazu do textu v mé aplikaci vyplynula z kladných ohlasů. Kvalita je srovnatelná s komerčně prodávanými API jiných výrobců (ABBYY OCR [1]), Aspose.OCR [2])).

Přestože sám výrobce upozorňuje na to, že byl engine původně vyvíjen pouze pro angličtinu, dnes je jeho velkou předností právě jednoduchost inicializace různých jazyků. Jednoduchým stažením trénovacích dat lze zvolit jazyk, v němž bude Tesseract pracovat. Kvalita distribuovaných dat se v závislosti na jazyku mění, což je vidět i na velikosti jednotlivých balíčků. Například trénovací data pro angličtinu mají velikost 12,1 MB, kdežto pro češtinu už pouze 1 MB. V praxi to nejčastěji znamená nižší podporu různorodých fontů písma.

Tesseract je napsán v jazyce C, ovšem existuje podpora pro většinu platforem včetně Linuxu, Windows, Mac OS X, iOS i Android.

3.3 Existující mobilní aplikace pro rozpoznávání textu

Existuje řada aplikací, jež se zabývají převodem tištěného textu do digitální podoby. Pro potřeby nevidomých uživatelů je ovšem třeba, aby bylo ovládání aplikace přizpůsobeno právě možnostem a požadavkům těchto uživatelů. Tyto požadavky však splňuje pouze několik málo aplikací.

3.3.1 KNFB Reader

KNFB Reader je aplikace pro iPhone, tedy pro chytré telefony vyráběné firmou Apple s operačním systémem iOS. Jejím cílem je rozpoznání a následné přečtení textu z reálných scén. Cílovou skupinou této aplikace by podle stránek výrobce měli být právě nevidomí uživatelé.

Nejsem si jistý, zda jsou nevidomí uživatelé skutečně schopni ovládat tuto aplikaci. Nastavení je poměrně obsáhlé a není nevidomým nijak zvlášť přizpůsobeno.

Aby se text správně rozeznal, je třeba fotoaparátem zamířit na text, počkat, až fotoparát zaostří, a následně vyfotit. Ani jedna z těchto akcí není uživateli zvukově oznámena.

V případě, že je text ostrý, je ovšem rychlost a přesnost aplikace obdivuhodná. Aplikace začne číst text během několika málo vteřin, a to aniž by měla nalezený kompletní text. Začne číst a během čtení paralelně pracuje na vyhodnocení zbylé části obrazu.

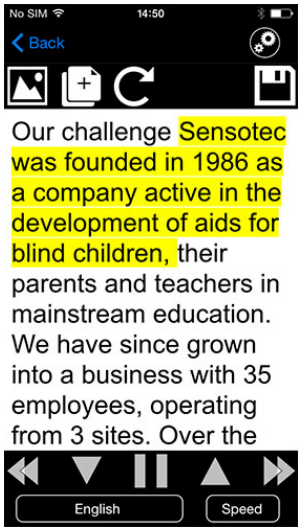
Aplikace pracuje asi v patnácti nejvíce světově používaných jazycích, v češtině ovšem ne.

3.3.2 Google Translate

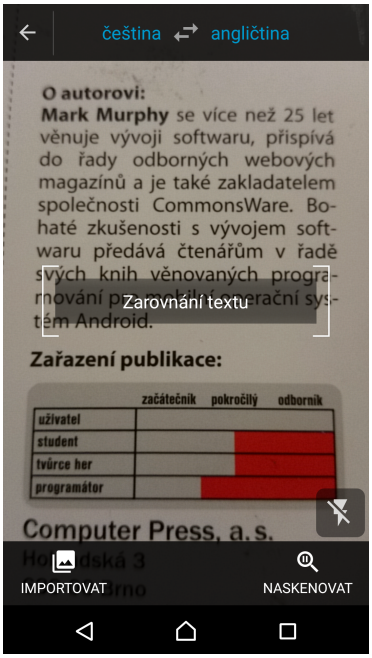
Již z názvu aplikace vyplývá, že její hlavní směr není rozeznání textu, ale překlad jednotlivých jazyků. Součástí aplikace je ovšem i možnost získat z obrázku text. Uživatel jednoduše označí oblast, kde se text nachází a ten se následně odešle na servery Googlu, kde je zpracován. Z toho také vyplývá nutnost internetového připojení pro chod aplikace.

[1]) `http://ocrsdk.com`
[2]) `http://www.aspose.com/cloud/ocr-api.aspx`

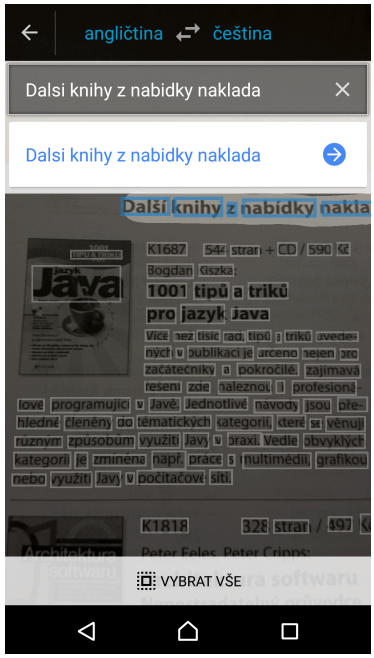
Obrázek 3.6. Prostředí aplikace KNFB Reader [8]



Obrázek 3.7. Rozeznaný text [8]



Obrázek 3.8. Aplikace požaduje správnou orientaci textu



Obrázek 3.9. Označený text okamžitě zobrazuje, případně překládá

Aplikace si při startu vyžádá orientaci, v níž chce, aby text v obrázku byl, a v níž jej také bude hledat. Pokud toto nedodržíme, aplikace skutečně nic smysluplného nenajde.

Samotná rychlost a spolehlivost jsou ovšem vysoké. Aplikace reaguje opravdu okamžitě po označení textu, což napovídá, že již v telefonu pravděpodobně probíhá nějaké zpracování obrazu a na servery Googlu se přenáší méně dat než celá fotografie.

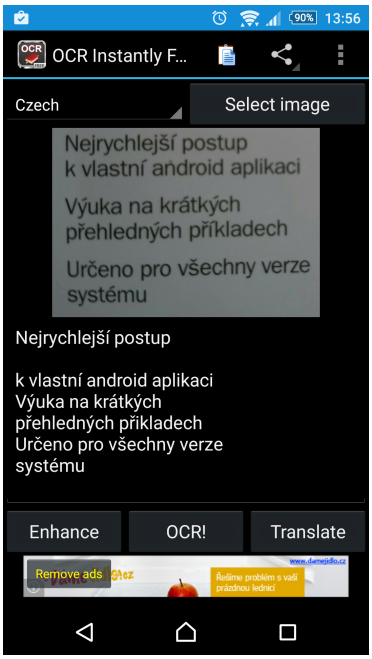
Poměrně nedávno Google do své aplikace přidal také překlad textu v rámci rozšířené reality. V praxi to funguje tak, že stačí telefon namířit na text, který chcete přeložit, a ten se na displeji rovnou ukáže ve vámi zvoleném jazyce. Telefon opět určuje orientaci

textu podle orientace telefonu. Tento překlad v rozšířené realitě funguje zatím pouze mezi vybranými jazyky, mezi které čeština nepatří.

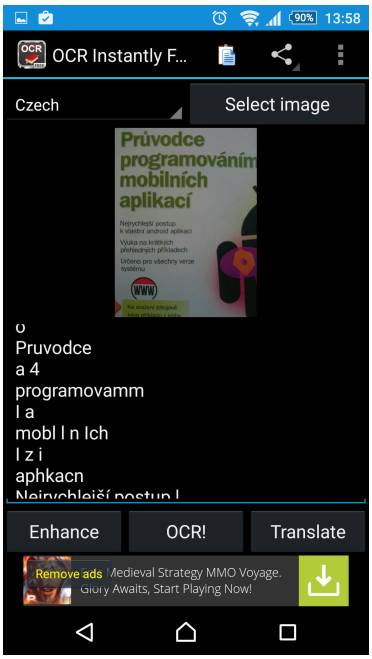
3.3.3 OCR Instantly Free

Není problém najít mnoho dalších aplikací, které mají dle svého popisu implementovanou technologii OCR. Ovládání i kvalita většiny z nich je dosti obdobná. Jedním příkladem za všechny může být OCR Instantly Free. Po spuštění aplikace jsme vyzváni k vybrání či pořízení fotografie. U vybrané fotografiie je opět nutné zvolit orientaci textu a pro lepší výsledek rozpoznání i oblast, v níž se text nalézá.

Na kvalitu rozpoznání textu má velký vliv kvalita fotografie, kontrastnost textu a pozadí či korektní orientace. Pakliže je toto dodrženo, rozpoznání probíhá celkem rychle a bez větších nepřesností.



Obrázek 3.10. Nejjednodušší scéna pro rozpoznání textu



Obrázek 3.11. Složitější scéna pro rozpoznání textu, více fontů a barev písma, různě velká písma

Kapitola 4
Převod textu do hlasového výstupu

Existují dva způsoby, jak s námi může software komunikovat pomocí mluveného slova. První z nich vypadá tak, že software obsahuje množinu audio souborů, které v případě požadavku přehrává. Výhoda tohoto způsobu je věrohodnost hlasového projevu, ovšem nevýhodou je právě omezení pouze na danou množinu audio souborů. Tento způsob se používá například v řadě automobilových navigací či u hlásičů na nádražích. Využívá se skládání těchto audio stop. Výsledná věta je složena z několika samostatně nahraných slov. Avšak k ozvučení aplikace to není zrovna vhodné. Velikost aplikace by přímo úměrně rostla s počtem frází, jež bychom chtěli vyslovovat. Pro použití na čtení rozpoznaného textu je toto řešení nepoužitelné.

Řešením je strojová syntéza řeči.

4.1 Syntéza řeči

Syntéza řeči je proces, jehož cílem je co nejvěrněji reprodukovat lidskou řeč konkrétního člověka. A to i s výškou, zabarvením a tónem řeči. Zkrátka převést text do lidského hlasu tak, jako by ho četl reálný člověk.

Technologie zde používaná se označuje jako TTS (text to speech). Text se nejprve analyzuje a normalizuje. Následně dochází k převodu do výslovnostní podoby, která se skládá z akustických jednotek. Z této formy pak vzniká vlastní hlasový výstup.

Mobilní operační systém Android převod textu na řeč podporuje a Google nabízí zdarma i jazyky s vysokou kvalitou výstupu. Jak je ovšem pochopitelným zvykem, jedná se pouze o světově nejčastěji používané jazyky, mezi nimiž čeština ani slovenština není.

Existuje ovšem řada aplikací zabývajících, se hlasovou syntézou s větší podporou jazyků. Například Acapela TTS Voices[1]) nebo SVOX [2]).

[1]) `https://play.google.com/store/apps/details?id=com.acapelagroup.android.tts`
[2]) `https://play.google.com/store/apps/details?id=com.svox.classic`

Kapitola 5
Tvorba android aplikace

K vytvoření své Android aplikace jsem použil software developer kit (SDK) [9]. Google doporučuje použít buď vývojové prostředí Eclipse s Android developer tools (ADT), nebo Android Studio, které jsem použil já a které je vyvíjeno právě Googlem. Programovacím jazykem, který se používá při tvorbě android aplikací, je objektově orientovaný jazyk Java.

5.1 Výběr API levelu

Volba API (Application Programming Interface) levelu je velmi důležitá. Definuje, s jakou nejnižší (a tím pádem i nejstarší) verzí systému Android bude aplikace kompatibilní.

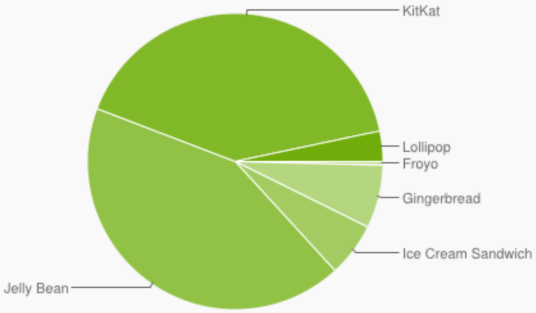
API level zaručuje zpětnou kompatibilitu, a tak by měla aplikace pez problémů fungovat na všech vyšších a tedy novějších verzích operačního systému Android.

Vím, že budu chtít importovat knihovnu Tesseract (viz kapitola 3.3) pro rozeznávání textu, ale i její minimální verze API je právě 15.

Protože žádné další omezující požadavky na API nemám, nebudu bezdůvodně zvyšovat počet nepodporovaných verzí systému, a budu tak pracovat s verzí API 15.

Z průzkumu vyplývá, že nižší verzi systému Android využívá asi 7,3 % uživatelů této platformy a toto číslo se stále snižuje. Pro tyto uživatele bude tedy aplikace nedostupná.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%



Obrázek 5.1. Zastoupení verzí API (k 23. 3. 2015) [9]

5.2 Ovládání aplikace

Protože moje aplikace bude primárně určena pro slabozraké či nevidomé uživatele, budu se snažit vytvořit co možná nejjednodušší ovládání.

Celé prostředí se bude chovat jako seznam, ve kterém se od jedné položky k další bude přecházet krátkým dotykem na pravé či levé části displeje telefonu. Otevření vybrané položky se provede dlouhým dotykem jednoho prstu na vybrané položce. Zavření

otevřené položky a tedy i návrat o úroveň výše se provede dlouhým stiskem dvou prstů na obrazovce. K vytvoření gest bude použit GestureDetector.

Každý pohyb uživatele bude navíc hlášen hlasovým výstupem tak, aby uživatel vždy věděl, co se děje. Konečná podoba ovládání aplikace je ovlivněná připomínkami získanými během testování aplikace (viz. kapitola 7.1) ve snaze vytvořit co nejjednodušší ovládání pro potřeby nevidomých [10] [11].

5.3 Vývojový diagram ovládání aplikace

Vývojový diagram (viz obr. 5.2) popisuje činnost a ovládání celé aplikace. Při spuštění aplikace se uživateli ukáže živý náhled z kamery telefonu. Aplikace se spustí v režimu, ve kterém jeden pixel snímače odpovídá jednomu pixelu displeje a blesk telefonu je zapnutý. Uživatel může klepnutím v levé polovině displeje obraz oddálit a klepnutím v pravé polovině displeje obraz přiblížit.

Dlouhým dotykem jednoho prstu na displeji uživatel vstoupí do menu, v němž má na výběr tři položky. Dlouhým stiskem dvou prstů uživatel toto menu opustí a vrátí se zpět k živému náhledu aplikace. Tento ovládací prvek, návrat o úroveň výše dlouhým stiskem dvou prstů, bude jednotný pro celou mou aplikaci.

První možností v tomto menu je pozastavení obrazu. V tento okamžik je uživateli zobrazen obrázek, který se uložil ve chvíli, kdy dlouhým dotykem jednoho prstu vcházel do tohoto menu. Zobrazí-li se tento obrázek, může uživatel opětovným dlouhým podržením jednoho prstu vyvolat rozpoznání textu v tomto obrázku. Dlouhým stiskem dvou prstů se opět vrátíme zpět do menu.

Druhou možností je rozsvícení či zhasnutí blesku fotoaparátu.

Poslední možností je inverze barev fotoaparátu. Tedy stav, kdy se bílá barva zobrazuje jako černá a naopak. Mezi aktivitou a neaktivitou těchto možností se přepíná dlouhým stiskem jednoho prstu.

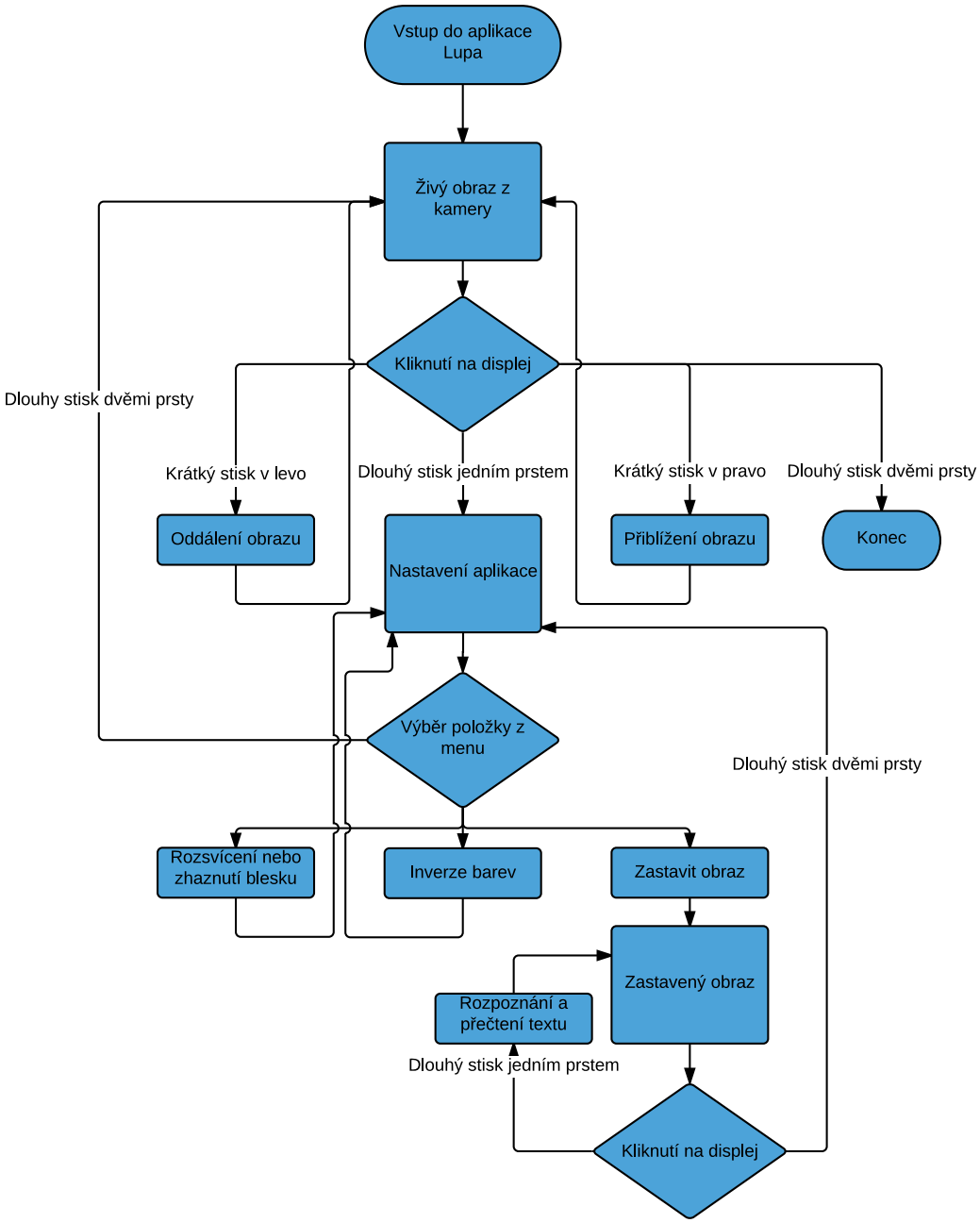
5.4 Vlastní iplementace a řešení problémů

5.4.1 Automatické zaostřování

Automatické zaostřování je velmi důležitá část aplikace. Nelze si například představit, že by rozpoznání textu pomocí OCR proběhlo v pořádku na neostrém obrázku. Také můžeme těžko předpokládat, že by nevidomý či špatně vidící uživatel byl schopen vždy rozeznat, zda je obraz opravdu ostrý.

Android od verze API 14 podporuje mód kamery FOCUS MODE CONTINUOUS PICTURE, pro plynulé zaostřování obrazu. Je pouze na svobodné vůli výrobce telefonu, zdali tento mód bude jeho firmware v telefonu podporovat.

Protože jsem tento mód našel pouze v těch nejlepších telefonech, vyřešil jsem tento problém jinak. Moje aplikace při vytvoření živého náhledu vytvoří ještě dvě další vlákna. První se bude starat o detekci pohybu telefonu. Uvažuji totiž tak, že pokud se telefon výrazně pohne, pravděpodobně se přesunul, a je tedy potřeba znovu zaostřit. Z toho také vyplývá, že pokud uživatel zjistí, že obraz není ostrý, stačí telefonem zaklepat a dojde k přeostření obrazu. Jedná se o další jednoduché ovládací gesto. Klasicky se v aplikacích ovládajících kameru zaostřuje prostým dotykem na obrazovku, čehož já však již využít nemohu, protože gesta dotyku používám pro přiblížení a oddálení.



Obrázek 5.2. Vývojový diagram aplikace ovládání

```
private final SensorEventListener
mSensorListener = new SensorEventListener() {

    public void onSensorChanged(SensorEvent se) {
        mX = se.values[0];
        mY = se.values[1];
        mZ = se.values[2];
```

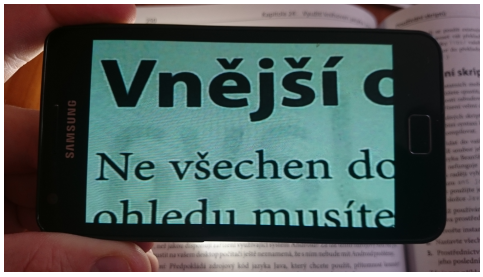
```
            mAccelLast = mAccelCurrent;
            mAccelCurrent = (float) Math.sqrt( (mX * mX + mY *
                                                mY + mZ * mZ));
            float delta = mAccelCurrent - mAccelLast;
            mAccel = mAccel * 0.9f + delta; // perform low-cut filter
        }

        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }
        };
```

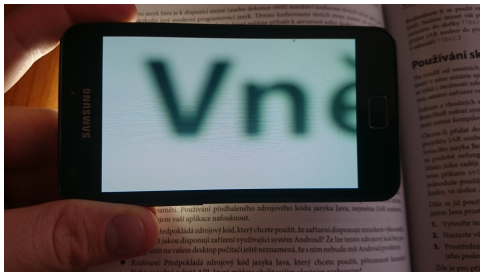
Jak je vidět v kódu výše, k detekci pohybu používám SenzorEventLisener[9], který vyvolá přerušení při každém pohybu telefonu. Uvnitř vlákna detekuji, o jak velký pohyb šlo a překročí-li danou prahovou hodnotu, nastavím požadavek na vyvolání autofocusu.

Druhé vlákno bude sloužit k samotnému autofocusu. Pokud přijde požadavek na zaostření, toto vlákno bude volat periodicky autofocus, dokud se mu zaostřit nepovede. K samotnému autofocusu využívám AutoFocusCallback[9].

Obě vlákna neběží trvale. Pravidelně se pozastavují a uvolňují prostředky zbytku systému, aby nedocházelo k „zamrzání“ telefonu. Obě vlákna také vznikají a zanikají současně. Vznikají ihned po vytvoření aplikace Lupa a zanikají ve chvili, kdy zanikne i aplikace. K jejich uspání dojde také při vstupu do menu, protože v tuto chvíli není třeba hlídat pohyb ani znovu zaostřovat obraz kamery.



Obrázek 5.3. Správně zaostřený obraz



Obrázek 5.4. Neostrý obraz

Správně zaostřený obraz je uživateli oznámen „cvakutím“, které připomíná zvuk reálného fotoaparátu při zaostření.

V tomto ohledu má moje řešení ještě dva nedostatky. Jak bylo napsáno výše, detekuji pohyb telefonu, po kterém nechám aplikaci znovu zaostřit Když ovšem uživatel bude dostatečně opatrný a nebude tedy telefonem třást, podaří se mu změnit ohniskovou vzdálenost, aniž by vyvolal přeostření, a obraz tak bude rozostřen.

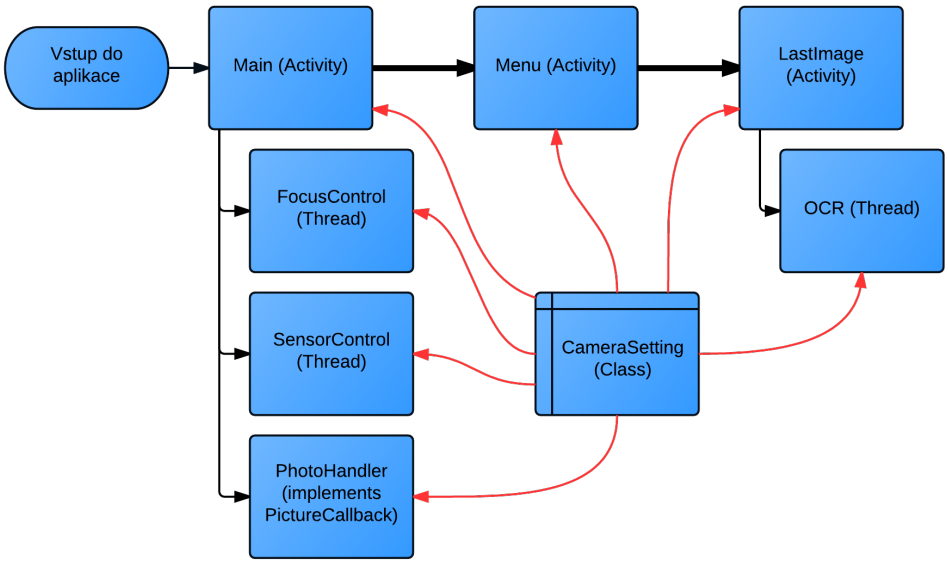
Další nedostatek je způsoben samotným fotoaparátem. Ten potřebuje jistou minimální ohniskovou vzdálenost foceného objektu od čočky fotoaparátu, aby na něj dokázal zaostřit (viz obr. 5.4). Věřím, že oba tyto nedostatky lze eliminovat seznámením uživatelů s ovládáním.

5.4.2 Nastavování parametrů kamery napříč aktivitami

K jakékoli změně stavu kamery je potřeba vždy ta samá rutina. Nejprve je nutné přečíst aktualní stav kamery a to pomocí metody getParameters() třídy Camera. Tyto parametry následně podle potřeby změnit a poté, je opět zapsat pomocí setParameters(). V aplikaci jsem řešil problém vznikající tím, že z menu chci nastavovat parametry kamery, která v tu chvíli není aktivní. Živý náhled pozastavuji při vstupu do menu, ale právě z tohoto menu chci nastavovat parametry kamery.

Současně také budu potřebovat přístup k datům kamery z vlákna starajícího se o autofocus.

Tento problém jsem vyřešil vytvořením vlastní třídy CameraSetting. V této třídě shromažďuji metody starající se o nastavení a práci s kamerou. Při tvorbě této třídy jsem využil návrhový vzor Singleton (jedináček).



Obrázek 5.5. Vývojový diagram programu aplikace Lupa, třida CameraSetting vytvořená podle návrhového vzoru singleton slouží ke sdílení informací mezi ostatními částmi aplikace

Singleton zajišťuje, že v programu existuje vždy pouze jedna instance dané třídy. Díky tomu můžu z jakékoli třídy nastavit parametry této třídy tak, jak je potřeba a toto nastavení vyčíst opět v jiné třídě, aniž by mezi nimi existoval nějaký přímý vztah.

5.4.3 Zastavení obrazu

Zastavení obrazu kamery je vyvoláno výběrem položky „Zastavit obraz“ v menu aplikace. Toto menu je ovšem až za živým náhledem, jak je znázorněno ve vývojovém diagramu (viz obr. 5.2). Uživatel tedy požádá o zastavení obrazu až ve chvíli, kdy již živý obraz nevidí. Zobrazit obraz, který v tuto chvíli kamera skutečně vidí, tedy není správná volba, a to proto, že by uživatel nevěděl, co vlastně snímá.

Tento problém, způsobený snahou o co nejjednodušší ovládání celé aplikace, jsem vyřešil následujícím způsobem. Při každém přechodu uživatele z obrazovky živého náhledu kamery do menu vyfotím poslední obraz na kameře a uložím jej do paměti telefonu. Pokud uživatel otevřel menu z jiného důvodu, než je zastavení obrazu, například z důvodu rozsvícení blesku telefonu nebo zapnutí inverzních barev, uložený obraz se ve chvíli návratu zpět k živému náhledu smaže. Pokud ale uživatel vybere zastavení obrazu, zobrazí se mu právě tento uložený obraz.

K samotnému vytvoření obrazu jsem použil třídu PhotoHandler implementující rozhraní PictureCallback. Toto rozhraní iplementuje metodu onPictureTaken, která nese vyfocený obrázek jako pole bytů (viz. následující kód).

Cestu k aktuálnímu obrázku si přenáším ve třídě CameraSetting (viz kapitola 5.4.2).

```
@Override
   public void onPictureTaken(byte[] data, Camera camera) {
       mCameraSetting = CameraSetting.getInstance();

       File pictureFileDir = getDir();

       if (!pictureFileDir.exists() && !pictureFileDir.mkdirs()) {
           Log.d(TAG, "Can't create directory to save image.");
           return;
       }
       SimpleDateFormat dateFormat =
                        new SimpleDateFormat("yyyymmddhhmmss");
       String date = dateFormat.format(new Date());
       String photoFile = "Picture_" + date + ".jpg";

       String filename = pictureFileDir.getPath() + File.separator +
                                                          photoFile;
       File pictureFile = new File(filename);
       mCameraSetting.setPathToLastPicture(filename);

       try {
           FileOutputStream fos = new FileOutputStream(pictureFile);
           fos.write(data);
           fos.close();

           Log.d(TAG, "New Image saved:" + photoFile);
       } catch (Exception error) {
           Log.d(TAG, "Image could not be saved.");
       }
   }
```

5.4.4 Načítání velkých bitmap

Během zobrazování fotografií uživateli jsem se opakovaně setkal s problémem, kdy aplikace přestala fungovat z důvodu nedostatku paměti, OutOfMemoryError. Tato chyba byla způsobena neefektivním načítáním velkých fotografií, jejichž rozlišení je několikrát větší, než je rozlišení displeje telefonu.

Řešením tohoto problému je načítat obrázky v rozlišení, v němž je budu i zobrazovat. Tento problém řeší v systému Android třída BitmapFactory, která slouží k vytváření bitmap z různých zdrojů.[9]

```
InputStream in = new FileInputStream(fileWithPicture);

// decode image size (decode metadata only, not the whole image)
BitmapFactory.Options options = new BitmapFactory.Options();
options.inJustDecodeBounds = true;
BitmapFactory.decodeStream(in, null, options);
in.close();

// save width and height
int inWidth = options.outWidth;
int inHeight = options.outHeight;

// decode full image pre-resized
```

```
options.inSampleSize = Math.min(inWidth/screenWidth,
                                    inHeight/screenHeight);
// decode full image
Bitmap roughBitmap = BitmapFactory.decodeStream(in, null, options);
```

Tím, že nastavíme hodnotu inJustBounds = true, se vyhneme přidělení paměti při dekódování. Můžeme tak zjistit skutečné rozměry obrázku, inWidth a inHeight.

Metoda inSampleSize ve třídě BitmapFactory slouží ke zmenšení obrázku. Nastavíme-li parametr roven 2, budou rozměry dekódovaného obrazu poloviční oproti obrazu původnímu.

Jako hodnotu inSampleSize použijeme menší z podílů rozměrů obrázku a displeje. Tím získáme bitmapu, jejíž rozměry budou korespondovat s rozměry displeje a přitom nebude paměť blokována informacemi, které nevyužijeme.

5.4.5 Implementace OCR

Jak jsem již uvedl dříve, použil jsem pro rozeznávání textu z obrazu Tesseract. Tesseract poskytuje API pro mnoho platforem včetně Androidu. Stáhnutí projektu „tess-two“, jenž obsahuje nástroje Tesseractu pro použití v operačním systému Android, následné překompilování pro použití v AndroidStudiu, je možné pomocí těchto příkazů v terminálu.

```
git clone git://github.com/rmtheis/tess-two tess
cd tess
cd tess-two
ndk-build
android update project --path .
ant release
```

Do vzniklého projektu je ještě třeba doplnit soubor „build.grandle“, který definuje, s jakými parametry bude projekt překládán (například určuje minimální podporované SDK či verzi jazyka JAVA).

Můj „build.grandle“ má nasledující tvar.

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.0.0'
    }
}

apply plugin: 'com.android.library'

android {
    compileSdkVersion 21
    buildToolsVersion "21.0.2"

    defaultConfig {
        minSdkVersion 15
        targetSdkVersion 21
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_7
```

```
        targetCompatibility JavaVersion.VERSION_1_7
    }

    sourceSets.main {
        manifest.srcFile 'AndroidManifest.xml'
        java.srcDirs = ['src']
        resources.srcDirs = ['src']
        res.srcDirs = ['res']
        jniLibs.srcDirs = ['libs']
    }
}
```

Poslední částí, kterou je třeba udělat pro úspěšnou implementaci Tesseractu, je přidání závislosti mého projektu na tomto projektu „tess-two“. Toho docílíme následovně:

- do „setting.grandle“ přidáme adresářovou cestu a závislost projektu, výsledek bude vypadat tedy asi takto:

```
include ':app', ':tess-two'
project(':tess-two').projectDir = new File('libs/tess-two')
```

- do „build.grandle(Module: app)“, který je součástí projektu, přidáme do části „Dependencies“ požadavek na překlad projektu „tess-two“

```
compile project(':tess-two')
```

Stejného efektu lze dosáhnout také pomocí AndroidStudia následujícím postupem: „File - Project structure... - Dependencies“ a zde přidat projekt „tess-two“.

5.4.6 Orientace textu v obraze

Tesseract požaduje, aby vstupní obrázek, ve kterém budeme hledat text, byl správně orientovaný. Znamená to, aby byl text v obrázku vodorovně s odchylkou maximálně několik stupňů.

Řešením je ukládat si informaci o natočení telefonu ve chvíli, kdy byla pořízena fotografie, v níž budeme text vyhledávat. Informace o natočení fotografie ukládám přímo do metadat pořízené fotografie.

Toto řešení funguje velmi dobře při klasickém fotografování, kdy člověk stojí a fotí objekt před sebou. Podle toho, co fotí, pak telefon drží klasicky na „stojato“ nebo na „ležato“.

Problém nastává, když uživatel fotí něco, co například leží na stole, a má tedy telefon vodorovně se zemí. V tuto chvíli akcelerometr telefonu nedává žádnou relevantní hodnotu a místo aktuálního natočení telefonu vrací hodnotu -1, která symbolizuje, že úhel není měřitelný.

První přístup, který mě napadl, byl pracovat s poslední známou hodnotou natočení. Tedy předpokládat, že uživatel nejprve telefon orientoval, a poté teprve naklonil nad focený papír. Ukázalo se, že toto není intuitivní ovládaní, protože nikdo, na kom jsem toto řešení vyzkoušel, takto nepostupoval.

Místo tohoto postupu jsem využil implementovaného Tesseractu, který dokáže vrátit hodnotu pravděpodobnosti toho, že jím nalezený text je správný. Tesseract vrátí hodnoty správného nalezení odstavců, řádků, slov i znaků. Tyto hodnoty jsem zprůměroval. Jestliže průměrná hodnota dosáhla prahové hodnoty, považuji aktuální orientaci za správnou a uživateli vrátím nalezený text. Pakliže této hodnoty pro aktuální orientaci nedosáhne, otočím obrázek o 90° a postup opakuji. Jestliže prahové hodnoty

nedosáhnu ani s jedním natočením obrazu, vrátím text, jenž dosáhl nevyšší míry pravděpodobnosti, ovšem pouze pokud překročím minimální prahovou hodnotu. V případě, že tuto hodnotu nepřekročím, dozví se uživatel pouze to, že se text nalézt nepodařilo.

Jestliže prohledám obrázek a naleznu orientaci vyhovující pro nalezení textu, tuto orientaci uložím pro hledání v dalším obrázku. V případě, že uživatel vyfotí další text, aniž by telefon narovnal a aktivoval tak akcelerometr, bude hledání správné orientace začínat na hodnotě, kde se text nalezl naposledy. Tímto způsobem jsem schopen podstatně zrychlit hledání textu v případě několika po sobě jdoucích obrázků.

Kapitola 6
Distanční stojánek na telefon

Během návštěvy v SONSu (Sjednocená organizace nevidomých a slabozrakých) v Praze [12] jsem měl možnost si v místní prodejně pomůcek pro nevidomé a krátkozraké prohlédnout několik zařízení a jednu kamerovou lupu, na které mě zaujalo její ovládání.

Tuto lupu totiž nebylo třeba držet v ruce nad papírem. Stačilo ji na papír pouze položit. Kamera lupy byla zapuštěna několik centimetrů do těla zařízení a díky této konstrukci byla zajištěna stálá vzdálenost kamery od snímaného textu, a nebylo tak třeba hlídat ohniskovou vzdálenost a přeostřovat.

Obrázek 6.1. Kamerová lupa ležící na papíře [13]

Podobnou jednoduchost a praktičnost používání jsem chtěl vytvořit i při používání své aplikace na telefonu. Objektiv tohoto zařízení byl schopen zaostřit i na předmět vzdálený pouhé 3 cm. Tuto vlastnost bohužel objektivy mobilních telefonů nemají. Vzdálenost, na kterou jsou schopny zaostřit, je přibližně dvojnásobná.

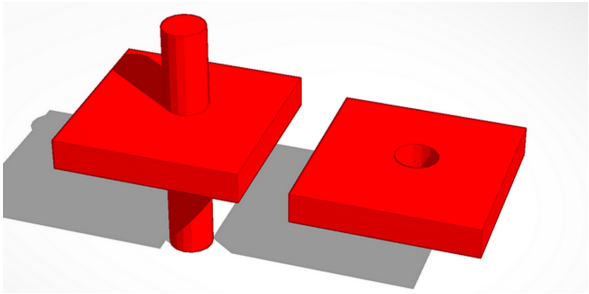
6.1 Tvorba stojánku na telefon

Abych dosáhl podobně jednoduchého používání aplikace jako u prodávaných kamerových lup, navrhl jsem stojánek. Díky tomuto stojánku bude telefon v nejnižší možné výšce nad papírem, v níž je ještě možné zaostřit. Protože vzdálenost telefonu od textu bude konstantní, nebude třeba přeostřovat. Stojánek bude vyroben na 3D tiskárně.

K návrhu stojánku jsem použil Tinkercad [1]). Jedná se o webový nástroj určený k tvorbě 3D modelů. Objekt se tvoří skládáním několika základních těles. Tělasa se

[1]) `https://www.tinkercad.com`

mohou skládat v pozitivním nebo negativním smyslu. Sečteme-li například kvádr a válec pozitivně, vytvoříme tak kvádr na hřídelce. Pokud ovšem stejná tělesa sečteme ve stavu, kdy válec bude negativní, vytvoříme tak kvádr s dírou o rozměrech zmíněného válce (viz. obrázek 6.2).



Obrázek 6.2. Vlevo je pozitivně sečten kvádr a válec, vpravo negativně

Pomocí těchto principů jsem vytvořil model stojánku na telefon. Do horní části jsem umístil otvory pro možnost připojení napájecího kabelu a ovládání bočních tlačítek telefonu.

Tento stojánek samozřejmě není univerzální, záleží na rozměrech telefonu, případně na umístění tlačítek a konektorů. Důležitá je také výška telefonu, která přímo souvisí se vzdáleností fotoaparátu od snímaného textu. Tento údaj, tedy minimální vzdálenost, na niž je telefon schopen zaostřit, výrobce přímo neuvádí. U všech telefonů, na kterých jsem aplikaci testoval, se ukázala hodnota 7 cm jako dostatečná.

Obrázek 6.3. Model stojánku pro Samsung Galaxy Core 2

Obrázek 6.4. Výsledek 3D tisku stojánku na Samsung Galaxy Core 2

6.2 Stojánek v praxi

Na obrázku je vidět výsledný distanční stojánek s telefonem Samsung Galaxy Trend Plus. Obrázek můžete porovnat s předlohou tohoto řešení pomocí kamerové lupy na obrázku 6.1.

Mnou navrhnuté řešení je plně funkční a myslím, že velmi zjednodušuje práci s aplikací, díky čemu by se její skutečná užitná hodnota mohla ještě více přiblížit prodávaným kamerovým lupám.

Obrázek 6.5. Stojánek s telefonem v akci, porovnejte s předlohou na obrázku 6.1

Díky tomu, že se 3D tisk stává v posledních letech stále běžnějším a dostupnějším, neměl by být problém nechat si vyrobit stojánek prakticky kdekoli na světě. Nakonec i samotná webová aplikace Thinkercad umožňuje nechat si vymodelovaný předmět vytisknout.

Kapitola 7
Testování aplikace

Aplikace byla postupně testována na třech skupinách. První skupinu tvořili zkušení uživatelé, jejichž přínos byl v objevení nestabilních míst aplikace. V druhé skupině byli zastoupeni nevidomí uživatelé, kteří se podíleli na návrhu ovládání aplikace. Třetí skupinou byli budoucí uživatelé aplikace, kteří byli seznámeni s ovládaním a poté aplikaci testovali. Zpětnou vazbu o aplikaci jsem od nich získal díky předtestovému a potestovému dotazníku.

7.1 Alfa testování, testování stability aplikace

Testování vytvořené aplikace probíhalo v několika fázích. Nejprve pouze na lidech v mém okolí za účelem objevení programových chyb, které by způsobovaly pády a nestabilitu aplikace.

Na tomto místě bych rád ocenil nápaditost svých přátel, zejména spolubydlících, u nichž se ukázala neočekávaná vynalézavost při hledání programových chyb. Během testování si aplikace a hlavně telefon sám prošly skutečným peklem a i díky tomu se podařilo objevit spoustu nedostatků způsobujících často „zamrznutí“ či zhroucení aplikace.

Většina těchto chyb byla způsobena prací ve více vláknech a jejich nedostatečnou synchronizací, což bylo záhy napraveno. Vylepšeno bylo také chování aplikace po zamknutí displeje s cílem co nejvíce šetřit baterii telefonu.

7.2 Beta testování, úprava ovládání

Další fází bylo testování na skupině zkušených nevidomých či slabozrakých uživatelů z organizace SONS. Jelikož se tito lidé přímo nepodíleli na vývoji, lze tuto fázi označit jako beta testování. Tito uživatelé mají mnoho zkušeností s testováním vyvíjených aplikací a díky tomu měli velmi cenné připomínky k chování a ovládání aplikace. Cílem bylo optimalizovat ovládání aplikace tak, aby bylo co nejvíce intuitivní a jednoduché.

Nejprve proběhla intenzivní e-mailová komunikace, z které vznikl celý koncept ovládacího menu, jež je až za živým náhledem z kamery telefonu. Jedná se o velmi jednoduché a přehledné řešení.V případě dalšího rozvoje aplikace bude doplnění dalších funkcí navíc velmi snadné. Stačí je do tohoto menu začlenit.

Další požadavek byl, aby bylo možné si pořízené fotografie ukládat a později prohlížet. Z tohoto důvodu jsou všechny pořízené fotografie ukládány do adresáře „Pictures“, jenž je v operačním systému standardním místem pro ukládání fotografií a pro jakoukoli aplikaci k tomu určenou je tedy snadné tyto fotografie načíst.

Po připomínkách SONSu byl také upraven zvukový projev aplikace. Původní verze, jež hlásila každou interakci s uživatelem (např. přiblíženo, oddáleno, zaostřeno), byla dle slov jednoho z beta testrů „ukecaná“. Na tyto výhrady jsem reagoval upravením zvukového projevu aplikace následovně:

- přiblížení a oddálení obrazu je signalizováno pouze ťuknutím, uživatel totiž vidí, zda se obraz přiblížil, či oddálil
- místo vyslovování „zaostřeno“ udělá aplikace pouze jakési „cvaknutí“, jež je typické hlavně pro skutečné fotoaparáty, které zaostřují mechanickým posunem optické soustavy fotoaparátu

Po úpravě výše zmíněného dostal SONS telefon s předinstalovanou aplikací a distanční stojánek k samotnému otestování. V reakci na připomínky bylo doplněno několik funkcí. První doplnění se týkalo ohlášení toho, že je aplikace aktivní. K tomu po startu poslouží hláška „připraveno“. Za druhé bylo doplněno periodické hlášení toho, že aplikace stále pracuje během hledání textu. Dále bylo pozměněno ovládání skrz menu. Po výběru inverze barev či rozsvícení blesku se okamžitě přejde zpět k živému náhledu s již aktivní změnou místo toho, aby se zůstávalo v menu a čekalo na případnou další akci.

7.3 Testování na běžných uživatelích

První kontakt se skutečnými potenciálními uživateli aplikace zažila během konference INSPO 2015 [1]). Jedná se o konferenci zaměřenou na technologie pro osoby se specifickými potřebami.

Během této konference prošel telefon s nainstalovanou aplikací rukama mnoha nevidomých i slabozrakých uživatelů. Zvláště samotná idea se dočkala velmi kladných ohlasů. Prakticky každý slabozraký uživatel byl po krátké instruktáži schopen aplikaci používat a to v rámečku i bez něho.

Všechny též velmi překvapila možnost nechat si rozpoznat text pomocí OCR, což je funkce, kterou většina prodávaných kamerových lup nedisponuje. V tomto ohledu má aplikace ještě spoustu prostoru pro zlepšení (viz. kapitola 8.2), na která uživatelé velmi rychle přišli a upozorňovali. Ovšem v případě, kdy se drželi mých pokynů, se ukázalo, že jsou schopni i tuto část využít.

Další část testování byla provedana opět v SONSu, tentokrát na jejich zákaznících, tedy opět běžných uživatelích.

7.3.1 Předtestový dotazník

Úkolem předtestového dotazníku bylo zjistit obecné informace o uživateli, jeho postižení a úrovni jeho zkušeností s dotykovými mobilními telefony a zařízeními pro přibližování textu a obrazu (viz. příloha A).

7.3.2 Potestový dotazník

Naopak úkolem potestového dotazníku bylo získat zpětnou vazbu bezprostředně po prvním seznámení s aplikací (viz. příloha B). Stejný dotazník bude uživatelům zaslán ještě řádově po dvou měsících používání aplikace. Sám totiž předpokládám, že během delší doby na testování se může objevit více nedostatků. Zároveň je možné, že si někteří uživatelé na ovládání a chování aplikace zvyknou.

7.4 Vyhodnocení výsledků

Z výsledků dotazníků vyplývá (viz. příloha C), že téměř všichni uživatelé již mají zkušenosti s dotykovými telefony, což je pochopitelné, protože telefony jiných konstrukcí se

[1]) `http://inspo.cz/`

přestávají vyrábět. Všichni uživatelé, kteří se setkali s dotykovým telefonem, mají také s jeho ovládáním nějaké potíže.

Ukázalo se, že uživatelé se zbytky zraku nějakou lupu používají, nejčastěji klasickou skleněnou a to víckrát denně. Tu by tedy skutečně mohla zastoupit aplikace v telefonu, kterou bude mít uživatel neustále na dosah.

Z potestového dotazníku vyplynulo (viz. příloha D), že uživatelé samotnou existenci aplikace hodnotí kladně, ovšem zejména v rozpoznávání textu jsou ještě rezervy a prostor pro zlepšení (viz. kapitola 8.2).

Kapitola 8
Závěr

8.1 Výsledek práce

Cílem práce bylo seznámit se s technologiemi pro přibližování textu, detekcí textu a převedení textu do hlasového výstupu pro potřeby nevidomých či slabozrakých lidí a navrhnout aplikaci pro přibližování textu s jednoduchým ovládáním, také implementovat algoritmus pro rozpoznávání textu a výslednou aplikaci otestovat na vzorku alespoň pěti uživatelů.

V rámci práce jsem vytvořil aplikaci pro mobilní operační systém Android, která pomocí jednoduchých gest umožňuje slabozrakým uživatelům přiblížit text či fotografie. Aplikace dále poskytuje režim se zobrazením inverzních barev či přečtení textu v obraze pomocí implementovaného algoritmu pro rozeznávání textu. Výsledná aplikace je zaměřena na slabozraké uživatele, ne na uživatele nevidomé.

Během práce na aplikaci jsem se inspiroval provedením kamerových lup, jež se moje aplikace snaží napodobit, a vytvořil distanční stojánek pro telefon, jenž je možné vytisknout na 3D tiskárně. Tento stojánek zásadně zjednodušší práci s aplikací na telefonu, protože uživatel díky němu nemusí držet telefon v ruce. Tím se eliminuje řada otřesů a zvýší se komfort při používání.

Návrh ovládání aplikace jsem diskutoval s členy organizace SONS (Sjednocená organizace nevidomých a slabozrakých ČR). Z této diskuze vznikl návrh ovládání a mnoho dalších cenných připomínek týkajících se chování a návyků nevidomých a slabozrakých uživatelů. Aplikace byla otestována na konferenci INSPO 2015, kde vyovolala pozornost mezi návštěvníky. Během testování na reálných potenciálních uživatelích jsem zíkal celkem pozitivní zpětnou vazbu, většina výtek se týkala převážně zlepšení algoritmu pro rozeznávání textu.

Mezi hlavní výhody svého řešení přibližování textu pomocí mobilního telefonu považuji jeho cenovou dostupnost. Cena mobilního telefonu s dostatečně kvalitním fotoaparátem a přisvětlovací diodou se pohybuje okolo 2000 Kč, prodávané kamerové lupy se až na výjimky prodávají za více než 10000 Kč.

Za další výhodu považuji fakt, že telefon má uživatel vždy po ruce a nemusí se starat o další zařízení. Bohužel tento fakt mohu označit i za nebezpečnou vlastnost, bude-li uživatel dlouho používat aplikaci, hrozí riziko, že se telefon vybije a nebude tak uživateli k dispozici v krizové situaci. Eliminace této skutečnosti je na uživateli. Pomoci mohou externí baterie či uživatelova předvídavost a plánování.

8.2 Další rozvoj

Přestože vytvořená aplikace Lupa je plně funkční, je zde několik možností vylepšení a alšího rozvoje.

Jak již bylo zmíněno, aplikace obsahuje režim, v němž ukazuje obraz v inverzních barvách. Po vzoru prodávaných kamerových lup by bylo dobré implementovat ještě jiné

různobarevné obarvení textu a pozadí. Vnímání barevného spektra se u slabozrakých uživatelů liší a mnohým z nich by toto mohlo pomoci v jednoduší čitelnosti textu.

Rozpoznávání textu v mé aplikaci vyžaduje, aby řádky textu byly rovnoběžné s některou hranou displeje. Místo toho by bylo lepší implementovat algoritmus, který by našel orientaci řádek nebo by v lepším případě hledal text v reálných scénách. Tento algoritmus by vracel obrázky (řádky) obsahující text.

Opravdu skvělé by bylo implementovat algoritmus, který by umožnil text bez problému přečíst i nevidomému uživateli. Tento algoritmus by musel v reálném čase prozkoumávat scénu a v ní detekovat text, na který by uživatele upozornil. Umím si představit, že by toto mohlo fungovat například u nápisů na dveřích či pouličních cedulích.

V případě rozpoznávání textu na celých stránkách by pravděpodobně bylo nejlepší uživatele provést nafocením několika fotografií, z nichž by se poskládala výsledná stránka, kde by se následně hledal text. Obě tyto úlohy jsou ovšem opravdu náročné.

Literatura

[1] *Spektra v.d.n.*
http://www.spektravox.cz/.

[2] Mgr. Radek Pavlíček RNDr. Hana Bubeníčková, Ing. Bc. Petr Karásek. *Kompenzační pomůcky pro uživatele se zrakovým postižením*. 2012.
http://www.centrumpronevidome.cz/doc/kompenzacni-pomucky.pdf.

[3] Karel Zimmermann. *Detekce a rozpoznávání znaků v obraze a videosekvencích*. Diplomová práce, 2004.
http://cmp.felk.cvut.cz/~zimmerk/lpd/diploma_thesis_zimm_text.pdf.

[4] L. Neumann a J. Matas. *Real-time scene text localization and recognition*. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Provi- dence, RI, USA: 2012.
http://dx.doi.org/10.1109/CVPR.2012.6248097.

[5] Martin Milichovský. *Mobilní aplikace pro vyhledávání a rozpoznání textu v obrazech reálných scén*. Diplomová práce, 2015.
https://dip.felk.cvut.cz/browse/details.php?f=F3&d=K13136&y=2015&a=milicmar&t=dipl.

[6] *Tesseract OCR* .
https://code.google.com/p/tesseract-ocr/.

[7] AIM Inc. *Optical Character Recognition*. 2000.
http://www.aimglobal.org/technologies/othertechnologies/ocr.pdf.

[8] *KNFB Reader*.
http://www.knfbreader.com.

[9] *Android developer*.
http://developer.android.com.

[10] E.P.Glinert R. Kline. *Improving GUI accessibility for people with low vision, Proceeding CHI '95 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

[11] Tony Gentry Al Copolillo. *Low Vision Intervention: Decision Making for Acquiring and Integrating Assistive Technology, International Handbook of Occupational Therapy Interventions 2015*.

[12] *Sjednocená organizace nevidomých a labozrakých ČR*.
http://www.sons.cz.

[13] *UNIOPTIK*.
http://www.unioptik.cz/p778/diditalni-a-kamerove-lupy-kompenzacni-pomucky-pro-slabozrake/kamerova-lupa-504003/.

Příloha A
Předtestový dotazník

1. Která z uvedených možností odpovídá Vašemu handicapu ? (vyberte jednu nebo více možností)
   - úplná nebo praktická slepota
   - zbytky zraku
   - poruchy binokulárního/barevného vidění
   - slabozrakost
   - postižené obě oči
   - porucha zraku je trvalá
   - porucha zraku je krátkodobá nebo opakující se
2. Kolik je Vám let ? (vyberte jednu možnost)
   - 0 - 20
   - 21 - 40
   - 41 - 50
   - 51 a výše
3. Jak dlouho máte handicap ? (vyberte jednu možnost)
   - 1 rok a více
   - 5 a více let
   - 10 a více let
   - od narození
4. Máte zkušenosti s dotykovým telefonem ? (vyberte jednu možnost)
   - ano
   - ne
5. Jaké máte problémy s ovládáním telefonu ? (krátký popis toho, co Vám dělá velké problémy)

6. Používáte nějaký druh lupy k přiblížení obrazu či textu? (vyberte jednu nebo více možností)
   - klasickou optickou (skleněnou) lupu
   - přenosnou kamerovou lupu
   - stolní kamerovou lupu
   - kamerovou lupu připojitelnou k PC nebo TV (myš do ruky, nebo skládací konstrukce)
7. Při jaké činnosti lupu nejčastěji používáte?
   - v práci
   - při studiu

- ve volném čase

8. Jak často lupu používáte?
   - 1 a vícekrát za den
   - několikrát do týdne
   - několikrát do měsíce
   - nevyužívám - uveďte důvod
9. Přenášíte svoji lupu z místa na místo nebo ji máte stále v jednom objektu (např. jenom doma, nebo v práci)?
   - přenáším
   - nepřenáším
10. Jaké funkce u své kamerové lupy používáte?
    - změna barvy pozadí
    - ozvučení snímaného textu
    - jiné (prosím napište jaké)

Příloha B
Potestový dotazník

1. Přijde Vám aplikace Lupa v telefonu užitečná?
   - ano
   - ne
2. Ohodnoťe prosím od 1 do 5, jak moc si myslíte, že by bylo možné nahradit vaši současnou lupu aplikací v telefonu (1 plně, 5 absolutně ne)
3. Ohodnoťe prosím od 1 do 5 kvalitu přiblíženého obrazu (1 velmi kvalitní, 5 nekvalitní)
4. Ohodnoťe prosím od 1 do 5 úspěšnost rozeznání textu v obraze (1 vše správně, 5 vše špatně)
5. Usnadnil Vám používání aplikace Lupa stojánek, který udržoval telefon ve správné výšce nad textem?
   - ano
   - ne

Příloha C
Odpovědi na předtestový dotazník

Uživatel	Otázka 1	Otázka 2	Otázka 3	Otázka 4	Otázka 5
1	c	b	b	b	žádné
2	b	b	b	a	psaní sms zpráv
3	b	c	d	a	celkově ovládat telefon
4	b	c	d	b	s obsluhou telefonu
5	b	c	c	b	vytáčení i psaní zpráv

Tabulka C.1. Odpovědi na předtestový dotazník 1/2

Uživatel	Otázka 6	Otázka 7	Otázka 8	Otázka 9	Otázka 10
1	a	a	c	a	-
2	a	c	a	a	-
3	b	ac	a	a	a
4	bc	abc	a	a	ab
5	a	c	c	b	-

Tabulka C.2. Odpovědi na předtestový dotazník 2/2

Příloha D
Odpovědi na potestový dotazník

Uživatel	Otázka 1	Otázka 2	Otázka 3	Otázka 4	Otázka 5
1	ano	2	3	4	ano
2	ano	2	2	2	ano
3	ano	3	3	3	ano
4	ano	2	3	3	ano
5	ano	2	2	4	ano

Tabulka D.3. Odpovědi na potestový dotazník

Příloha E
Zkratky a symboly

E.1 Zkratky

Seznam zkratek použitých v práci.

Px	Pixel, zkratka anglických slop picture element, neboli obrazový prvek
MPx	Mega pixel, milion pixelů.
API	Application Programming Interface
ADT	Android Developer Tools
SDK	Software Developer Kit
SONS	Sjednocená organizace nevidomých a slabozrakých České republiky
OCR	Optical Character Recognition - optické rozpoznávání znaků
3D	trojdimenzionální nebo trojrozměrný, označuje prostor jenž je možné popsat třemi rozměry
JAVA	objektově orientovaný programovací jazyk, též americký slengový výraz pro kávu po němž má tento jazyk svůj název
TTS	Text to speech, převod textu na mluvené slovo
SPZ	státní poznávací značka

Příloha F
Obsah přiloženého CD

/Bakalářská práce	Tato práce ve formátu PDF
/Obrázky	Obrázky použité v této práci
/Program	Zdrojové kódy programu
/Přílohy	Přílohy práce
/3D Model	Navrhnuté stojánky ve formátu *.stl