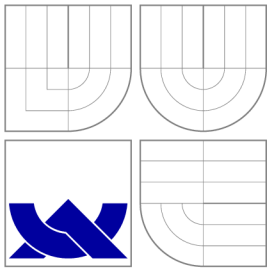
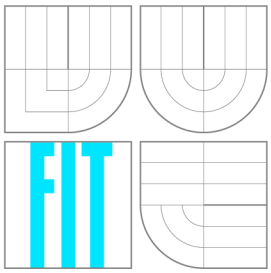


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ TLUMOČNÍK PRO ANDROID

MOBILE INTERPRETER FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VLADIMÍR HOMOLA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2013

Abstrakt

Tato bakalářská práce pojednává o vývoji aplikace Mobilní tlumočník, přičemž se zaměřuje na návrh vhodného uživatelského rozhraní. Cílem je vytvořit aplikaci s takovým rozhraním, se kterým budou uživatele schopni pracovat efektivně a s radostí. První část práce obsahuje shrnutí poznatků získaných studiem této problematiky. Po definování budoucího uživatele a řešeného problému je zde popsán návrh systému a jeho rozhraní následovaný popisem implementace a uživatelských testů.

Abstract

This bachelor thesis describes implementation of the Mobile interpreter application with focus on its user interface. The goal is to create an application with such interface with which the users will be able to work effectively and with pleasure. The first part contains a summary of the knowledge learned during study of this problem. After that is definition of the future user, situations in which he used it and design of system and its interface. Description of implementation and user testing is in the last part.

Klíčová slova

Android, OCR, uživatelské rozhraní, rozpoznání textu, uživatelské testy, překlad, Tesseract

Keywords

Android, OCR, user interface, text recognition, usability testing, translation, Tesseract

Citace

Vladimír Homola: Mobilní tlumočník pro Android, bakalářská práce, Brno, FIT VUT v Brně, 2013

Mobilní tlumočník pro Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana, PhD. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vladimír Homola
15. května 2013

Poděkování

Rád bych poděkoval vedoucímu mé práce panu Ing. Vítězslavu Beranovi, Ph.D., za jeho vstřícný a přátelský přístup a za vedení této práce.

Obsah

Kapitola 1

Úvod

V dnešní době jsme svědky stále většího nasazení mobilních technologií. Výkon těchto zařízení neustále roste a stejně tak rostou možnosti jejich použití. Doby, kdy se mobilní telefony používaly pouze k telefonování a psaní SMS, jsou již dávno pryč, čemuž odpovídá i skutečnost, že s mobilním telefonem v ruce nyní trávíme denně v průměru o jednu hodinu a dvacet minut déle než v roce 1999[1].

Jedna z takových činností u které můžeme vidět vývoj sledující právě výpočetní možnosti kapesních zařízení je překlad textu do jiného jazyka. V dobách ne tak dávno minulých byl takřka jedinou pomůckou cestovatele při překládání knižní slovník. Nyní je však výkon mobilních zařízení na takové úrovni, že tuto činnost provádíme pomocí nich a to dokonce bez nutnosti daný text ručně přepisovat.

Cílem této práce je navrhnout právě takovou aplikaci a její uživatelské rozhraní. Aplikace bude schopna přeložit text, který uživatel vyfotografuje a označí. Jedna z věcí, podle které bude uživatel tuto aplikaci posuzovat je uživatelské rozhraní, kterému proto budeme při návrhu věnovat největší pozornost.

Druhá kapitola této práce obsahuje souhrn znalostí získaných studiem rozpoznávání textu, jeho překladu a také vývojem aplikací pro Android, především pak jejich rozhraní spolu s uživatelskými testy.

Další část popisuje analýzu uživatele aplikace a situací, při kterých bude aplikaci používat. Na základě této analýzy je navržena funkčnost aplikace tak, aby pomáhala řešit situace, při kterých uživatel potřebuje provádět překlad. Je zde také popsán návrh dvou variant uživatelského rozhraní a metoda překladu textu do jiného jazyka. Dále jsou sestaveny uživatelské testy, které budou provedeny na obou rozhraních s cílem zjistit, které bude použitelnější.

Ve čtvrté kapitole je popsaná implementace aplikace, provedení testů a jejich vyhodnocení.

[1] Teens are spending more time consuming media, on mobile devices http://articles.washingtonpost.com/2013-03-13/news/37675597_1_teens-cellphones-video-games.

Kapitola 2

Studium

Tato kapitola obsahuje souhrn znalostí získaných při studiu. Je zde obsažen popis principů rozpoznávání textu v obraze a strojního překladu. Dále jsou zde informace o systému Android využité při další práci a také souhrn zásad tvorby uživatelského rozhraní pro tuto platformu. V další části jsou popsány poznatky studia testování aplikací a uživatelských testů. V poslední kapitole je uvedeno pár příkladů stávajícího řešení mobilního překladu.

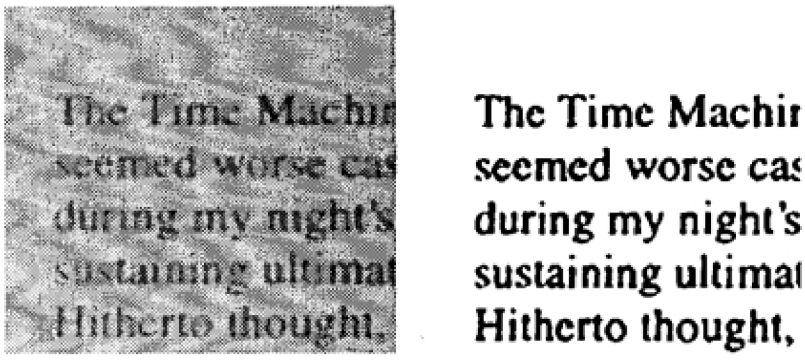
2.1 Rozpoznávání textu, OCR

OCR neboli Optical Character Recognition je metoda optického rozpoznávání textu z jeho obrazu. Pomocí OCR můžeme převádět texty z tištěných textů do elektronické podoby a posléze je strojně zpracovávat na počítači. Tento problém se řeší již dlouhou dobu a celý tento vývoj je zmapován v [13], kde se například dočteme že již v roce 1929 byl v Německu vydán na OCR patent.

Na začátku celého procesu OCR je samozřejmě pořízení obrazu textu. To může být provedeno naskenováním, či jako v našem případě vyfocením. Ještě před samotným rozpoznáváním znaků se na obraze provedou úpravy vedoucí k lepším výsledkům rozpoznání. Šum lze odstranit pomocí lineárních či nelineárních filtrů. Dále lze upravit natočení obrazu, jeho perspektivu či rozlišení. Obraz je také nutno binarizovat, čili provést na něm prahování tak, aby vznikl obraz složený jen z čistě bílých a černých bodů. Při prahování je však obtížné určit správnou hodnotu prahu pro celý obraz, proto se dnes používají metody adaptivního prahování, při kterých se práh určuje pro různé části dokumentu zvlášť. Existují i metody prahování jako [14], které byly vyvinuty přímo pro zlepšení výsledků OCR na kamerou pořízených snímcích textu. Na obrázku 2.1 vidíme jak se metoda BST vypořádá s prahováním nekvalitního obrazu.

V tomto obraze se pak vyhledávají kandidátní znaky, což obvykle bývají spojité obrazce. Z těchto obrazců se následně extrahují příznaky podle kterých se určí o jaký znak se jedná. Znak se například pomocí mřížky rozdělí na části, u nichž se zkoumají histogramy. Další metoda je založená na počtu průsečíků písmena s předem určenými vektory umístěnými v políčku písmena. Může se však také vyhodnocovat počet děr ve znaku či vzdálenosti hran od políčka písmene. Takto získané příznaky se následně pomocí některého z typů klasifikátoru klasifikují do skupin znaků.

Nepříjemnou situací je, pokud se prahováním slije více znaků dohromady, nebo když se jeden znak rozpadne na více částí tak, jak je znázorněno na obrázku 2.2. V těchto případech se musí znaky rozpojovat, respektive slepovat a při každém pokusu provést klasifikaci. Pak



Obrázek 2.1: Ukázka prahování metodou BST, originální obraz (vlevo), obraz po prahování (vpravo), převzato z [14].

se vyhodnocuje, která varianta přinesla kvalitnější rozpoznání, a ta se následně použije.

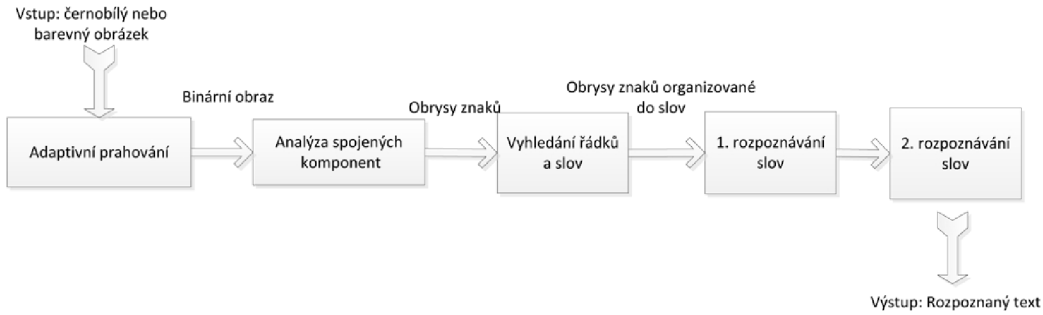
Obrázek 2.2: Ukázka znaků rozpadlých při prahování (vlevo) a více znaků spojených dohromady (vpravo)

V současné době existuje celá řada komplexních programů zaměřených na převod textu. Jejich přehled a cenové srovnání je k dispozici například na [4]. Pokud chceme použít některý z OCR enginů pro vývoj nové aplikace, musíme se zaměřit na ty, které nabízejí software development kit. Seznam OCR enginů i s porovnáním jejich vlastností nalezneme na [5]. Z tohoto seznamu jsem pro svou práci vybral Tesseract, neboť je jako jeden z mála volně k dispozici a dá se s ním pracovat i v Androidu. Aby jsme mohli s Tesseractem pracovat, je nejdříve nutné nastudovat princip jeho činnosti.

Tesseract

V Tesseractu není zahrnuta segmentace obrazu na oblasti s textem, tento systém vyvinulo HP odděleně pro své ostatní produkty a proto není nabízen spolu s Tesseractem. Tím, že byl tento systém v době vývoje tesseractu již k dispozici, je jako vstup pro analýzu uvažován již přímo obraz textu. Tesseract přijímá na vstup jak barevné, tak i černobíle obrázky. Zjednodušený průběh celého zpracování je ilustrován na obrázku 2.4.

Obrázek 2.3: Průběh polygonové aproximace, převzato z [9].



Obrázek 2.4: Průběh zpracování Tesseractu, převzato z [9].

Vstupní obraz nejprve projde adaptivním prahováním, jehož výstupem je binární obraz. V tomto obraze se následně vyhledávají spojité části, u kterých se aproximují jejich obrysy pomocí polygonů tak, jak vidíme na obrázku 2.3. Touto aproximací se potlačí šum a nepotřebné informace.

Tyto obrysy se posléze organizují do slov a slova se umisťují do řádků. Samotný proces rozpoznávání slov probíhá ve dvou iteracích. Tesseract používá adaptivní klasifikátor, který při průchodu textem zařazuje rozpoznaná slova do své trénovací množiny. Z tohoto důvodu mají slova umístěná dále v textu větší šanci být správně rozpoznána. Tato nesymetrie se vyrovná v druhém průchodu, kdy již lépe natrénovaný klasifikátor prochází text znova a rozpozná i slova neúspěšně analyzovaná v průchodu prvním.

Kvalita rozpoznání

Pokud chceme zjistit přesnost Tesseractu, můžeme se podívat na výsledky soutěže 4th UNLV annual test of OCR accuracy[10] konané v roce 1995, které se Tesseract účastnil pod názvem HP Labs OCR a umístil se mezi tři nejlepší. Od tohoto roku se však kód Tesseractu značně změnil a výsledky jsou ještě lepší. Verze 2.0 vydaná v roce 2007 dosahuje na stejné testovací sadě o 7.31% lepší výsledky v rozpoznání písmen a zlepšení 5.39% v rozpoznání slov. V současné době je Tesseract dostupný ke stažení ve verzi 3.02, přičemž kvalita rozpoznání by oproti verzi 2.0 měla ještě vzrůst.

2.2 Jazykový překlad

Metodě překladu prováděné na nějakém stroji, jako je například počítač říkáme strojový překlad. Metody strojového překladu se stále ještě vyvíjejí. Vývoj se snaží pokrýt překlady běžného jazyka jako například text novin, lidská konverzace či dopis. Naopak překlady abstraktního použití jazyka či dokonce uměleckých děl je úplně mimo rozsah této domény, neboť ty vyžadují dokonalou znalost jazyka a tvůrčí schopnost jeho použití. Dosavadní vývoj strojového překladu je popsán v dokumentu [7]. V této kapitole shrneme některé používané metody.

V dnešní době době tyto systémy dosahují úspěchů na poli přibližného překladu, překladu s post-editací člověkem, či překladů s úzkým lingvistickým zaměřením. Obecně lze říci, že se náročnost překladu odvíjí od podobnosti zdrojového a cílového jazyka, jejich slovech, gramatice a větné skladbě.

Existují dva základní přístupy jak sestavit systém překladače zohledňujícího gramatiku jazyka:

- První možností je ručně definovat gramatiky obou jazyků a vazby mezi nimi. Je třeba vytvořit systém pravidel, podle kterých jsou věty analyzovány, překládány a znovu sestavovány. Vytvoření těchto pravidel vyžaduje množství odborné práce jazykových vědců.

- Druhou možností je shromáždění obrovského množství textu vždy v obou jazycích. Systém pak v těchto překladech vyhledává podobnosti a kontextové závislosti a další překlady realizuje na základě těchto dat. Tento systém používá i překladač Google, který využívá svůj katalog vícejazyčných webových stránek. Kvalita výsledného překladu pak závisí od kvality překladu trénovacích dat.

Nejjednodužší překladače gramatiku nezohledňují a překládají postupně jednotlivá slova. Pořadí těchto slov v překladu zůstává stejné jako bylo ve zdrojovém textu, což může vést ke gramatické nesprávnosti. Zůstává na uživateli, aby si přeložená slova sám zorganizoval. Může se zde vyskytnout překlad jednoho slova pomocí více slov nebo také, že některá slova při překladu zanikají.

2.3 Platforma Android

Pod názvem Android se skrývá softwarový balíček obsahující operační systém, middleware a klíčové aplikace. Android byl původně vyvíjen firmou Android inc. než jí v roce 2005 koupila společnost Google. První zařízení s tímto operačním systémem přišlo o tři roky později. Byl to mobilní telefon T-Mobile G1 s verzí systému 1.0. V této kapitole jsou popsány některé z vlastností systému Android, které budou využity při implementaci aplikace.

SQLite

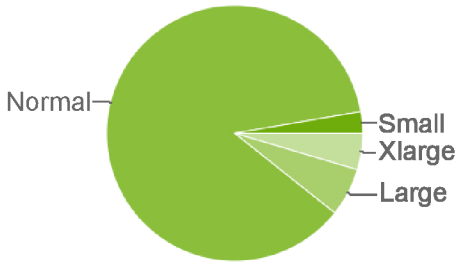
V systému Android je všem aplikacím dostupný databázový systém SQLite. SQLite se v mobilních telefonech s oblibou používá pro svou rychlost, paměťovou nenáročnost a také protože implementuje většinu standardu SQL92. Systém je napsán v jazyce C ve formě knihovny. Pokud chce aplikace tento systém používat, stačí pouze přilinkovat SQLite knihovnu. Celá databáze je umístěna v jediném souboru a tak může být lehce sdílena mezi různými zařízeními. SQLite databáze je typeles, což znamená do jakéhokoliv sloupce můžeme uložit jakékoliv data. Výjimku tvoří sloupec `INTIGER PRIMARY KEY` který vyžaduje 32-bitový signed integer.

Statistiky zařízení

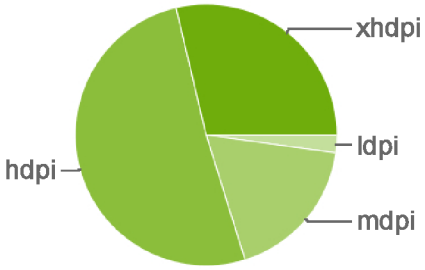
Jak již bylo uvedeno výše, Android využívá celá škála různých zařízení. V našem případě nás u těchto zařízení zajímají zejména vlastnosti jejich displejů, abychom věděli jak velikou pozornost máme jednotlivým typům displejů věnovat. Tomuto tématu se věnují stránky [1], kde se nacházejí průběžně aktualizované statistiky Android zařízení.

Velikosti obrazovek zařízení jsou generalizovány do čtyř kategorii: `small`, `normal`, `large` a `extra large` (více v kapitole 2.4).Graf na obrázku 2.5 ukazuje procentuální zastoupení jednotlivých kategorií.

Hustota pixelů na displeji je také generalizována do čtyř kategorí: `ldpi`, `mdpi`, `hdpi` a `xhdpi` (více v kapotile 2.4). Graf na obrázku ukazuje 2.6 procentuální zastoupení jednotlivých kategorií.



Obrázek 2.5: Procentuální zastoupení velikostí obrazovek v zařízeních Android, převzato z [1]



Obrázek 2.6: Procentuální zastoupení hustot rozlišení obrazovek v zařízeních Android, převzato z [1]

Pro návrh uživatelského rozhraní nás také zajímá spojení velikosti displeje s hustotou pixelů. Procentuální zastoupení jednotlivých kombinací je uvedeno v tabulce 2.1. Z tabulky vidíme, že nejpočetnější skupina zařízení je osazena obrazovkou velikosti normal s hustotou rozlišení hdpi.

		Hustota pixelů			
		ldpi	mdpi	hdpi	xhdpi
Velikost displeje	small	1,7%		0,1%	
	normal	0,4%	11,0%	50,1%	25,1%
	large	0,1%	2,4%		3,6%
	xlarge		4,6%		

Tabulka 2.1: Procentuální zastoupení jednotlivých typů obrazovek v zařízeních Android

2.4 Návrh uživatelského rozhraní pro OS Android

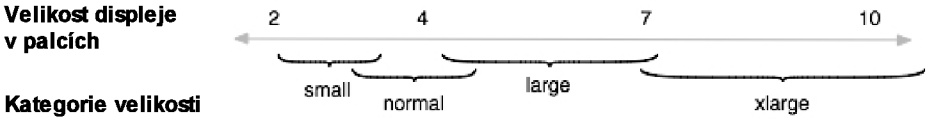
Operační systém Android běží na celé škále mobilních zařízení. Můžeme se setkat s přístroji s různou velikostí obrazovky, poměrem stran či hustotou pixelů. Platforma Android však nabízí API poskytující kontrolu nad všemi těmito aspekty tvorby rozhraní. Správný návrh pak musí počítat se všemi zařízeními a zajistit to, aby aplikace běžící na jakémkoliv z nich poskytovala stejný uživatelský komfort. To může ve výsledku znamenat, že se UI zobrazí pro všechny zařízení stejně, nebo také pro každé jinak. V této kapitole jsou popsány principy správného návrhu UI pro široké spektrum zařízení tak, jak je doporučeno v [2].

Sledované vlastnosti zařízení

Při návrhu UI musíme brát ohled především na typ obrazovky. Obrazovky se mohou lišit v těchto vlastnostech:

- **Velikost obrazovky**

  Fyzická velikost obrazovky vyjádřená délkou úhlopříčky displeje. Podle tohoto parametru můžeme zařízení dělit do čtyř kategorií: `small`, `normal`, `large` a `extra large`. Rozdělení obrazovek do skupin podle jejich velikosti je znázorněno na obrázku 2.7 Obecně se dá říct, že pro malé obrazovky je návrh složitější, neboť od uživatele vyžadují větší přesnost při interakci.



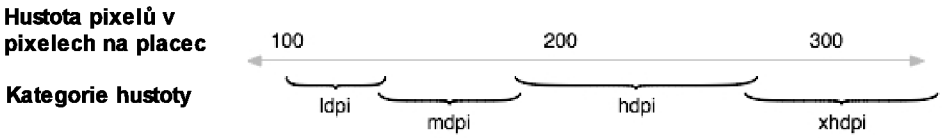
Obrázek 2.7: Rozdělení obrazovek do skupin podle jejich velikosti, převzato z [2].

- **Rozlišení displeje**

  Tento údaj udává počet pixelů, které je zařízení schopno zobrazit. Je to počet pixelů fyzicky obsažených na displeji.

- **Hustota pixelů (Screen density)**

  Udává počet pixelů obsažených v jednotce plochy displeje. Zřízení s nižší hustotou dokáže zobrazit méně informací oproti zařízení se stejnou velikostí a vyšší hustotou. Systém Android opět dělí zařízení do čtyř skupin: `ldpi`, `mdpi`, `hdpi` a `xhdpi`. Rozdělení obrazovek do skupin podle pustoty pixelů je znázorněno na obrázku 2.8



Obrázek 2.8: Rozdělení obrazovek do skupin podle hustoty pixelů, převzato z [2].

- **Orientace displeje**

  Jako další vlastnost může být chápána i orientace displeje. Orientace displeje znamená, v jaké poloze se zařízení nachází oproti uživateli. Uživatel může zařízení držet jak na výšku, tak na šířku, přičemž každá z těchto poloh nabízí jiné typy zobrazení a tím i jiné možnosti práce s nimi. Orientace displeje však není statická vlastnost a mění se v čase. Záleží na každém uživateli, jaký režim si vybere. Záleží také na typu situace. Někdy potřebujeme zabrat objekt na výšku, jindy zas na šířku. V návrhu proto zvážíme oba typy zobrazení a pamatujeme na to, že se mezi sebou můžou kdykoliv přepnout. Přepnutí zobrazení by proto nemělo narušit celkový dojem z UI ani jeho ovládání.

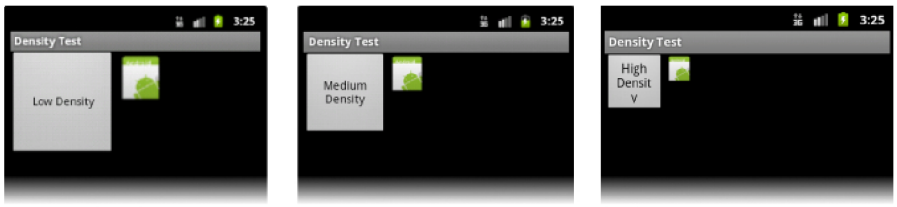
Density-independent pixel(dp)

Aby bylo možné navrhovat rozhraní najednou pro přístroje s různou hustotou pixelů používáme při návrhu virtuální pixel nezávislý na hustotě rozlišení. Jeden `dp` odpovídá jednomu fyzickému pixelu na obrazovce s hustotou 160`dpi`, jež je brána jako základ pro třídu hustoty medium. O přepočet velikosti pixelů založený na použité obrazovce se pak stará systém za běhu programu. Pro každou obrazovku se vypočítá počet pixelů odpovídající jednomu `dp` podle vzorce:

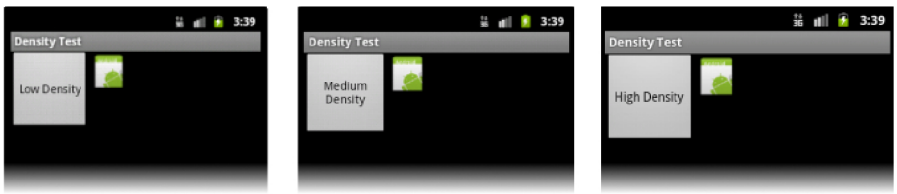
$$px = dp * (dpi/160)$$

Density independence

Aplikace je density independence pokud zobrazované elementy mají stejnou fyzickou velikost i na zařízeních s různou densitou displeje. V aplikaci, která není density independent se jednotlivé elementy UI zobrazují menší či větší v závislosti na densitě displeje. To způsobuje nekonzistentní vzhled aplikace a někdy i problémy s funkčností či ovládáním.



Obrázek 2.9: Ukázka aplikace bez podpory pro zařízení s různou hustotou pixelů, převzato z [2].



Obrázek 2.10: Ukázka aplikace s podporou pro zařízení s různou hustotou pixelů, převzato z [2].

Na obrázku 2.9 vidíme nekonzistentní UI aplikace, která není density independent. Velikost vykreslených entit byla v této aplikaci definována pomocí klasických pixelů `px`. Entity tak mají ve všech třech případech stejný počet pixelů, avšak jsou podle density displeje zobrazovány větší či menší. UI na obrázku 2.10 vypadá konzistentně i pro různé density displeje. Je toho dosaženo tak, že velikosti vykreslených entit jsou definovány pomocí `dp`. Můžeme si všimnout, že UI na zařízení s medium density vypadá v obou případech stejně, neboť medium density je použita jako referenční densita pro `dp`. Velikost všech vykreslovaných bitmapových entit je pak podle potřeby automaticky měněna operačním systém za chodu aplikace.

Zásady návrhu rozhraní

Jak bylo popsáno výše, systém je schopný se automaticky postarat o to, aby bylo rozhraní konzistentní napříč spektrem různých densit zařízení. Toho je dosaženo upravováním proporcí definovaného layoutu a s tím souvisejícími změnami velikostí vykreslovaných bitmapových entit. Toto však pro správný návrh rozhraní ještě nestačí. Pro správný návrh je kromě density independence doporučeno také provést následující kroky:

- Explicitně deklarovat pro jaké zařízení je aplikace určena

  V manifest souboru můžeme deklarovat pro jaké zařízení je aplikace určena. Tím zajistíme, že aplikace poběží jen na podporovaných zařízeních. Můžeme například určit, že aplikace vyžaduje dotykové ovládání, fotoaparát a není určená pro zařízení s malou obrazovkou.

- Pro každou velikost obrazovky poskytnou jiné schéma

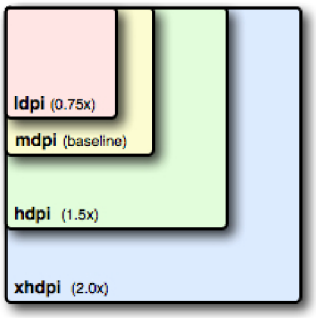
  Systém se automaticky snaží změnit rozložení rozhraní vždy podle použité obrazovky. Ve většině případů je toto dostačující řešení. Někdy se však může stát, že výsledné rozhraní nevypadá dobře a potřebuje provést změny. Například zjistíme, že se na malou obrazovku nevejdou všechny prvky rozhraní tak, aby s nimi bylo možné pracovat nebo můžeme chtít využít volný prostor na velké obrazovce k lepšímu rozmístění ovládacích prvků. V takovém případě můžeme pro každou z normovaných velikostí obrazovky vytvořit vlastní schéma uživatelského rozhraní. Systém pak při spuštění aplikace vybere to rozhraní, které nejlépe vyhovuje detekované velikosti obrazovky.

- Ošetřit orientaci zařízení

  Pokud dojde za běhu programu ke změně orientace zařízení, například z `portrait` na `landscape`, dojde k automatické úpravě schématu rozhrají tak, že se vykreslované prvky otočí dle orientace a zarovnají se na novou šířku, respektive výšku obrazovky. Taková úprava je však ve většině případů nepřípustná a proto musíme poskytnout schéma rozhraní zvlášť pro oba případy. Této situaci se můžeme vyhnou tím, že v manifest souboru určíme jakou orientaci displeje aplikace používá.

- Poskytnout různé bitmapy pro různé density

  Systém v základu přizpůsobuje velikost obrázků tím, že je sám podle potřeby zmenšuje nebo zvětšuje, to však může mít za následek zhoršenou kvalitu výsledných obrázků. Pro zajištění optimálního vzhledu obrázků musíme poskytnout pro každou kategorii density vlastní sadu obrázků. Poměry mezi velikostmi obrázků mezi jednotlivými kategoriemi `ldpi`, `mdpi`, `hdpi` a `xhdpi` jsou 3:4:6:8, tak jak je znázorněno na obrázku 2.11. Pokud systém nenalezne obrázek odpovídající příslušné densitě musí vybrat nejvhodnější kategorii a použít obrázek z ní. Například, pokud aplikace běží na systému s `ldpi` a tato kategorie nemá své vlastní zdroje, nevyužijí se pro resizing zdroje kategorie `mdpi`, které by musely být zmenšeny na 0.75%, ale použijí se zdroje kategorie `hdpi`, které zmenší svou velikost o polovinu.



Obrázek 2.11: Odstupňování velikosti obrázků pro jednotlivé hustoty pixelů, převzato z [2].

2.5 Testování uživatelského rozhraní

Spolu s vývojem mobilního tlumočníka budeme provádět dva druhy testování aplikace. Jednou formou testování je kognitivní průchod aplikací a druhou je testování aplikace na uživatelích neboli testy použitelnosti. V této kapitole shrneme poznatky získané studiem těchto dvou metod.

Kognitivní průchod

Následující informace jsou čerpány zejména z [11] a [3].

Kognitivní průchod je postup snažící se odhalit úskalí, která mohou potkat uživatele při prvním styku s produktem. Nejprve je potřeba definovat cíle, kterých bude chtít budoucí uživatel s produktem dosáhnout a následně sestavit posloupnost akcí, jejichž provedení k cíli vede. Samotný kognitivní průchod pak probíhá tak, že testující osoba provádí tyto akce a snaží se při tom vžít do pohledu uživatele, který je provádí poprvé. Každou provedenou akci si musí zdůvodnit a zaznamenat každou případnou nejasnost.

Kognitivní průchod se dá provádět již ve stádiu návrhu rozhraní, přičemž samotná funkcionalita ještě nemusí být implementována. Důležité je však, aby všechny prvky rozhraní již měly definovanou funkci a chování. Dle [3] může být tato metoda u menších projektů prováděna neformálně samotným návrhářem.

Uživatelské testy

Během vývoje budeme provádět tzv. studii použitelnosti aplikace. Díky této studii budeme schopni porozumět skutečným potřebám uživatele a zjistíme jeho reakce a odezvy na chování aplikace. Tyto nově získané poznatky pak použijeme pro úpravu rozhraní nebo i celé aplikace tak, aby uživatelům co nejvíce vyhovovala. Účelem tohoto procesu je, aby byli uživalé s mobilním tlumočníkem spokojeni, používali ho bez komplikací a námahy, a hlavně aby jej používat chtěli.

Abychom mohli tuto studii provést, musíme nejprve porozumět termínu použitelnost. V knize [6] je uveden výňatek definice použitelnosti ze standardu ISO 9241-11, podle kterého je použitelnost definována jako míra, do které může být produkt používán specifikovaným uživatelem k dosažení specifikovaných cílů ve specifickém prostředí a to s efektivností, účinností a spokojeností.

Pokud chceme aplikovat tuto poněkud formální definici, musíme před samotnou studií definovat tyto čtyři prvky: uživatele, cíle aplikace a podmínky za jakých bude aplikace

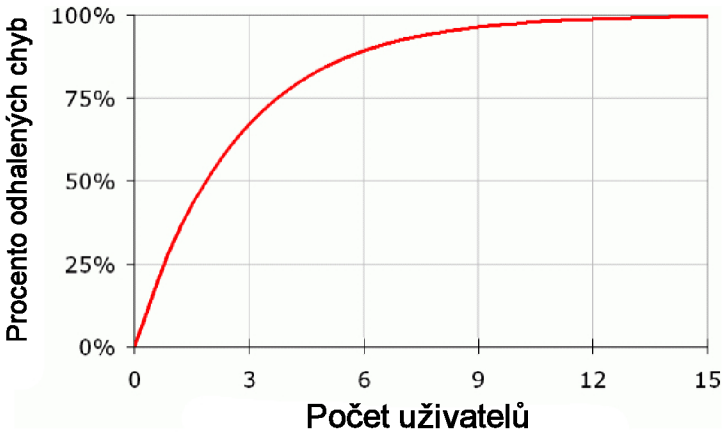
používaná. Studie se pak zaměří na tři hlavní aspekty, kterými jsou efektivnost, účinnost a spokojenost uživatele.

Podobně je použitelnost popsána v první kapitole knihy [12] a to jako soubor těchto hodnot: užitečnost, výkonnost, efektivnost, pochopitelnost a uspokojení. Tyto pojmy jsou v této knize dále popsány.

Abychom mohli testovat, musíme si obstarat množinu testovacích uživatelů. Sedmá kapitola knihy [12] popisuje postupy vhodné k výběru uživatelů a místa vhodného k testování. Uživatelé mohou být specifikování do více skupin podle určitých vlastností. Těmito vlastnostmi mohou být například věk či zkušenost. V závislosti na počtu výsledných skupin závisí i celkový počet uživatelů potřebných k testu. V článku [8] autor uvádí, že při definici pouze jedné skupiny stačí ke kvalitním výsledkům jen 5 uživatelů. Dále zde popisuje vztah mezi počtem uživatelů a procentem odhalených problémů. Procento odhalených chyb během testů s **n** uživateli se dá vypočítat pomocí vzorce:

$$N * (1 - (1 - L)^n)$$

Kde N je totální počet chyb v návrhu a L je procento chyb odhalených při testu jednoho uživatele. Typická hodnota L je 31%. Na obrázku 2.12 je tato závislost vykreslana v grafu. Autor článku však doporučuje při použití dvou skupin vybrat tři až čtyři zástupce z každé skupiny a při třech či více skupinách pak použít po třech uživatelích z každé.



Obrázek 2.12: Vztah mezi počtem uživatelů a procentem odhalených chyb, převzato z [8]

V druhé kapitole knihy [6] se pojednává o vhodných místech k provádění testů. Nejkvalitnější výsledky přináší testování v prostředí, ve kterém se bude testovaný produkt používat. Takovéto testy jsou však extrémně finančně i časově náročné a proto se testy provádějí v tzv. testovací laboratoři. Jako neformální testovací laboratoř může být použita jakákoliv místnost, ve které nebudou testy rušeny. Důležité však je, aby testy probíhaly pro všechny uživatele stejně. Proto je nutná formalizace všech testů.

2.6 Stávající řešení

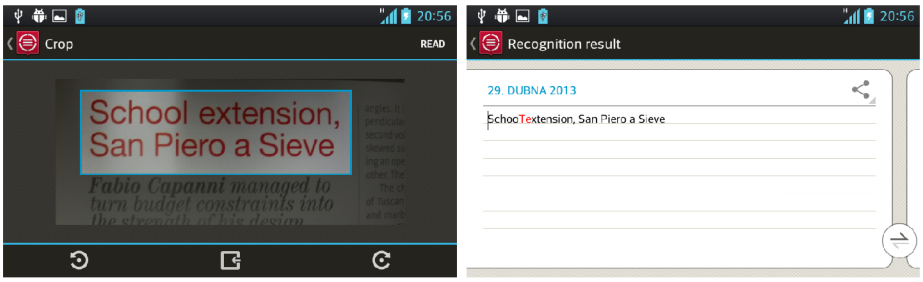
Pokud se zaměříme na současná řešení využívající k překladu mobilní telefon, najdeme na trhu velkou škálu aplikací. Ze všech aplikací vybírám dva zástupce high-end třídy.

ABBYY TextGrabber + Translator

Tato aplikace vyvíjená firmou ABBYY[1] nabízí překlad do 61 jazyků. Některé jazyky překládá s využitím služby Google Translate a u některých provádí pouze slovní překlad pomocí ABBYY Lingvo. K provedení OCR využívá ABBYY Mobile OCR Engine. Jako přidanou hodnotu aplikace mužeme považovat propojení se sociálními sítěmi či mailovým klientem.

Po pořízení fotografie se výběr textu provádí ručně obdélníkovým výběrem. Obrazovka výběru textu je zobrazena na obrázku 2.13.

Jakmile je proveden pokus o překlad, nedá se již navrátit k opakovanému výběru textu. Rozpoznaný text je zobrazen na samostatné obrazovce 2.13 a je ho možné ještě ručně upravit. Jakmile je text přepsán správně, může se provést online překlad. Jakmile je jednou proveden pokus o překlad, nedá se již navrátit k opakovanému výběru textu.



Obrázek 2.13: ABBY: Zobrazení výběru textu (v levo) a rozpoznaného textu (v pravo).

Google Translate

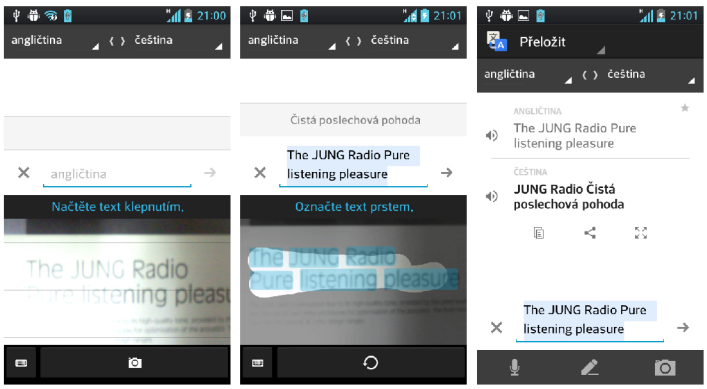
Druhým zástupcem je velice populární aplikace Google Translate[2]. Možnosti této aplikace zdaleka přesahují rozsah funkcí Mobilního tlumočníka, například můžeme uvést překlad z řeči, automatické rozpoznání jazyka, rozpoznávání ručně psaného textu, režim konverzace či hlasový výstup.

Kvůli komplexitě celé aplikace a tím i uživatelského rozhraní se zaměříme pouze na tu část, kde dochází k pořízení snímku a následného překladu.

Náhled kamery je pouze přes polovinu obrazovky avšak práce s kamerou je stéle velice příjemná. Po vyfocení se snímek automaticky analyzuje a označí se detekovaný text. Při výběru slov k překladu pak stačí jednoduše dotknout se daného slova a jeho text se automaticky přesune do textového pole nad snímkem. Obsah textového pole se může kdykoliv editovat a vždy automaticky proběhne překlad. Pokud jsme s vyfoceným textem spokojeni, můžeme se po stisknutí šipky přesunout na další obrazovku, kde znovu vidíme kompletní překlad.

[1] Viz http://www.abbyy.com.

[2] Viz https://play.google.com/store/apps/details?id=com.google.android.apps.translate&hl=cs.



Obrázek 2.14: Google translate: Pořízení fotografie textu (vlevo), Výběr textu (uprostřed), Překlad (vpravo).

Kapitola 3

Návrh řešení

V této kapitole jsou definováni budoucí uživatelé Mobilního tlumočníka spolu se situacemi, při kterých jej budou používat. Je zde popsán návrh koncepce překladu a uložení slovníkových dat. Dále je zde popsán návrh uživatelských rozhraní **A** a **B**, jež se liší v postupu vedoucímu k překladu. Rozhraní **A** by mělo být více konzervativní a koncept jednotlivých prvků by tak měl být podobný jako u základních aplikací telefonu, načež rozhraní **B** je řešeno jiným způsobem a snaží se urychlit překlad spojením více akcí dohromady. K těmto rozhraním jsou zde navrženy uživatelské testy.

3.1 Analýza problému

V této kapitole detailně analyzujeme potenciální uživatele, situace při kterých budou aplikaci používat a také vymezíme hranice řešeného problému. Musíme definovat problematické situace a nabídnout k nim adekvátní řešení. Z množiny všech možných situací, při kterých potřebuje uživatel provádět překlad však vybíráme jen ty, pro které je aplikace tohoto typu schopna poskytnout účinnou pomoc. Jsme si vědomi, že tato aplikace nepomůže uživateli se všemi problémy, ale chceme dosáhnout toho, že když aplikace nějaké řešení nabídne, bude toto řešení dostačující a užitečné. Poznatky získané touto analýzou budou dále uplatněny při návrhu aplikace a jejího uživatelského rozhraní.

Analýza uživatele

Typickým uživatelem může být turista na dovolené v cizí zemi vlastnící mobilní telefon platformy Android, který z nějakého důvodu nemá připojení k internetu. Tím důvodem většinou bývá cena datových služeb nebo špatné pokrytí signálem. Dá se předpokládat, že se tento turista vybaví kromě této aplikace také klasickou slovníkovou aplikací, tudíž bude vždy moci zvolit tu variantu, která je podle něj pro konkrétní úkol vhodnější. Naší snahou pak musí být to, aby když bude zapotřebí přeložit jakýkoli vyfotitelný text, byla zvolena právě naše aplikace. Kromě aplikace v mobilním telefonu v současné době nevidím žádnou jinou reálnou konkurenci, neboť turisté se snaží pokud možno vyhnout se dalším věcem, které je potřeba vždy nosit s sebou. Existuje samozřejmě skupina lidí, kteří sebou stále vozí slovník nebo notebook, ale zřejmě jej nevezmou sebou na pláž, do restaurace či na výlet. Použijí ho spíše jen na hotelovém pokoji, kde zase s větší pravděpodobností mají internetové připojení a mohou tak využít mnohem pokročilejší internetové překladače.

Uživatelem však nemusí být pouze turista. Může jím být i člověk, u kterého se ani nepředpokládá, že by musel řešit problémy překladu textu v cizím jazyce. Tento člověk si

aplikaci nainstaloval jen jako další součást výbavy svého telefonu pro případ, že by se mohla jednou hodit. Taková situace nastane až po dlouhé době, pokud vůbec. Uživatel pak musí být schopen aplikaci použít rychle a kdykoli i bez toho aniž by si ještě pamatoval jak se ní pracovalo.

Uživatelem naopak nebude někdo, kdo se problematikou překladů zabývá na vyšší úrovni. Pro výkon své práce ji jistě nebude používat tlumočník, překladatel ani průvodce. Stejně tak ji nevyužije ani ten, kdo bude v cizí zemi pobývat delší dobu, například student na výměnném pobytu, člověk pracující v cizině či přistěhovalec. Tito lidé si jsou schopni v cizím prostředím vytvořit jisté zázemí, jehož součástí bude zajisté i připojení k datovým službám, počítač či slovník.

Analýza situací

Nyní, když již víme, kdo bude mobilní tlumočník používat, můžeme se zaměřit na analýzu situací se kterými mohou přijít do styku a při nichž budou moci tlumočník použít. Budeme se snažit u každé ze situací odhalit potřeby uživatele a nabídnout způsoby a možnosti řešení.

Typickou situací, se kterou se již setkal snad každý turista je cizojazyčné menu v restauraci. V této situaci je nejdůležitější zjistit z názvu pokrmu alespoň to, z čeho se jídlo skládá. Zajímá nás, jestli je maso kuřecí nebo hovězí, jestli je jako příloha rýže či brambory. Z tohoto důvodu nám stačí jednoduchý překlad jednotlivých slov, přičemž překlad abstraktních jmen pokrmů jako například Katův šleh pro nás nemá téměř žádnou výpovědní hodnotu, ale stejně tak je tomu i v jazyce, který známe. Přestože jsou názvy pokrmů jako položky jídelního lístku relativně krátkými texty, nepředpokládáme že by uživatel fotil celou stránku jídelního lístku najednou, ale vyfotí vždy jen jedno, maximálně pár jídel pod sebou, viz 3.1. Tento předpoklad staví na tom, že v restauraci většinou sedíme u stolu a proto by vyfocení celého jídelního lístu najednou znamenalo nepříjemné a nápadné zvedání telefonu tak, aby fotoaparát pokryl celý list.

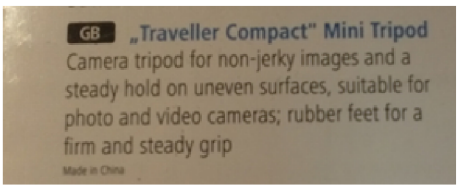
Obrázek 3.1: Předpokládaný vstup překladu jídla.

Dalším častým problémem je nutnost porozumění cizojazyčnému nápisu nebo ceduli. Nevíme, jestli se jedná o nápis informačního charakteru, jestli je to zákaz či výstraha nebo jestli jde jen o reklamu. Tyto nápisy bývají často krátké a výstižné. V takovémto případě je důležité, aby překlad alespoň vystihl charakter sdělení. Dá se předpokládat, že tato situace nastává hlavně někde na na silnici či ve městě. Turista tak nemusí mít v ruchu města dostatek času ani prostoru, aby se zbytečně dlouho zdržoval prací s tlumočníkem. Aplikace proto musí nabízet jednoduché a rychlé rozhraní se snadným ovládáním a to pokud možno i jednou rukou.

Turista se také denně setkává s velkým množstvím reklamních či jiných sdělení ve formě tištěných letáků, plakátů, kartiček či vizitek. Tyto formy textu většinou nebývají příliš dlouhé, proto se dá předpokládat, že se uživateli podaří vyfotit celý leták najednou. Fotit pak může leták položený někde na stole, také však může daný leták držet v ruce a telefon pak ovládat jen jednou rukou. Aby byly reklamní texty více nápadné a lépe tak upoutávaly naší pozornost, bývají často doplněny nejrůznějšími fotografiemi či grafikou, která pak kazí možnosti segmentace fotografie na oblasti s textem a zhoršují tak kvalitu rozpoznaného

textu. Pak může přijít vhod možnost překládat text z jednoho snímku na více částí tak, že uživatel postupně označí a přeloží partie letáku na kterých není rušivá grafika. Proto se později musí dát k vyfocenému snímku znova vrátit.

Mobilní tlumočník také může být užitečný při nákupech, kdy si nejsme jisti daným zbožím. Potřebujeme pak podle textu na obalu zjistit o co se vlastně jedná, neboť né vždy je to na první pohled jasné. Pokud pomineme názvy výrobků ve smyslu obchodních značek, zajímají nás názvy jako mléko, pomerančový džus a podobně. Také nás ale mohou zajímat položky ve složení výrobku. V obou případech se však jedná o heslovité názvy k jejichž pochopení postačuje jednoduchý slovní překlad. Ukázka příkladu, kdy se může hodit překlad textu na obalu výrobku je na obrázku 3.2.



Obrázek 3.2: Ukázka textu na obalu výrobku.

Z této analýzy vyplývá, že uživatel bude mobilního tlumočníka používat typicky s delšími časovými přestávkami (intervaly mezi dovolenou, od instalace k prvnímu použití) a proto musí být její rozhraní navrženo tak, aby mu seznámení se s ním zabralo co nejméně času a aby se k aplikaci byl vždy schopen vrátit bez nepříjemného vzpomínání jak se s ní kdysi pracovalo. Také musí být přizpůsobeno nárokům na práci v časovém shonu a pouze jednou rukou. Většina překládaných textů by rovněž měla být relativně krátká.

3.2 Rozhraní A

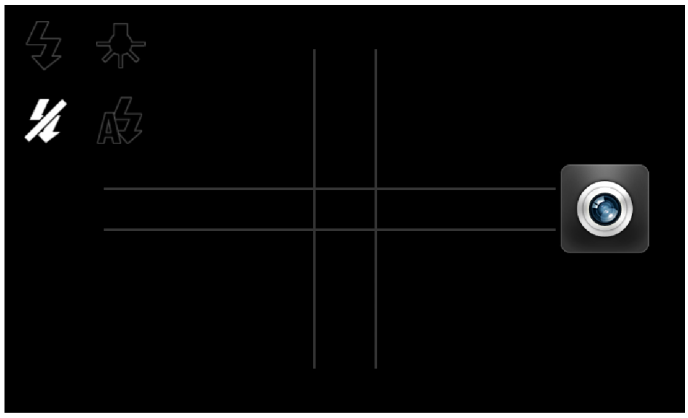
Rozhraní **A** umožňuje provést přeložení textu ve třech krátkých krocích a to pořízením fotografie, následnou selekcí textu a překladem tak jak vidíme na obrázku 3.3. Samotný překlad je pak spolu s rozpoznaným textem zobrazen na samostatné obrazovce, kde je samozřejmě možné ještě eventuálně špatně rozpoznaný text ručně opravit a provést aktualizaci překladu. Z překladu se lze také vrátit zpět k selekci textu na pořízeném snímku a v případě potřeby změnit oblast s označeným textem.

Obrázek 3.3: Diagram práce s rozhraním A

Návrh obrazovky fotoaparátu se snaží vycházet z již zažitých vlastností klasických fotoaparátových aplikací. Zaostření se provede dotykem na plochu náhledu kamery a o provádění ostření uživatele informuje zezelenalý záměrný kříž. Tlačítku spouště je dedikováno samostatné tlačítko, které je umístěno na okraji obrazovky a je tak dosažitelné prstem i při

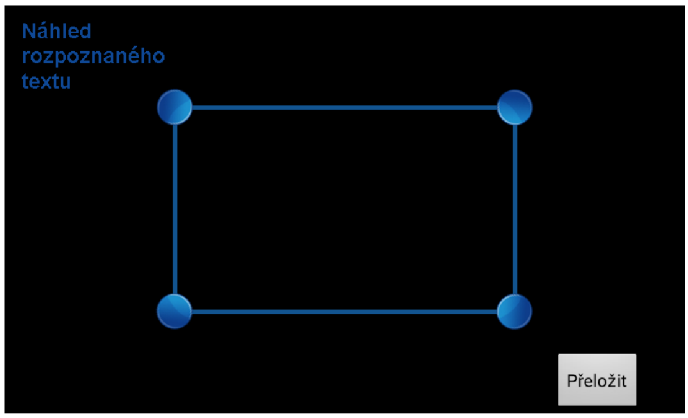
držení jednou rukou a to při všech polohách obrazovky. Při zmáčknutí tlačítka spouště se rovněž provede automatické zaostření. Připraveny jsou také čtyři funkce blesku a to: zapnutý, vypnutý, automatická a také funkce stálého přisvětlování diodou. Jak již bylo zmíněno, fotografování je možné při všech orientacích zařízení. Uživatele o tom přesvědčují tlačítka blesku, která se automaticky natočí podle orientace displeje.

Aby bylo možné kvalitně rozpoznat vyfotografovaný text, musí být text vyfotografován rovně. Proto je náhled kamery doplněn o záměrný kříž, podle kterého se uživatel může při focení orientovat. Na obrázku 3.4 můžete vidět návrh schématu náhledu fotoaparátu při orientaci `ladscape`.



Obrázek 3.4: Návrh fotoaparátu při orientaci landscape

Po vyfocení snímku přejde aplikace do režimu výběru textu. Obrazovku výběru textu vidíme na obrázku 3.5. Na displeji je zobrazen pořízený snímek překrytý obdélníkovým výběrovým polem. V této fázi musí uživatel roztáhnout toto pole přes oblast s textem, který chce přeložit. Vzhled a chování výběrového obdélníku je podobný zažitým vlastnostem podobných ovládacích prvků. Všechny čtyři strany jdou nezávisle roztahovat a celým výběrem lze posouvat. Pro lepší přehlednost je fotografie mimo výběr ztmavena, a proto vybraný obraz na obrazovce vyniká. Jakmile je požadovaný text označen, může se pomocí tlačítka přeložit přejít k obrazovce s překladem.



Obrázek 3.5: Obrazovka selekce textu spolu s průběžnými výsledky OCR.

Abychom minimalizovali počet zbytečných návratů z překladu zpět na selekci textu

kvůli špatně rozpoznanému textu, bylo by dobré, aby bylo ještě před provedením překladu, respektive stiskem tlačítka přeložit jasné, jaký text byl ve výběru fotografie rozpoznán.

Řešením je zobrazení výsledků OCR analýzy ještě na obrazovce selekce. OCR by mohlo být provedeno na přání uživatele stiskem speciálního tlačítka a podle výsledků by tak mohl výběr ještě upravit. Toto řešení vyžaduje po uživateli další a to možná opakovanou interakci s rozhraním, což mu může přijít zbytečné a nebude to chtít dělat.

Další variantou je neustálé provádění OCR v cyklu v novém vláknu aplikace. Na obrazovce by se tak průběžně zobrazoval text rozpoznaný z aktuálního výběru. Nevýhodou při takovémto zpracování může být časová prodleva mezi úpravou výběru a zobrazením prvních výsledků OCR z tohoto výběru. Tato prodleva je však u krátkých textů prakticky zanedbatelná.

Neustále měnící se text průběžných výsledků se může jevit celkově rušivě a proto je také vhodné omezit maximální délku zobrazovaného textu tak, aby zbytečně nezasahovala do výběrového pole. Proto se vždy budou zobrazovat maximálně tři řádky z rozpoznaného textu.

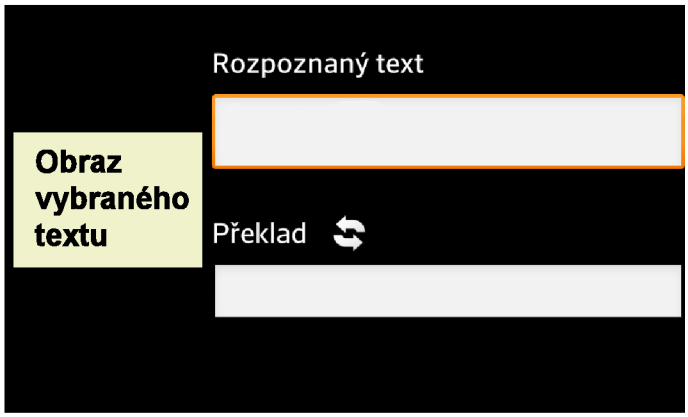
Neboť sami nedokážeme posoudit, která varianta bude uživatelsky příznivější, zahrneme tento bod do uživatelských testů a na základě ohlasů uživatelů vybereme ten lepší.

Předpokládáme-li, že uživatel nezná překládaná slova, bude pro něj orientace v rozpoznaném textu a jeho kontrola s textem na fotografii obtížná. K lepšímu přehledu o tom, co bylo správně rozpoznáno, by mohly přispět tzv. bounding boxy okolo slov. Bounding boxy se projevují jako tenké barevné čtverce kolem jednotlivých oblastí, které byly Tesseractem vyhodnoceny jako slovo, tak jak to vidíme na obrázku 3.6. Otázkou však zůstává, jestli si uživatel všimne souvislosti mezi umístěním boxů a kvalitou rozpoznaného textu a jestli ho tyto boxy nebudou při selekci textu rušit. Vhodnost zobrazení bounding boxů posoudíme na základě reakcí uživatelů při uživatelských testech.

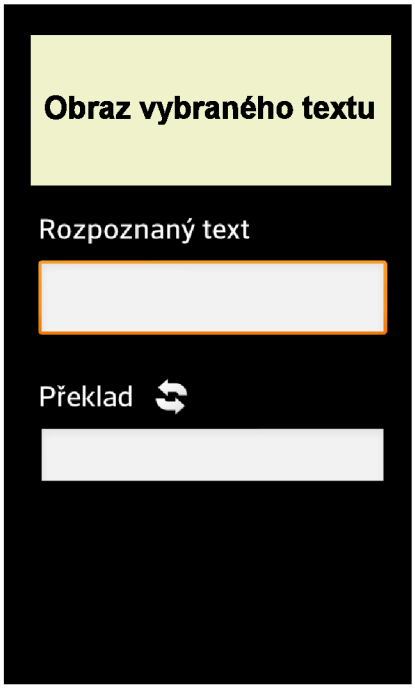
Obrázek 3.6: Návrh bounding boxu v rozhraní A

Obrazovka s výsledkem překladu je zobrazena na obrázku 3.8 a 3.7. Proporce této obrazovky závisí na orientaci zařízení, a to z důvodu efektivnějšího využití místa. Na této obrazovce vidíme zmenšeninu výřezu fotografie s textem a dvě textová pole. V jednom poli je text rozpoznaný z fotografie, čili text v cizím jazyce a v druhém poli je jeho překlad. Obsah obou polí lze dodatečně editovat a také provést opětovný překlad upraveného textu. Opětovný překlad se vyvolá stiskem tlačítka umístěného u pole s překladem. Při orientaci zařízení landscape se navíc zmenšenina výřezu fotografie pokud je její šířka mnohem větší než výška otočí o 90° tak, aby se náhled mohl roztáhnout na výšku, zabíral větší plochu a byl tak lépe čitelný. V pozadí celé obrazovky je pro efektivnější vzhled vidět tlumený

náhled kamery.



Obrázek 3.7: Vzhled obrazovky s překladem při orientaci landscape.

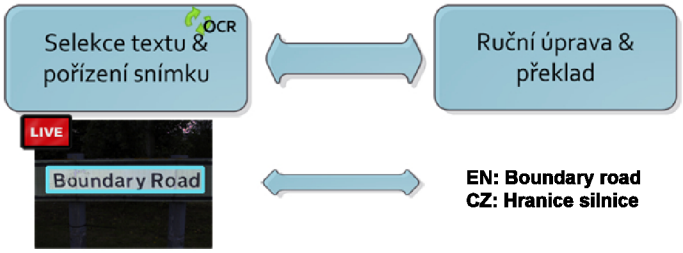


Obrázek 3.8: Vzhled obrazovky s překladem při orientaci portrait.

3.3 Rozhraní B

Toto rozhraní je navrženo tak, aby urychlilo celý proces překladu tím, že sloučí pořízení fotografie a selekci textu do jednoho kroku. Nyní je celý proces překladu rozdělen pouze do dvou částí jak vidíme na obrázku 3.9.

Selekce textu se provádí zároveň s pořizováním snímku. Toho je dosaženo vložením výběrového obdélníku do náhledu kamery, tak jak je vidět na obrázku 3.10. Výběrový obdélník je nyní pevně umístěn do středu, pohyb obdélníku je nyní nahrazen pohybem

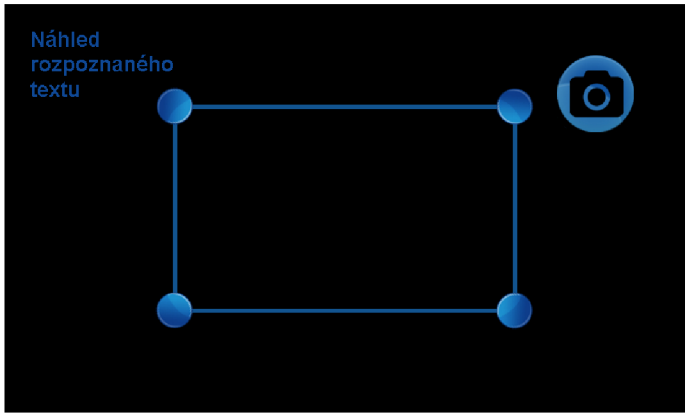


Obrázek 3.9: Diagram práce s rozhraním B

celého telefonu (zaměřením na text). Obraz mimo výběr je opět pro lepší orientaci ztmaven. Při změnách velikosti výběru se souběžně roztahují protilehlé strany to proto, aby výběr zůstal středově souměrný vůči středu náhledu.

Změnou oproti chování výběru z rozhraní **A** je také ovládání velikosti. Pro změnu velikosti se nyní není třeba trefit prstem na označené body v rozích obdélníku a pohybovat s nimi. Celá obrazovka je rozdělena na čtyři kvadranty, viz obrázek 3.11. Pohybem prstu v jakémkoliv kvadrantu hýbeme s obdélníkem tak, jako bychom jej roztahovali přes rohový bod v daném kvadrantu. Výhodou je pak snazší ovládání jednou rukou, není třeba se trefovat do bodu a také nám prst nezakrývá část selekce.

Toto rozhraní umožňuje práci pouze při orientaci landcape. Dále zde již není volba blesků, neboť jsem z osobní zkušenosti zjistil, že použití blesku v rozhraní **A** většinou vede k odleskům na textech a tím i k značné degradaci výsledku OCR.

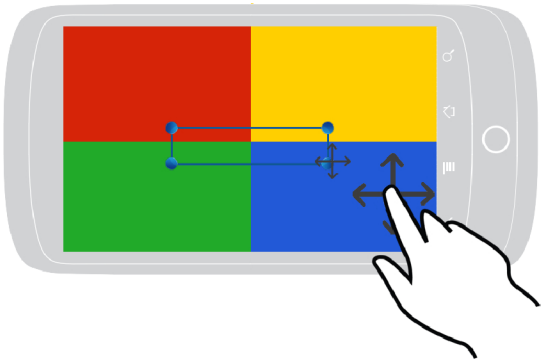


Obrázek 3.10: Náhled kamery spolu s výběrovým obdélníkem na rozhraní B.

Stejně jako tomu bylo v rozhraní **A** musíme uživatele nějak informovat, jaký text je ORC analýzou ve výběru rozpoznán. V tomto případě se analýza musí provádět souběžně s náhledem kamery a to v novém vlákně. Rozpoznaný text se zobrazuje v levém horním rohu náhledu. Rozpoznaný text se zobrazuje pouze v omezeném rozsahu tří řádků, to proto, aby nepřekážel v náhledu.

I v tomto případě může být zobrazovaný text pro větší přehled doplněn bounding boxy okolo rozpoznaných slov. U boundingboxů je nyní zvolen jiný vzhled než u rozhraní **A**, protože chceme vytvořit druhou variantu, kterou by uživatelé mohli srovnávat. Rozpoznaná slova jsou barevně zvýrazněna průsvitným obdélníkem, podobně jako text označený zvýrazňovačem. Návrh boundingboxů je vidět na obrázku 3.12.

Aby bylo toto zvýraznění užitečné však záleží na rychlosti OCR analýzy v závislosti



Obrázek 3.11: Ovládání výběrového obdélníku na rozhraní B.

na pohybu kamery. Pokud se bude kamera pohybovat rychleji než stačí OCR produkovat výsledky, budou bounding boxy umístěny na neodpovídajících místech. V rámci testů použitelnosti budeme chtít zjistit odezvu uživatelů na toto chování a jejich ohodnocení přínosu samotných bounding boxů.

Obrázek 3.12: Návrh bounding boxů v rozhraní B

Přechod k obrazovce překladu se může vyvolat buďto kliknutím do výběrové oblasti nebo na tlačítko označené fotoaparátem. V rámci testů použitelnosti bychom rádi zjistili, jestli budou uživatelé používat obě varianty, nebo jestli je lepší změnit akci při doteku do obdélníku za focus.

3.4 Uživatelské testy

Pří návrhu uživatelského rozhraní byly vytvořeny dva návrhy. Oba tyto návrhy musíme do uživatelských testů zahrnout, neboť právě na základě jejich výsledků odůvodníme zvolení finálního návrhu. Také se budeme snažit získat reakce uživatelů na sporné místa návrhu, abychom pro finální návrh použili správnou variantu. Naším cílem je sestavit plán a dokumentaci k testům v takové podobě, aby samotné testy mohly být provedeny osobou, která není zainteresovaná do vývoje aplikace.

Aplikace je určena širokému spektru uživatelů. Pro účely testů proto rozdělíme uživatele do skupin podle jejich zkušeností s chytrými telefony. Aplikace může být používána zkušenou osobou, která již má zkušenosti s podobnými aplikacemi, ale také někým, kdo takový telefon vlastní jen krátce. Uživatele proto rozdělíme do tří kategorií: nezkušený, běžný a zkušený uživatel. Každý uživatel před zařazením do procesů testování vyplní krátký dotazník, ve kterém odpoví na otázky týkající se jeho zkušeností s tímto typem mobilních telefonů. Pomocí dotazníku chceme zjistit, zda-li uživatel vlastní takovýto telefon, pokud ne, tak jestli s ním umí zacházet a také jestli někdy používal aplikace slovníkového typu. Po vyhodnocení odpovědí se uživatel zařadí do příslušné skupiny. Tento dotazník je k dispozici na přiloženém CD.

Podle doporučení ohledně počtu uživatelů z kapitoly 2.5 vybereme z každé skupiny tři uživatele. Celkem tedy pro testy potřebujeme získat devět osob.

Každý uživatel bude přizván ke dvěma testovacím sezením. V prvním sezení zjistíme jeho reakce při prvním kontaktu s aplikací, a při druhém již budeme sledovat uživatele, který již ví jak s aplikací pracovat. Mezi těmito sezeními musí uplynout časový interval minimálně čtrnáct dní. V rámci jednoho sezení budou testovány obě varianty rozhraní, a to v předem určeném pořadí. Aby se vykompenzoval vliv dojmu z prvního rozhraní, bude jedna polovina uživatelů začínat s rozhraním **A** a druhá s rozhraním **B**. Při druhém sezení se pak toto pořadí prohodí. Uživatelé tak budou moci obě rozhraní mezi sebou srovnávat.

Každé rozhraní je testováno dvěmi variantami úkolů. Při první variantě pracuje uživatel s aplikací bez jakýchkoliv omezení a může tak používat obě ruce. Druhá varianta se snaží simulovat prostředí, ve kterém může uživatel pracovat pouze jednou rukou. Cílem úkolu je přeložit zadaný text a úkol úspěšně končí, jakmile je uživatel s překladem spokojený. Každý úkol se skládá z překladu několika typů textu. Tyto texty jsou vybírány tak, aby reprezentovaly reálné případy použití. V rámci úkolu se tak k přeložení překládá reklamní sdělení v časopise, nápis na ceduli visící na zdi a jídlo z jídelního lístku. Zadání úkolů je pro oba typy rozhraní samozřejmě stejné, avšak liší se sledované hodnoty. Zatímco u rozhraní **A** můžeme sledovat dobu zabranou pořizováním fotografie a také dobu výběru textu, v rozhraní **B** se nám tyto dvě hodnoty slijí do jediné. Pro každý úkol je tedy vhodné vytvořit samostatný protokol popisující průběh testu a měřené hodnoty. Tyto protokoly jsou k dispozici na přiloženém CD.

Kromě zisku takto zpracovatelných dat budeme s každým uživatelem po skončení úkolů vyplňovat krátký dotazník. Tento dotazník sbírá uživatelovy subjektivní pocity. Na tyto otázky uživatel odpoví přiřazením hodnoty ze stupnice $1-5$. Odpovědi můžeme také zpracovat a porovnávat hodnoty mezi rozhraními. Pro doplnění celkové informace jsou zde ještě otázky na které uživatel odpoví slovně.

Po provedení prvního úkolu je vyplněn ještě doplňující dotazník, týkající se právě dojmů z prvního použití. Tyto dotazníky jsou k dispozici na přiloženém CD.

Další metodou jak získat povědomí o pocitech a názoru uživatele je společný komentovaný průchod. Po dokončení úkolů provedeme s uživatelem ještě jeden společný průchod aplikací při němž můžeme klást uživateli otázky týkající se rozhraní. Zejména chceme zjistit jestli jsou všechny prvky UI srozumitelné a jestli je uživatel pochopil právě tak, jak jsou navrženy. Dále také uživateli ukazujeme a diskutujeme s ním sporné body obou rozhraní. Ptáme se na funkci všech prvků rozhraní a uživatel nám popíše co dělají. Hledáme místa, které jsou pochopena sporně. Ptáme se, jestli by si některou část UI nepředstavil raději jinak nebo jaké řešení by bylo podle jeho názoru lepší. Při tomto průchodu má uživatel možnost plně vyjádřit své pocity a přání. Takto získané informace nám poslouží při případných úpravách rozhraní.

3.5 Překlad

Aplikace bude překládat text po jednotlivých slovech, nebude zohledňovat kontextové závislosti mezi slovy či větami. Výsledný překlad by tak měl vypadat stejně, jako kdyby uživatel použil klasický slovník a začal postupně po slovech text překládat.

Základem každého slovníku je dostatečná slovní zásoba. Tak jako při použití klasického slovníku se při zvyšování počtu obsažených výrazů zvyšuje pravděpodobnost nalezení hledaného slova. K realizaci je tedy potřeba získat seznam alespoň nejpoužívanějších slov daného jazyka a ke všem těmto slovům musíme přidat překlad do cílového jazyka. Jelikož překlad jednoho slova může být složen z více slov, například anglické slovo thermocouple má český ekvivalent termoelektrický článek, jdou takto vytvořené dvojice použít pouze pro překlad v jednou směru.

Slovníková data jsou uložena v databázi, která by měla zajišťovat dostatečnou rychlost vyhledávání. Každé slovo má uloženo své ID, text v původním jazyce a text překladu. Tato databáze je při instalaci přiložená k aplikaci, tak aby ji nebylo potřeba dodatečně stahovat.

Kapitola 4

Realizace

Tato kapitola popisuje postup implementace aplikace a průběh uživatelských testů spolu s jejich vyhodnocením. V kapitole 4.1 je popsána implementace jednotlivých částí systému a postup sestavení uživatelského rozhraní. Kapitola 4.2 shrnuje průběh uživatelských testů a vyhodnocuje jejich výsledky. Na závěr je pak vybráno vítězné rozhraní.

4.1 Implementace

Tato kapitola popisuje postup implementace aplikace podle návrhu z kapitoly 3. Aplikaci jsem vyvíjel v prostředí Eclipse, do kterého byl přidán ADT plugin, který umožňuje vytvářet Android projekty. Aby bylo možno pracovat s Androidem musíme mít také nainstalováno Android SDK, které poskytuje potřebné knihovny, nástroje a systémové obrazy potřebné pro emulátor Android zařízení. Kvůli použití tesseractu, který není napsán v Javě ale v C++ musíme použít také Android NDK, které aplikacím umožňuje používat knihovny psané v nativním kódu.

Rozhraní

Jednotlivé layouty rozhraní byly sestavovány přímo v Eclipse, kde ADT plugin nabízí grafický editor rozhraní.

Rozhraní **A** je složeno celkem ze tří layoutů (fotoaparát, výběr textu, překlad) a rozhraní **B** ze dvou(výběr textu, překlad).

Do layoutu lze vložit pouze potomky třídy `View`, proto musí všechny vytvořené zobrazované prvky tuto třídu dědit.

Ikony mají nastavený parametr velikosti na `wrap_content`, což znamená že se obrázky vykreslují v takové velikosti, ve které jsou uloženy. Neboť byla grafika vytvořena pro všechny skupiny hustot pixelů, viz příklad na obrázku 4.1, bude aplikace density independent.

Obrázek 4.1: Škála velikostí ikon pro jednotlivé kategorie hustoty pixelů.

Práce s kamerou

Aby bylo možné pracovat s kamerou, musí být do manifestu aplikace přidána žádost o povolení používání kamery. Také je nutno ověřit jestli zařízení obsahuje kameru. Pokud ano, můžeme se pokusit kameru otevřít.

Dále je potřeba vytvořit třídu, která bude zajišťovat zobrazení náhledu kamery. Za tímto účelem je vytvořená třída `Preview`, která dědí od `SurfaceView` a implementuje rozhraní `SurfaceHolder`. Neboť je třída potomkem `SurfaceView`, může se libovolně vkládat do layoutu rozhraní.

Tato třída je schopna zobrazovat obraz přicházející z fotoaparátu. Pokud navíc implementujeme rozhraní `Camera.PreviewCallback` je schopna zachytit jednotlivé snímky náhledu a zpracovávat je, což je využito u rozhraní **B**, kde je potřeba získávat výřezy náhledu pro rozpoznání textu.

Jednotlivé typy blesků se kameře nastavují jako její parametry. S bleskem typu torch, neboli trvalým přisvícením diodou, může u některých zařízení nastat problém projevující se zhasnutím diody těsně před dokončením pořízení fotografie.

K zaostření i pořízení snímku je potřeba předat kameře callback, který se spustí po dokončení dané akce. Callback pro pořízení fotografie obdrží data snímku, které se v tomto callbacku zpracují. Callback pro zaostření obdrží informaci, zda-li bylo zaostření úspěšné. Také zpracovávání snímků náhledu je prováděno pomocí volání callbacku pro každý nový snímek.

Kvůli náročnosti a době provádění OCR není možné provádět rozpoznávání textu na všech snímcích náhledu, ale musí se vždy počkat na dokončení právě prováděné analýzy a teprve pak zpracovat nový snímek.

Orientace zařízení

Návrh rozhraní A vyžaduje, aby bylo možno aplikaci používat při všech orientacích displeje. Normálně by se o přepínání orientace postaral systém automaticky tak, že aplikaci nataví layout pro danou orientaci. Při použití kamery však přepínání layoutů znamená i několik desetin vteřiny dlouhé přebliknutí náhledu kamery mezi zahozením starého náhledu a vytvořením nového pro jinou orientaci. Toto přebliknutí vypadá velice rušivě a z toho důvodu nemůžeme tento systém použít.

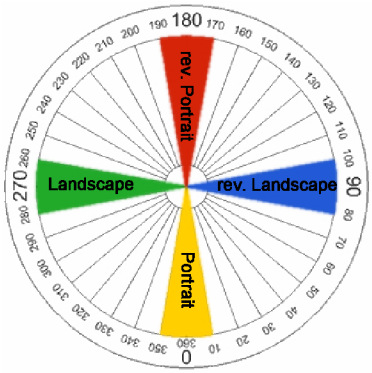
Aby jsme se tomu vyhnuli, ošetřujeme orientaci zařízení programově. K tomu je zapotřebí:

- Nastavit orientaci pevně na `landscape` a v manifestu uvést, že o změny orientace budeme starat sami.
- Sledovat data o náklonu z g-senzoru a podle nich se rozhodovat o aktuální orientaci.

Při zjistění změny orientace se vnitřně zaznamená nový stav a překreslí se orientace ikon blesku, přičemž náhled kamery běží stále nezměněn v režimu landscape a nemusí se proto vytvářet nový.

Mezi stupni náklonu zařízení, při kterých se provádí přepnutí orientace musí být rozestup, aby se nestalo, že uživatel drží telefon zrovna v hraničním úhlu a dvě sousední orientace se mezi sebou stále přepínají. Proto je pro každou orientaci stanoveno pouze pásmo 20°, ve kterém se orientace přepíná, tak jak je znázorněno na obrázku 4.2.

Při pořízení snímku je pak ještě zapotřebí správně přetočit fotografii, neboť operační systém si stále myslí, že byla pořízena v režimu landscape.



Obrázek 4.2: Znázornění pásem, při kterých se přepínají orientace zařízení.

Výběrový obdélník

Výběr textu se provádí metodou manuálního označení pomocí výběrového obdélníku. Tento nástroj musíme implementovat tak, aby šel použít v obou typech rozhraní. Vzhled obdélníku je u obou rozhraní stejný, avšak liší se jeho chování. Musíme proto oddělit část, která se zobrazuje od vnitřní logiky výběru.

O logiku výběru se stará třída `Selection`, ve které jsou uloženy souřadnice vrcholů výběru a která se stará o pohyb a změny velikosti.

Zobrazovaná část výběru je implementovaná jako potomek třídy `View`, kterou lze jednoduše umístit do schématu rozhraní. V této třídě je přepsána metoda `onDraw`, která vykresluje obraz na přidělené plátno. Na plátno můžeme vykreslovat jak základní geometrické obrazce tak i bitmapové obrázky. Tímto plátnem je v našem případě plocha celé obrazovky.

Podle návrhu vzhledu je metoda `onDraw` přepsaná tak, že je schopa:

- **Zobrazit čtyři krajní body** zadané jako bitmapy a propojit je přímkou tak, aby byl výběr ohraničen.
- **Ztmavit vnější část obdélníku.** Nevybraná část obdélníku je pro lepší vzhled ztmavená. To je provedeno překrytím obrazu černou barvou, u které je pomocí alfa kanálu nastavena průsvitnost.
- **Zobrazit boundingboxy**, které jsou produkovány tesseractem ve formě objektu `Pixa`. `Pixa` v sobě obsahuje seznam objektů `Box`, které uchovávají rozměry jednotlivých boundingboxů. Výběrový obdélník musí tento seznam uchovávat, aby věděl kde má vykreslovat. Spolu s tímto seznamem musí mít informaci, jestli jsou aktuální souřadnice boxů platné. Platnost aktuálních boxů skončí v momentě, kdy se obdélník posune nebo změní velikost. Souřadný systém boxů by tak byl posunut a ty by se zobrazovaly na nesprávných místech. Samozřejmě, že by šlo podle posunutí obdélníku dopočítat novou pozici boxů, ale ta by stejně nebyla platná, protože při novém výběru by nemusela být ta samá slova rozpoznána. Neplatné boxy je tudíž neradno zobrazovat, aby uživatele zbytečně nemátly. Boxy začnou být platné až s následující aktualizací od Tesseractu.

Uživatel ovládá výběr pomocí tahů prstem po obrazovce, proto tyto tahy musíme zachytávat a rozpoznávat. Každý potomek třídy `View` má na zachycování doteků obrazovky metodu `onTouchEvent`, která zachycuje událost doteku prstu. Aby bylo možné rozpoznávat

gesta jako například tažení prstem, přeposílají tyto události třídě `GestureListener` implementující rozhraní GestureDetector.OnGestureListener, které vyhodnocuje a ošetřuje gesta prstu.

Můžeme tak implementovat jeho metody, které jsou volány při výskytu daného gesta. Zajímá nás tedy především metoda `onDown` spouštěná při položení prstu na obrazovku a metoda `onScroll` spouštěná při rozpoznání tahu prstu. V případě doteku musíme zjistit kde přesně se nachází neboť od toho se pak odvíjí chování při jeho posunu na obrazovce. Toto vyhodnocení se odehrává ve třídě `Selection` a závisí na typu rozhraní.

U rozhraní **A** se podle souřadnic doteku vyhodnocuje, jestli byl prst umístěn v některém z rohových bodů výběru, uvnitř obdélníku nebo někde mimo něj. V metodě `onDown` se tedy zavolá metoda `setSector`, která to podle souřadnic doteku vyhodnotí a zaznamená. Při vyvolání gesta `scroll`, tedy pohybu prstu po obrazovce se zavolá metoda třídy podle původního umístění prstu hýbe buď konkrétním rohovým bodem nebo pokud byl dotek uvnitř obdélníku celým výběrem.

U rozhraní **B** je situace jednodušší, neboť je potřeba znát pouze kvadrant, ve kterém se prst obrazovky dotkl. Podle kvadrantu pak při pohybu prstu hýbeme rohovým bodem ležícím v daném kvadrantu. Aby však byla zachována souměrnost obdélníku vůči středu obrazovky, provádí se opačný pohyb i s protilehlým rohem.

Slovník

Slovník je pro oba typy rozhraní stejný a jeho výsledná implementace bude tedy použita v obou případech. V kapitole 3.5 je navrženo, aby byla slovníková data uložena v databázi a překlad probíhal formou dotazování této databáze. Na platformě Android se nabízí využití vestavěné databáze SQLite, která by měla splnit jak nároky na rychlost tak i na zacházení s databázovým souborem. Databázový soubor může být přiložen k aplikaci ve složce assets a při prvním spuštění uložen do složky telefonu `/data/data/PackageName/databases/`, kde by měly být databáze umístěné.

Pro práci s databází je vytvořena třída `DatabaseHandler`, dědící od `SQLiteOpenHelper`. Třída `SQLiteOpenHelper` zjednodušuje vytváření a otevírání databáze. V našem případě se však databáze nebude vytvářet, avšak kopírovat. Tím pádem se nevyužije metoda `onCreate`, použije se metoda, která pokud ještě databáze neexistuje vytvoří prázdnou databázi a zkopíruje do ní databázový soubor.

Dále je zapotřebí implementovat metodu, která z rozpoznaného textu extrahuje slova, jež se budou v databázi vyhledávat. Rozpoznaný text je rozdělen na slova podle bílých znaků. Kolem těchto slov však ještě může být interpunkce nebo jiné speciální znaky, se kterýma nemůžeme slovo vyhledávat. Tyto znaky se pomocí regulárního výrazu od slova odstraní a uloží, aby mohly být po překladu ke slovu znovu přidány. Takto upravené slovo se vyhledává v databázi. Pokud není slovo v databázi nalezeno, navrací se původní slovo. Výsledky překladu jsou pak opětovně skládány za sebe a doplněny o odebranou interpunkci a mezery.

Tesseract

Pro práci s Tesseractem je použit projekt tess-two[1], který je klon Tesseract Tools for Android. Tato varianta je vybrána na základě dobrých zkušeností ostatních uživatelů a s tím i spojeným množstvím informací.

[1] Viz https://github.com/rmtheis/tess-two.

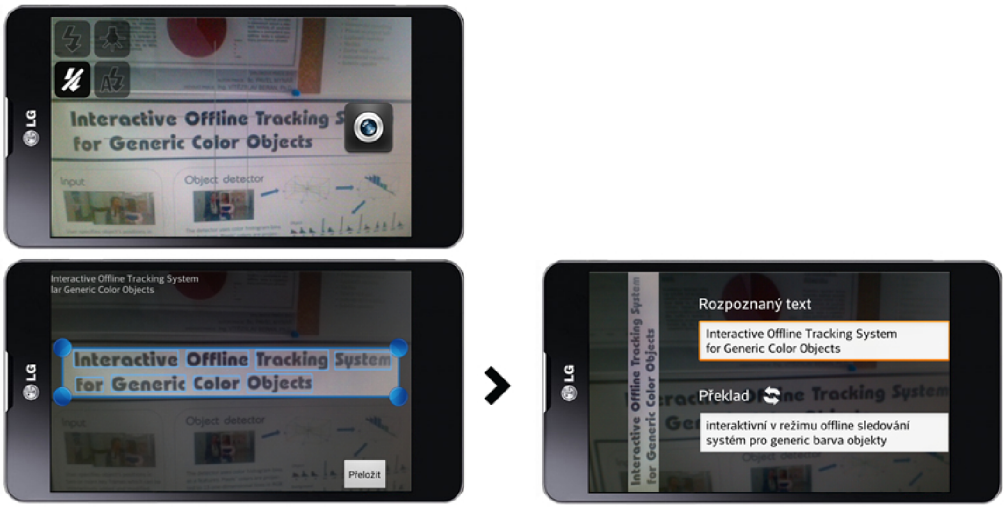
Tess-two se do Eclipse vkládá jako knihovna a poskytuje Java API pro přístup aplikace k nativnímu kódu Tesseractu a Leptonicy. V současné době dokáže Tess-two pracovat s verzí Tesseractu 3.02 avšak tato verze se zdá být nestabilní a při překladu často padá. Z tohoto důvodu použijeme starší verzi, ve které se využívá stabilnější verze 3.01. Po vytvoření objektu rozhraní `TessBaseAPI` a nastavení jazykových dat se dá rozpoznání textu provést pouhými dvěmi příkazy. Nastavení obrazu k rozpoznání `setImage(Bitmap)` a pak získání textu metodou `getUTF8Text()`. Po každém rozpoznání je také vhodné uvolnit rozpoznaná data pomocí `clear()`. Rozhraní umožňuje vrátit boundingboxy okolo rozpoznaných slov a to ve formě objektu `Pixa` po zavolání metody `getWords()`.

Záznam testů

Pro snazší vyhodnocování uživatelských testů byla vytvořena třída `TestLogger`, která v průběhu testu zaznamenává průběhy stisků tlačítek. Ukládá čas stisku tlačítka od začátku testu a také počet jeho stisknutí. Z těchto dat v průběhu testu vypočítává sledované hodnoty, jako například čas mezi spuštěním aplikace a první provedenou akcí nebo jednotlivé časy strávené výběrem textu. Ulehčí nám tak práci při testování, kdy není nutné tyto hodnoty měřit ručně či vypočítávat z průběhů stisků tlačítek. Po skončení testu uloží zpracovaná data do textového souboru.

Ukázka aplikace

Demonstrace použití rozhraní **A** je ukázána na obrázku 4.3. Ačkoliv je na textu použit nestandardní font písma a fotografie není úplně ostrá, bylo chybně rozpoznáno jen jedno slovo. Zobrazen je překlad po úpravě tohoto slova.



Obrázek 4.3: Ukázka práce s rozhraním A.

Příklad překladu textu na rozhraní **B** je znázorněn na obrázku 4.4. V tomto případě byla všechna slova rozpoznána správně, avšak pro některá nebyl nalezen překlad.

Obrázek 4.4: Ukázka práce s rozhraním B.

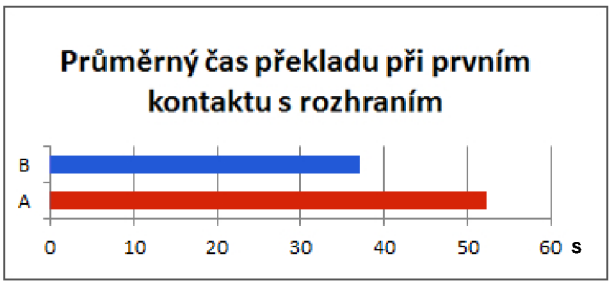
4.2 Uživatelské testy

Testy začaly, jakmile byla dokončena implementace obou rozhraní. Ačkoliv se na vývoji aplikace pracovalo i po zahájení testů, nebyl již proveden zásah do vzhledu či funkce rozhraní.

Uživatelských testů se zúčastnilo celkem devět vybraných uživatelů. Uživatelé byli vybráni tak, jak bylo navrženo v kapitole 3.4.

Po dokončení prvního sezení bylo možné vyhodnotit čas prvního překladu s každým rozhraním. Jako první se překládal text cedule, což byl nejsnazší text k překladu. Ze dvanácti překladů, které uživatel během sezení provedl, nás zajímá jen ten první s rozhraním **A** a první s rozhraním **B**. Na obrázku 4.5 vidíme graf znázorňující průměrný čas prvního překladu s daným rozhraním. Vidíme, že překlad s rozhraním **B** je rychlejší v průměru o 27,2%, což dělá rozdíl 14 sekund.

Je však nutno dodat, že rozhraní **B** není rychlejší jenom proto, že by bylo intuitivnější, ale také, že se nemusí pořizovat a zpracovávat fotografie a také proto, že se ušetří čas při přepínání jednoho layoutu. Tento rozdíl byl experimentálně zjištěn jako 4 sekundy.

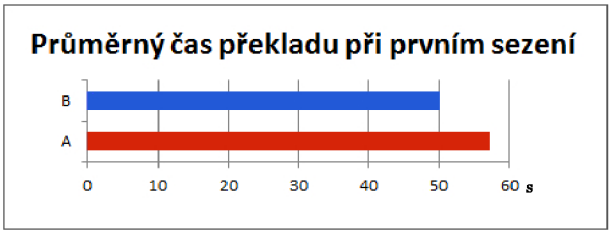


Obrázek 4.5: Graf znázorňující průměrnou dobu strávenou překladem při prvním kontaktu s daným rozhraním.

Při shrnutí časů všech překladů prvního sezení se však časový rozdíl sníží na 7 sekund. Na obrázku 4.6 vidíme graf průměrného času potřebného k jednomu překladu při prvním sezení. To, že průměrný čas při všech překladech je vyšší než čas při prvním kontaktu s aplikací je dáno tím, že ostatní překládané texty byly mnohem složitější než cedule překládaná při prvním překladu.

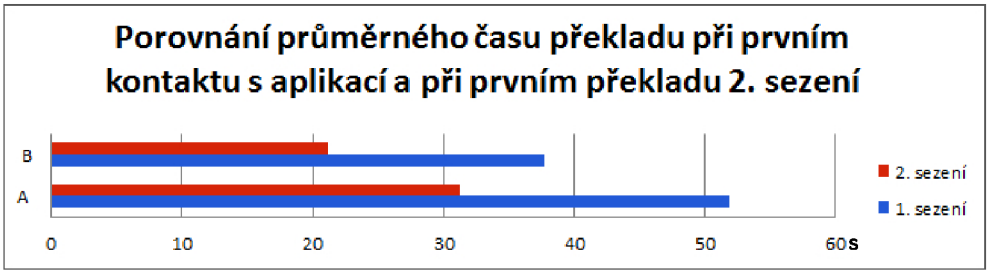
Po dokončení druhých sezení jsme zjistili, jak se zrychlí překlad, pokud už uživatel umí s aplikací zacházet.

Můžeme porovnat časy prvních překladů při obou sezeních, čímž zjistíme o kolik se liší úplně první překlad oproti překladu zkušeného uživatele po čase, kdy s aplikací nepracoval.



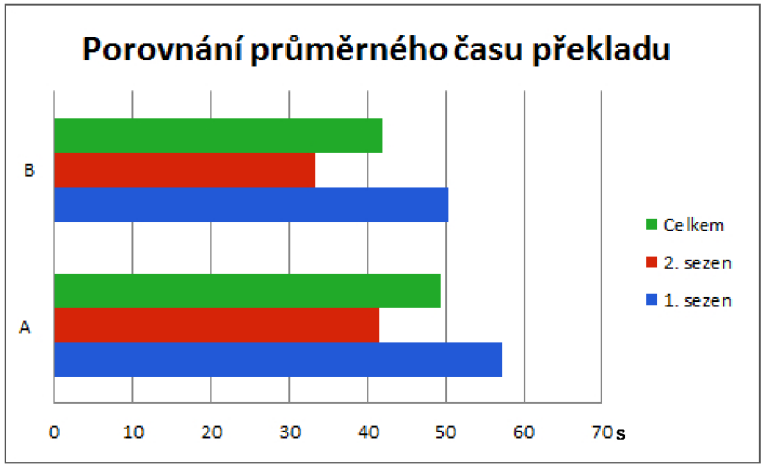
Obrázek 4.6: Graf znázorňující průměrnou dobu strávenou překladem při prvním sezení.

Graf na obrázku 4.7 ukazuje časy obou překladů. Při druhém sezení je čas u rozhraní **A** kratší o 40% a u rozhraní **B** dokonce o 44%.



Obrázek 4.7: Graf znázorňující porovnání mezi časem potřebným k prvnímu překladu při sezení 1 a 2.

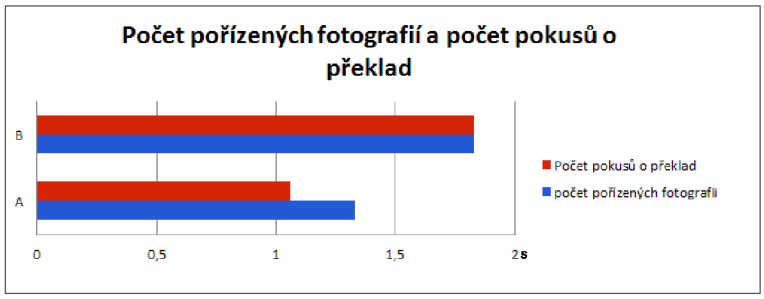
Mnohem přínosnější je však srovnání průměrného času všech překladů mezi prvním a druhým sezením a celkového průměru.celkovým průměrem. Na obrázku 4.8 vidíme graf zobrazující srovnání všech těchto hodnot. Vidíme tak srovnání průměrného času ze 108 překladů na každém rozhraní. Práce na rozhraní **B** je i po vyhodnocení všech výsledků rychlejší, a to o 7,4 sekund.



Obrázek 4.8: Graf znázorňující porovnání mezi časem potřebným k překladu při 1. a 2. sezení spolu s průměrem za obě sezení.

Dalším parametrem podle kterého můžeme posoudit objem práce, kterou uživatel musí vynaložit, je počet pořízených fotografií a počet pokusů o překlad během jednoho překladu.

Počet pokusů o překlad chápeme jako počet přechodů na obrazovku překladu, přičemž u rozhraní **B** je to stejná hodnota jako počet pořízených snímků. Z grafu na obrázku 4.9 vidíme, že obě tyto hodnoty jsou nižší u rozhraní **A**.



Obrázek 4.9: Graf znázorňující porovnání mezi počtem pořízených fotografií a pokusů o překlad.

Při zpracování výsledků dotazníků nevidíme žádnou výraznější preferenci jednoho či druhého rozhraní. Při následné diskuzi s uživateli jsem však zjistil, že preferují spíše rozhraní **B**, neboť se jim překlad s tímto rozhraním zdá snazší a rychlejší. Avšak by i u tohoto rozhraní uvítali možnost práce při různých orientacích displeje, aby se s ním lépe pracovalo při držení jednou rukou.

Z dotazníku se také dozvídáme, že i uživatelé, kteří o OCR nic nevědí mají správné tušení o tom, co ovlivňuje kvalitu rozpoznání textu. Uživatelé také plně chápou význam boungingboxů a orientují se podle nich více než podle náhledu rozpoznaného textu. Když byla uživatelům u rozhraní **A** ukázána možnost provádět rozpoznání textu pomocí speciálního tlačítka, všichni tuto variantu odmítli a chtěli, aby se text rozpoznával automaticky v cyklu.

Návrh změn

Při sledování uživatelů při práci a při společném průchodu aplikací bylo zaznamenáno několik problémových míst rozhraní, které si žádají úpravu.

Prvním problémovým bodem je tlačítko aktualizace překladu. Současné tlačítko evokuje v uživatelích spíše funkci vzájemné záměny rozpoznaného a přeloženého textu a tím i změnu směru překladu. Tlačítko je totiž podobné tomu, jaké má pro tuto funkci v překladači Google. Nové tlačítko by mělo vypadat spíše více jako refresh prohlížeče. Návrh změny je zobrazen na obrázku 4.10.

Obrázek 4.10: Návrh změny obrázku tlačítka pro obnovení překladu.

Další problém je způsoben spíše přehlédnutým bugem, kdy se do přeloženého textu nevkládají znaky nových řádků na místa, kde by měly být. Uživatel se pak v překladu hůře orientuje. Řádkování překladu proto musí být stejné jako u rozpoznaného textu.

Při sledování uživatelů při práci s výběrovým obdélníkem jsem také vypozoroval, že párkrát chtěli zkusit pohnout pouze jednou stranou, přičemž ji chtěli táhnout přes její hranu, nikoliv přes rohový bod. Doplnění obdélníku o možnost pohybovat pouze s jednou hranou by v některých situacích mohlo práci s ním urychlit.

Uživatelé nechtějí mít u rozhraní **B** možnost pořídit snímek dotykem na vyznačený text. Místo toho by tam měl být také focus, neboť když je výběr roztažený přes velkou část obrazovky je obtížné provést focus kliknutím na neoznačenou část.

Výsledné rozhraní

Na základe výsledků uživatelských testů a přání samotných uživatelů bylo do finální verze zvoleno rozhraní **B** se všemi navrženými změnami.

Kapitola 5

Závěr

Cílem této práce bylo navrhnout a implementovat uživatelské rozhraní Mobilního tlumočníku pro Android. Aby bylo možno s rozhraním pracovat, bylo zapotřebí sestavit funkční systém schopný automaticky rozpoznávat text a následně jej přeložit.

V první kapitole práce jsou shrnuty poznatky získané studiem. Je zde popsán princip převodu obrazu na text i metody základního jazykového překladu. Dále jsou popsány doporučení návrhu rozhraní pro Android, metody provádění uživatelských testů a inspirace z již stávajících řešení.

V rámci práce byli definováni budoucí uživatelé a situace, při nichž budou tuto aplikaci používat. Při návrhu budoucího rozhraní byly vytvořeny dvě možné varianty. Ačkoliv obě varianty vedou ke stejnému výsledku, liší se v počtu provedených kroků potřebných k překladu. Druhá varianta rozhraní se snaží zkrátit dobu překladu sloučením dvou kroků dohromady. K aplikaci jsou sestaveny uživatelské testy, při kterých bude sledovat a vyhodnocovat práci uživatelů s rozhraními.

V práci je také popsána implementace prvků aplikace a průběh uživatelských testů. Na základě těchto testů byly v návrhu rozhraní odhaleny možné chyby a ze dvou původních návrhů byl vybrán jeden, který bude použit pro finální aplikaci.

Vytvořená aplikace je schopna rychle provádět překlady dobře čitelného textu. Kvalita rozpoznání textu závisí na Tesseractu, který není navržen pro překlad textu z fotografií reálných scén. Přesnost rozpoznání textu by mohla být zvýšena přidáním předzpracování fotografie. V pořízeném snímku by bylo dobré odstranit nechtěné natočení a perspektivu. Nápisy, které jsou tvořeny pouze odlišnou texturou, jako například nápisy vysekané ve dřevě či kamenu, není možné pomocí Tesseractu rozpoznat. Řešením by mohla být segmentace vstupního obrazu podle textury a následné rozpoznání textu v takto vytvořeném obraze.

Literatura

[1] Dashboards. [online], [cit. 2013-01-01].
URL http://developer.android.com/about/dashboards/index.html

[2] Supporting Multiple Screens. [online], [cit. 2013-01-01].
URL http://developer.android.com/guide/practices/screens_support.html

[3] Task-Centered User Interface Design. [online], [cit. 2013-01-01].
URL http://hcibib.org/tcuid/chap-4.html#4-1

[4] OCR Software Review 2013. [online], [cit. 2013-04-18].
URL http://ocr-software-review.toptenreviews.com/

[5] Comparison of optical character recognition software. [online], [cit. 2013-05-1].
URL hhttp://en.wikipedia.org/wiki/Comparison_of_optical_character_recognition_software

[6] Barnum, C. M.: *Usability Testing Essentials: Ready, Set...Test!* Morgan Kaufmann, první vydání, 2011, ISBN 978-0-12-375092-1.

[7] Marcello, F.: Machine Translation Overview. [cit. 2013-04-18].
URL
https://hermessvn.fbk.eu/svn/hermes/open/federico/slides/SMT13.01.pdf

[8] Nielsen, J.: Why You Only Need to Test with 5 Users. [online], Poslední modifikace 19. 3. 2010. [cit. 2013-04-18].
URL http:
//www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[9] Ray, S.: Tesseract OCR Engine: What it is, where it came from, where it is going. [online], Poslední modifikace 26. 7. 2007. [cit. 2013-04-18].
URL http://tesseract-ocr.googlecode.com/files/TesseractOSCON.pdf

[10] Rice, S. V.; Jenkins, F. R.; Nartker, T. A.: The Fourth Annual Test of OCR Accuracy. Technická zpráva, Information Science Research Institute, University of Nevada, 1995.

[11] Rieman, J.; Franzke, M.; Redmiles, D.: Usability evaluation with the cognitive walkthrough. In *Conference Companion on Human Factors in Computing Systems.*

[12] Rubin, J.; CHisnell, D.: *Handbook of Usability Testing: Howto Plan, Design, and Conduct Effective Tests.* Wiley, druhé vydání, 2008, ISBN 978-0-470-18548-3.

[13] S., M.; C.Y., S.; K., Y.: Historical review of OCR research and development. ročník 80, č. 7, 1992: s. 1029–1058, ISSN 0018-9219.

[14] Seeger, M.; Dance, C.: Binarising camera images for OCR. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, 2001, s. 54–58.

Příloha A

Obsah CD

`/TZ` - tato technická zpráva ve formátu .pdf

`/POSTER` - Plakát

`/VIDEO` - Demonstrační video

`/TESTY` - Data uživatelských testů, formuláře

- `/VSTUPNI_DOTAZNIKY` - Přepis vstupních dotazníků
- `/ZAZNAMY` - Záznamy práce s aplikací
- `/DOTAZNIKY` - Přepis dotazníků za 1. a 2. sezení

`/CODE` - Android projekty aplikací demonstrujících rozhraní A a B

`/DICT` - Slovníková data