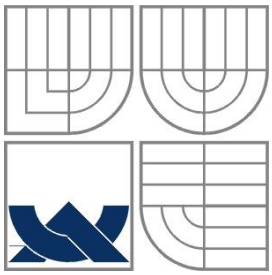
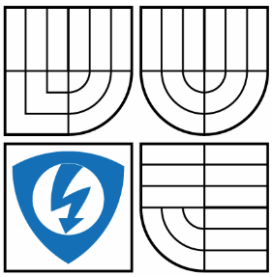


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ZPRACOVÁNÍ OBRAZU V SYSTÉMU ANDROID - ODEČET HODNOTY PLYNOMĚRU

IMAGE PROCESSING USING ANDROID DEVICE - GAS-METER VALUE RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

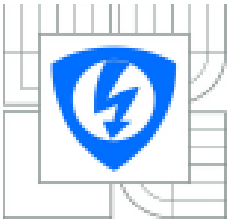
AUTOR PRÁCE
AUTHOR

Bc. MICHAL WERTHEIM

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETER HONEC, Ph.D.

BRNO 2015



VYSOKE UCENI
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Michal Wertheim	***ID:***	140272
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Zpracování obrazu v systému Android - odečet hodnoty plynoměru

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce bude vytvoření programu pro zařízení Android, který bude využívat integrovanou kameru/fotoaparát k získávání obrazu. Práce bude rozdělena do následujících bodů:
1. Proveďte rešerši Android vývojového prostředí a algoritmů zpracování obrazu souvisejích s řešenou problematikou.
2. Vytvořte GUI a interface pro fotoaparát.
3. Navrhněte algoritmy pro zpracování obrazu - vyhodnocení snímku plynoměru, detekce pozic číslic včetně desetiného místa a odečtení hodnoty měřidla.
4. Implementujte algoritmy do zařízení.
5. Ověřte funkcionalitu a spolehlivost detekce a rozpoznání na reálných snímcích. Zaměřte se na nejčastější typy plynoměrů.

DOPORUČENÁ LITERATURA:

Hlaváč, Šonka, Počítačové vidění.
Šonka, Hlávač, Boyle - IMAGE PROCESSING, ANALYSIS, AND MACHINE VISION,

Termín zadání:	9.2.2015	***Termín odevzdání:***	18.5.2015

Vedoucí práce: Ing. Peter Honec, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Diplomová práce se zabývá návrhem zpracování obrazu v systému Android. Volbou vývojového prostředí a jeho implementací. Pracovní postup řešení problematiky zahrnuje vytvoření aplikace a grafického uživatelského rozhraní. Text zahrnuje popis funkcionality aplikace, komunikace s fotoaparátem, uložení a načítání dat. Dále popisuje použité algoritmy a metody zpracování obrazu pro detekci hodnot plynoměru.

Klíčová slova

Zpracování obrazu, Android, Java, počítačové vidění, Eclipse, OpenCV, OCR, detekce údaje plynoměru.

Abstract

This thesis describes the design of the image processing for Android system, consisting of the choice of the development environment and its implementation. Workflow solution to the problem involves development of the Androidapplication and it's graphical user interface. The text includes description of the application functionality, communicationwith a camera, storing and retrieving data. It also describes used algorithms and image processing methods used for detecting values from the counter of the gas meter.

Keywords

Image processing, Android, Java, computer vision, Eclipse, OpenCV, OCR, detection meter data.

Bibliografická citace:

WERTHEIM, M. *Zpracování obrazu v systému Android - odečet hodnoty plynoměru*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 29s. Vedoucí diplomové práce byl Ing. Peter Honec, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Zpracování obrazu v systému Android - odečet hodnoty plynoměru jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **18. května 2015** ……
podpis autora

Poděkování (nepovinné)

Děkuji vedoucímu diplomové práce Ing. Petru Honecovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **18. května 2015** ……

podpis autora

OBSAH

1 ÚVOD

Cílem diplomové práce je vytvoření aplikace pro platformu Android, která bude obsahovat grafické uživatelské rozhraní pro pořízení snímků k dalšímu zpracování. Práce se zabývá vyšetřováním vhodných metod pro detekci údajů plynoměru. Grafické uživatelské rozhraní umožní intuitivní ovládání a funkce pro danou problematiku. Práce pojednává o optimálním využití algoritmů obrazu pro rychlé a spolehlivé zpracování obrazu. Detekce a vyhodnocování snímané scény probíhá autonomně. V práci jsou dále zmíněny možnosti a výhody uvažované platformy. Při postupu práce jsem si vytýčil realizaci problému v jediném jazyce Java.

2 POČÍTAČOVÉ VIDĚNÍ

Zpracování obrazu je forma signálového zpracování, kdy vstupním parametrem systému je obrázek, jako například fotografie nebo snímek videa. Výstupem procesu při zpracování obrazu může být obrázek nebo jeho charakteristika. Metody, které se při zpracování obrazu používají, jsou většinou zpracovány jako dvourozměrný signál, který je kalkulován známými algoritmy ze signal processingu. Počítačové vidění je blízké studiu vidění biologického, jenž studuje a modeluje fyziologické procesy umožňující vizuální vjemy u živých tvorů. Počítačové vidění dále studuje a popisuje softwarově a hardwarově implementované procesy za umělými zrakovými systémy. Mezidisciplinární výměna mezi biologickým a počítačovým viděním přinesla nové poznatky do obou oborů. Počítačové vidění je v určitém směru opak počítačové grafiky. Počítačové vidění je opačný proces, neboť počítačová grafika vytváří obrazová data z informací popisujících zachycené objekty. V současnosti je trendem obě disciplíny kombinovat, jako například v systémech rozšířené reality. Podobory počítačového vidění zahrnují rekonstrukci scény, detekci dějů, vyhledávání událostí v pohyblivé scéně, poznávaní objektů, učení, indexování, odhad pohybu a rekonstrukci obrazu. Počítačové vidění se uplatňuje v následujících oblastech:

- ovládání procesů, například v autonomních vozidlech nebo průmyslových robotech,
- detekce jevů, například při sledování změn bezpečnostního kamerového záznamu,
- organizace informací při indexování databází obrázků nebo videí,
- modelování objektů nebo prostředí, kupříkladu při analýze obrazů z medicínských zobrazovacích technik,
- interakce, například pro zpracování vstupu při interakci počítače se člověkem. [6]

2.1 Vybrané metody počítačového vidění

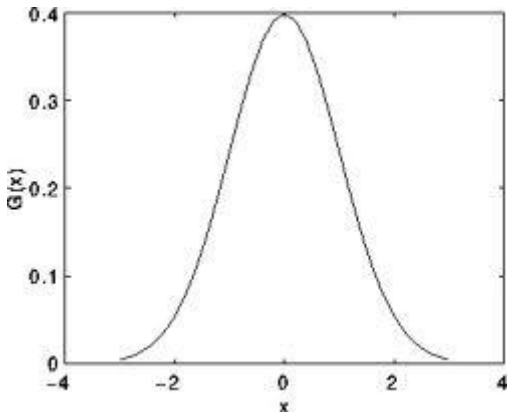
V následující kapitole se věnuji metodám zpracování obrazu, jenž byly použity k řešení úlohy.

2.1.1 Segmentace obrazu

Segmentace obrazu představuje skupinu postupů k dosažení automatické separace požadované scény, na oblasti se společnými oblastmi. Nejběžnější metoda segmentace obrazu je prahování, jenž představuje oddělení obrazu na dvě části s úrovní jasu vyšší než je jedna konkrétní hodnota a část s jasem nižším. Adaptivní prahování představuje pokročilou techniku práhování, umožňující stanovit práh v závislosti na lokálních parametrech obrazu. Metoda segmentace, jenž je jednou z nejbližších ke vnímání lidského oka, je barevný model HSV. Jedná se o rozložení obrazu do tří složek – sytosti, jasu a barevného tónu – metoda se významně uplatnila při hledání textových polí plynoměrů.

2.1.2 Filtrace obrazu

Filtrace obrazu slouží ke zvýrazňování určité informace, jež obraz nese. Filtrací obrazu lze docílit potlačení šumu, vyhlazení obrazu, zvýraznění kontrastu, nebo také detekce hran. V práci se jevila jako velmi užitečná Gaussova filtrace k rozostření obrazu. Tato metoda je jednoduchá a při zpracovávaní obrazu velmi často používaná. Účelem vyhlazení obrázku je redukování šumu. Gausova filtrace je nejužitečnější, ale není nejrychlejší. Gausova filtrace je prováděna konvolucí každého bodu vstupního pole obrázku podle Gausovi funkce. Nevýhodou této metody je rozmazávaní hran a zobtížnění jejich detekce.



Obrázek 1. 1D obraz, kde má bod ležící uprostřed největší váhu. Váha sousedních bodů směrem od středu obrazu klesá. [7]

2.1.3 Houghova transformace

Jednou z problémových oblastí praktické části DP byla segmentace části obrazu, která obsahuje zkoumanou oblast – pole s čítačem měřiče. Tvar čítače je obdélníkového tvaru, jako metodu pro selekci hledané části obrazu jsem tedy zvolil Houghovu transformaci. Výhodou užití HT pro segmentaci objektů, jejichž hranice lze popsat přímkami je odolnost metody vůči jejich nepravidelnostem a přerušením. Houghova transformace představuje metodu užívanou ve zpracování obrazu a počítačovém vidění, která slouží k detekci určitých tvarů na vstupním obraze. Ve své původní podobě, tak jak ji v roce 1962 patentoval Paul Hough sloužila k detekci přímek. Richard O. Duda a Peter E. Hart později rozšiřují HT o možnost vyhledávání obecných analyticky popsatelných tvarů, zejména kuželoseček. [8]

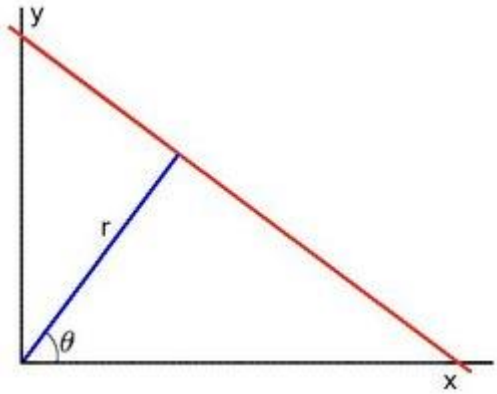
2.1.4 Houghova transformace přímek

Při detekci přímek v obraze HT vychází za normálového tvaru analytického zápisu přímky, který je nezávislý na orientaci os souřadnic.

$$r = x \cos\theta \; y \sin\theta \qquad (1)$$

Kde r představuje velikost normály od počátku a

θ ředstavuje velikost úhlu vůči ose x.

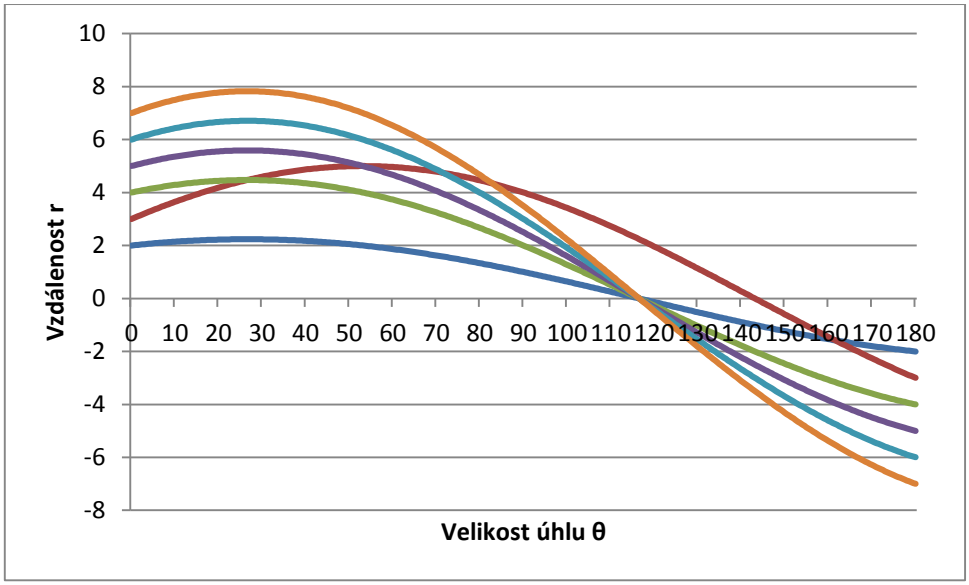


Obrázek 2. Parametry normálového tvaru rovnice přímky [9]

Následně proběhne transformace jednotlivých bodů (pixelů) z prostoru souřadnic (x, y) do Houghova parametrického prostoru, jehož dimenze je udávaná počtem parametrů (v případě hledání přímek se jedná o dvourozměrnou dimenzi) a to tak, že pro každý pixel je vykreslena množina všech možných parametrů přímek, které daným bodem mohou procházet, tyto tvoří sinusovou křivku. Budeme-li uvažovat následující body:

Table 1. Zvolené body

Bod č.	***1***	***2***	***3***	***4***	***5***	***6***
x	2	3	4	5	6	7
y	1	3	2	2,5	3	3,5



Obrázek 3. Parametrické pole θ, r všech přímek, které procházejí vybranými body

Pro všechny body, u kterých se křivky po transformaci protínají, platí, že jsou kolineární (parametry normálového tvaru přímky, která libovolnými dvěma body prochází, se shodují). Jednotlivé průsečíky jsou zaznamenávány do akumulačního pole, přičemž body s největší četností indikují přítomnost zřetelných přímek na vstupním obraze.

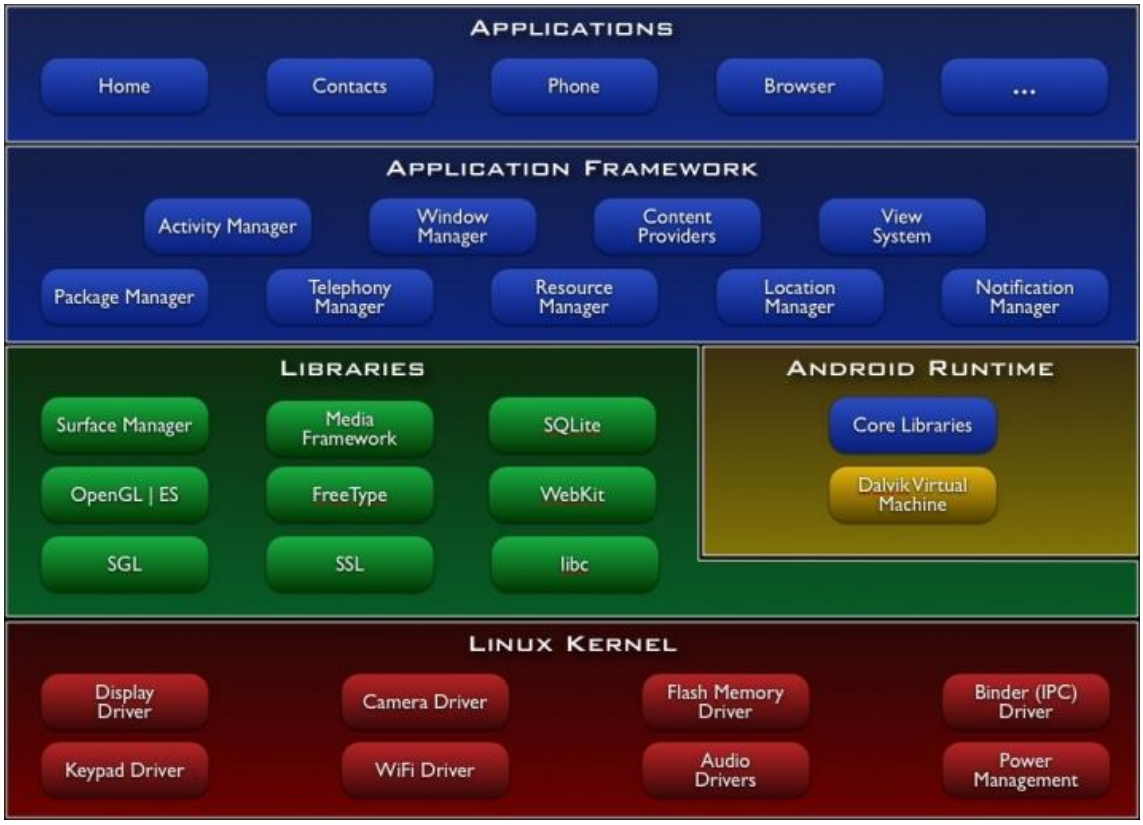
3 ANDROID

Android je open source operační systém založený na Linuxovém jádře pro přenosná zařízení jako jsou chytré telefony a tablety. Tento operační systém vznikl v roce 2003, kdy společnost stejného názvu založili Andy Rubin, Rich Miner, Nick Sears a Chris White, jejichž záměrem bylo vytvoření chytřejšího mobilního telefonu, který splní nároky uživatele a jeho umístění. Do širšího povědomí se systém dostal v roce 2005, kdy byla společnost Android Inc. odkoupena firmou Googlem načež se zrychlil vývoj systému. V roce 2007 bylo po uvedení iPhone vytvořeno konsorcium Open Handset Alliance (mimo Google založeno firmami Nvidia, Samsung, LG, HTC, Motorola, Intel, Qualcomm, Ebay, T-Mobile a Telefonica a další), jež představilo nový operační systém Android, který je založen na otevřených standardech. První mobilní telefon s operačním systémem Android se začal prodávat na konci roku 2008 a jednalo se o T-Mobile G1, jenž byl vyroben firmou HTC. Android operační systém je upraven pro konkrétní požadavky využití na mobilních zařízeních zejména s procesory ARM. V první fázi byly úpravy systému součástí hlavní verze linuxového jádra, později však Google začalo se samostatnou správou.

Obrázek 4. Logo systému Android

3.1 Architektura systému android

Aplikace na systému android komunikují prostřednictvím jednotného API, s pomocí kterého mají aplikace přístup k funkcím zařízení (akcelerometru, GPS modulu, displeje a podobně) a operačního systému. Běh aplikací je zajištěn virtuální mašinou Dalvik VM, jenž je obdoba JAVA VM. Aplikace jsou vyvíjené v jazyce JAVA, avšak následně přeložena pro virtuální mašinu, které je optimalizované pro běh na mobilních zařízeních.



Obrázek 5. Grafické znázornění architektury systému Android

Linuxové jádro včetně provedených úprav je poskytováno pod licencí GPL a zbývající část operačního systému Android je poskytována pod licencí Apache 2.0. Tím je zdrojový kód přístupný komukoliv, díky čemuž se systém značně rozšířil mezi výrobci telefonů. ¨

3.1.1 Charakteristické vlastnosti systému Android

Významnou vlastností Androidu je podpora multitaskingu, jenž představuje běh každé aplikace ve vlastní virtuální mašině, v samostatném odděleném procesu. V situaci, kdy uživatel pošle aplikaci na pozadí, systém zajistí její další běh. Aplikace se mohou nacházet v různých stavech, mezi kterými systém přepíná podle potřeby uživatele a kapacity dostupné operační paměti (jedná se o stavy běžící, uspané, nebo zastavené). Ve většině tento princip stavů operuje korektně, a uživatel se nemusí starat o ukončování aplikací či sledování jejich běhu. Od verze androidu 2.3 lze použít v případě potřeby integrovaný správce úloh a není potřeba doinstalovat správce úloh třetích stran, jejichž používání nebylo v očích vývojářů příliš vítáno. Významnou charakteristikou systému je jeho otevřenost, kdy je výrobcům a koncovým zákazníkům umožněno zpřístupnění kupříkladu grafického rozšíření uživatelského rozhraní a lišit se tímto způsobem od výrobců zařízení s totožným operačním systémem. Typickým příkladem je rozhraní HTC Sense, nebo TouchWiz od Samsungu viz obrázek 3. [1]

Obrázek 6. HTC Sense vlevo a TouchWiz vpravo [2]

3.1.2 Linux Kernel

Linux kernel představuje nejnižší vrstvu architektury systému android. Je postaveno na verzi Linux kernelu 2.6, přičemž android užívá verzi jádra s několika specifickými doplňky jako wakelocks, což je systém zprávy paměti, který je však agresivnější při zachování paměti (data vybraných aplikací). Dalším důležitým doplňkem pro systém android je ovládač Binder IPC, jenž umožňuje vyšší vrstvě API komunikovat se službami systému Android.

3.1.3 Knihovny

Nativní knihovna umožňuje zařízení zpracování různých typů dat. Tyto knihovny jsou psány v jazycích C a C++, a jsou specifikovány pro konkrétní hardware. Mezi důležité knihovny patří:

- Surface Manager - umožňuje přístup k subsystému displeje
- Media Framework- rámec médií poskytuje multimediální kodeky umožňující nahrávání a přehrávání různých formátů médií
- SQLite – představuje databázový engin používaný v Androidu pro účely ukládání dat
- WebKit – jedná se o prohlížecí engin sloužící k zobrazení HTML obsahu
- OpenGL – slouží k vykreslení dvou a třídimenziální grafiky na obrazovce.

3.1.4 Android Runtime

Android Runtime obsahuje dvě části - Dalvik Virtual Macheni a knihovny Java Core. Dalvik Virtual Machine je typem Java Virtual Machine použité v zařízeních Android ke spouštění aplikací a je optimalizována pro nízkou spotřebu výpočetního výkonu a paměťových prostředků. Na rozdíl od Java Virtual Machine Dalvik Virtual Machine

nespouští soubory s koncovkou class, nýbrž soubory s příponou dex, které jsou vytvořeny po zkompilování souborů.class a poskytují vyšší účinnost v prostředí s nižšími výpočetními prostředky. Dalvik Virtual Machine umožňuje současné vytvoření většího množství instancí virtuální mašiny, přičemž je zajištěna bezpečnost, izolace, správa paměti a podpora vláknování. Knihovny Java Core se liší od knihoven Java SE a Java ME, obsahují však většinu funkcionalit, jenž jsou definovány v knihovnách Java SE.

3.1.5 Application Framework

Představuje bloky, které vyvinuté aplikace přímo integrují. Tyto programy spravují základní funkce telefonu, jako je hospodaření s prostředky, správa hlasových hovorů aj. Vývojář by měl tyto bloky brát jako základní nástroje pro tvorbu aplikací. Důležité bloky Application Framework jsou:
Activity Manager – řídí životní cyklus aktivity aplikace
Content Providers – spravuje sdílení dat mezi jednotlivými aplikacemi
Telephony Manager – řídí všechny hlasové hovory a je použitý, pokud je v aplikaci žádoucí přístup k hlasovým hovorům.
Location Manager – správa polohy pomocí vestavěného GPS modulu, nebo mobilní sítě
Resource Manager – správa různých typů prostředků použitých v palikaci

3.1.6 Applications

Aplikace jsou nejvyšší vrstvou architekturi Androidu. Některé standardní aplikace jsou předinstalovány na každém zařízení. Vývojář má možnost vytvoření aplikace s přístupem k veškerým funkcím Androidu.

3.2 Základní části aplikace Android

Veškeré aplikace běžící na systému android musí obsahovat části, jenž jsou popsány v následujících podkapitolách.

3.2.1 Aktivity

Aktivita sestává z jedné obrazovky s uživatelským rozhraním. Kupříkladu aplikace elektronické pošty může obsahovat jednu aktivitu pro náhled příchozí pošty, další aktivitu ke psaní pošty a poslední aktivitu pro účel čtení elektronické pošty. Přestože aktivity operují pohromadě k vytvoření komfrtu pro uživatele aplikace, každá aktivita je na ostatních aktivitách nezávislá. Jiná aplikace může navíc spustit některou z aktivit, pokud to aplikace pošty umožňuje – například aplikace fotoaparát může spustit aktivitu elektronické pošty pro vytvoření nové zprávy, aby mohl uživatel sdílet pořízenou fotografii.

3.2.2 Služby

Služba je komponenta, která běží na pozadí a vykonává dlouhotrvající operace nebo konají činnost pro vzdálených procesů. Služba neposkytuje uživatelské rozhraní. Kupříkladu služba může přehrávat hudbu na pozadí, zatímco uživatel používá jinou aplikaci, nebo také může sbírat data prostřednictvím sítě, aniž by došlo k blokování interakce uživatele s aktivitou. Další komponenta, jako je aktivita, může spustit službu a nechte jí běžet, nebo jí k sobě přiřadit, aby s ní mohla komunikovat.
Služba je implementována jako podtřída služby a můžete se dozvědět více o tom v průvodcovských služeb pro vývojáře. Služba je implementována jako podtřída *Service.*

3.2.3 Poskytovatelé obsahu

Poskytovatelé obsahu spravují sdílený soubor aplikačních dat. Data mohou být uložena v souborovém systému, v databázi SQLite, na internetu, nebo na jakémkoliv jiném perzistentním úložišti, ke kterému aplikace může získat přístup. Prostřednictvím poskytovatele obsahu mohou jiné aplikace požadovat, nebo dokonce měnit data (pokud to ovšem poskytovatel obsahu umožní). Systém Android například poskytuje poskytovatele obsahu, který spravuje kontaktní informace uživatele, načež může jakákoliv aplikace s příslušným oprávněním zpřístupnit součást poskytovatele obsahu (například *ContactsContract.Data*) ke čtení nebo zapsání informací o určité osobě. Použité poskytovatelů obsahu je rovněž vhodné ke čtení a zápisu dat, které jsou soukromé pro vlastní aplikaci a nesdílené, jako aplikace poznámkový blok, které používá poskytovatele obsahu k uložení poznámek. Poskytovatel obsahu je realizován jako podtřída *ContentProvider* a musí implementovat standardní sadu rozhraní API, které umožní dalším aplikacím provádění úkonů.

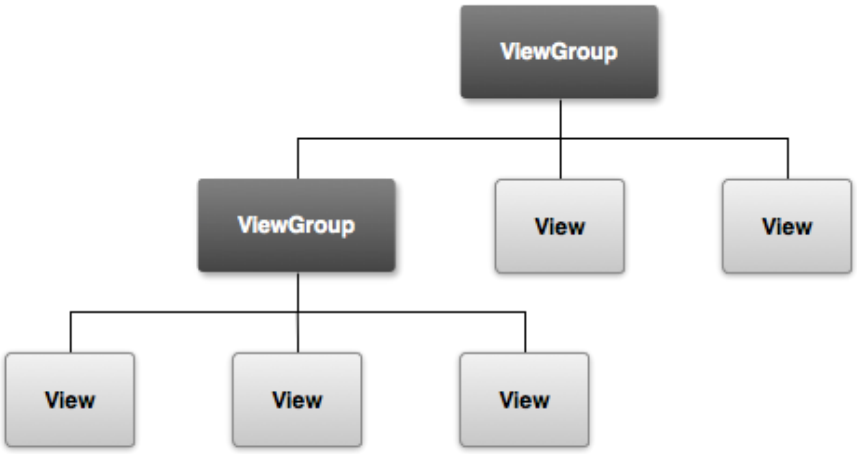
3.2.4 Přijímače vysílání

Broadcast receiver neboli přijímač vysílání je komponenta, které reaguje na celý systém vysílání oznamování. Mnoho vysílání pocházejí ze systému - například vyslání oznámil o vypnutí obrazovky, nízkém stavu nabití baterie, nebo o pořízení obrázku. Aplikace může také zahájit vysílání, kupř., aby dala vědět dalším aplikacím, že některá data byla stažena do zařízení, a jsou pro tyto aplikace k dispozici pro použití. Přestože přijímače vysílání nezobrazují uživatelské rozhraní, mohou vytvořit oznámení ve stavové liště pro upozornění uživatele, ve chvíli když událost vysílání nastane. Obvykle je příjem vysílání pouze bránou k jiným komponentám, kdy je zapotřebí jen minimální množství práce. Například je možné zahájit vykonávání služby na základě uskutečnění nějaké události. Přijímač vysílání je implementován jako podtřída *BroadcastReceiver* a každé vysílání je doručeno jako objekt *Intent* (záměr). [4]

3.2.5 Další komponenty systému Android

Pro účely vyvíjení aplikací pro platformu Android je nutno seznámit se s dalšími součástmi, které používá většina aplikací:

- Layouts – představují dílčí deklaraci grafického uživatelského rozhraní veškerých obrazovek aplikace v jazyce XML.
- Manifest – jedná se o soubor v kořenovém adresáři projektu a obsahuje základní informace o aplikaci. Definuje základní předpoklady pro spuštění kódu aplikace jako například minimální úrovně API, nebo také oprávnění aplikace k přístupu k aplikačním komponentám (v tomto případě zabudovaný modul fotoaparátu).
- View a ViewGroup – jsou objekty, představující grafické uživatelské rozhraní aplikace viz obr 4. *View* objekty představují UI UI widgety jako například textová pole (*EditText*), nebo tlačítka (*Button*). Objekty *ViewGroup* jsou neviditelné kontejnery, definující seskupení potomků *View*.
- Resources – jedná se o podadresáře zdroje aplikace (res/) jenž obsahuje ilustrace, texty, jazyky a jejich parametrů.



- *Obrázek 7. Ilustrace objektů ViewGroup a potomků View [5]*

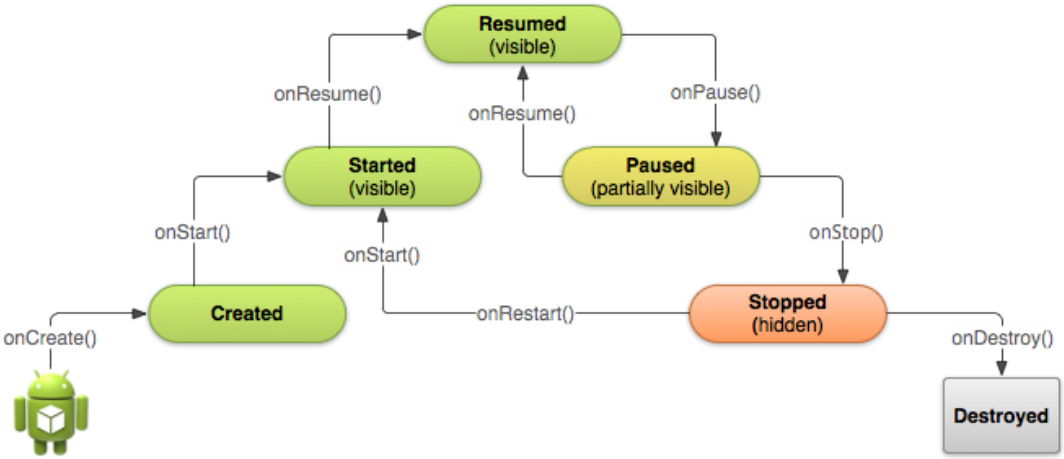
3.2.6 Správa aplikace

Pokud uživatel spustí aplikaci a později přepne na jinou, nebo jen vypne podsvícení displeje, instance aktivity ve vlastní aplikaci přechází mezi různými stavy ve svém životním cyklu. Pokud je aktivita spuštěná poprvé, přechází do popředí systému a dostává uživatelovu pozornost. V průběhu tohoto procesu systém Android volá řadu metod životního cyklu aktivity, přičemž je nutno je potřeba mít dopředu nakonfigurovány komponenty a grafické uživatelské rozhraní.

3.2.7 Životní cyklus aktivity

V průběhu trvání aktivity systém volá sady metod životního cyklu v sekvenci, připomínající stupňovitou pyramidu viz obrázek 5, přičemž každá fáze životního cyklu aktivity představuje samostatný krok na diagramu. Ve chvíli, kdy systém vytvoří novou

instanci aktivity, každé volání metod posouvá stav aktivity o jeden krok směrem k vrcholu diagramu. Vrcholem diagramu je stav (*Resumed*), kdy aktivita běží na popředí a uživatel s ní může komunikovat. Po opuštění aktivity systém volá metody, které posouvají stav aktivity v diagramu směrem dolů, za účelem zničení aktivity. V některých případech se však aktivita může vrátit do stavu při opouštění.



Obrázek 8. Grafické znázornění životního cyklu aktivity

3.2.8 Stavy životního cyklu aktivity

- Created a Started – jedná se o přechodné stavy, kdy po systémovém volání metody *onCreate()* okamžitě následuje volání metody *onStart()* a poté *onResume().*
- Resumed – neboli stav běžící představuje stav, kdy se aktivita nachází v popředí a přijímá informace o uživatelském vstupu.
- Paused – pokud přechází aktivita do pozastaveného stavu, znamená to částečné překrytí jonou aktivitou, nebo se nad ní může nacházet jiná průhledná aktivita. Aktivita je tedy v tomto stavu částečně viditelná, nicméně není přístupná uživatelským vstupům.
- Stopped – aktivita je zastavena v situaci, kdy pokud není vidět, avšak Framework její objekt doposud nezničil. Aktivita nepřijímá uživatelský vstup a je možný opětovný návrat k aktivitě pokud je v paměti dostatek prostředků. [6]

4 SOFTWAROVÉ VYBAVENÍ POTŘEBNÉ PRO PRÁCI

4.1 Java

Java je globální standard pro vývoj embedded a mobilních aplikací, her, Web-based obsahu a enterprise software. Hlavními výhodami vývoje aplikací v Javě je hlavně přenositelnost mezi platformami, díky využití JVM (Java Virtual Machine). JVM je soubor programů, který využívá virtuální stroj ke zpracování mezikódu (Java bytecode). Dále se pak Java využívá k vývoji programů, které mohou běžet přímo ve webových prohlížečích, fórech, ale i aplikací pro mobilní zařízení jako jsou telefony, mikrokontroléry, sensory a podobně. Java je developery volena jako první jazyk a je to také nejvíce volena platforma. K vývoji aplikací v Javě slouží JDK (Java Development Kit) pro konkrétní platformu a jako IDE (Integrated Development Environment) se nejčastěji používají nástroje: Netbeans, Eclipse a nově také Android Studio. [10]

4.2 Eclipse

Eclipse je open-source IDE a podporuje vývoj aplikací v různých jazycích, není tedy omezena jen na Javu. Pomocí doplňků je možno dále rozšiřovat záběr použitelnosti nástroje. Pro vývoj aplikací určených pro android je nutné do Eclipse nainstalovat plugin ADT (Android Development Tools). ADT umožňuje vytvořit navíc i aplikační UI, přidávat balíky založené na Android Framework API, debugovat aplikace pomocí SDK tools a také exportovat .apk soubory.
K instalaci ADT je nutná Eclipse a také Android SDK (Standard Development Kit). Ten je tvořen z modulárních balíků, které lze stáhnout pomocí Android SDK Manageru. Takto lze snadno vybrat a získat nové balíky, ale zároveň snadno aktualizovat doplňky starší. Další součástí je emulátor, na kterém lze testovat aplikace i bez připojení skutečného zařízení. [11]

4.3 Tesseract OCR

Tesseract je open-source OCR (Optical character recognition) engine. Pracuje s binárním obrázkem na vstupu, který převádí na čistý text. Převod probíhá v několika fázích. První z nich je connected component analysis, při které jsou nalezeny outlines jednotlivých komponent a uloženy do Blobs. Blobs jsou děleny na řádky textu a ty jsou dále děleny na slova podle mezer mezi znaky. Další analýza probíhá ve dvou částech. V první části se pokouší rozpoznat jednotlivá slova a každé, které je dostatečně rozpoznáno je předáno adaptive classifieru, který díky tomu může lépe rozpoznávat další slova na stránce. Pro vylepšení výsledků je průchod opakován ještě jednou.

Hlavními částmi procesu rozpoznávání textu jsou blob filtering a line construction. Filtrování porovnává výšku znaků (v rámci blobu), bere jejich medián a ty, které jsou značně menší označuje jako interpunkční znaménka a naopak ty, které přesahují hranici filtruje, resp. porovnává překryv blobů pro detekci diakritických znamének, které přiděluje ke konkrétnímu řádku textu. Tímto způsobem z blobů vytvoří unikátní řádky textu, se kterými dále pracuje. Pro zlepšení výsledků se využívá baseline fitting, což umožňuje detekci zaoblených řádku, které vznikají při skenování stran. Pokud je to možné, snaží se identifikovat fixed pitch a porportional words (non-fixed pitch). Když najde fixed pitch, rozdělí slovo na znaky. U proportional je to složitější.
Pokud je výsledek po rozdělení slova na znaky nedostatečný, jsou body pro useknutí znaků nalezeny v konkávních vertices po polygonální transformaci. Na takvém místě je provedeno rozdělení na znak a pokud je výsledek opět nedostatečný, je změna vrácena do původního stavu. I po provedení těchto úprav nemusí být slovo stále dostatečně rozpoznatelné a přichází na řadu associator, který se snaží ze segmentů blobů detekovat nekompletní znaky.
Během trénování Static character classifieru jsou použity segmenty polynomiální aproximace, se kterými jsou během rozpoznávání porovnávány malé features, které jsou extrahovány z outline. Klasifikace probíhá ve dvou částech. V první části je vytvořen seznam tříd znaků, které mohou být nalezeny, a po průchodu jsou u každé třídy uvedeny počty nalezených výskytů. [12]

4.4 OpenCV

OpenCV (Open Source Computer Vision) je knihovna programových funkcí, zaměřených především na počítačového vidění v reálném čas, vyvinutá ve výzkumném centru Intel. Knihovna je zdarma pro použití v rámci BSD open-source licence. Knihovna podporuje široké spektrum platform. Zaměřuje se především na zpracování obrazu v reálném čase. Knihovna je nejčastěji implementována v jazyce c++, jazyce Python, avšak na rozmachu je i její rozšířený v jazyce Java.

5 VÝVOJ APLIKACE

V době započatí činnosti na diplomové práci, byl Eclipse IDE s doplňkem Android Development Tools (ADT) pro vývoj aplikací pro operační systém Android primárním nástrojem, který umožňuje rychlou konfiguraci projektu a uživatelského rozhraní aplikace pro Android. Po několika týdnech byl však z pozice primárního vývojového nástroje Eclipse nahrazen nástrojem Android Studio IDE. Po porovnání vlastností obou nástrojů lze na první pohled vyzdvihnout stabilnější a svižnější chod nástroje Android Studio IDE, načež lze v budoucí fázi vývoje aplikací očekávat přechod vývojářů k novějšímu IDE. Mezi vývojáři je také populární vývojové prostředí Netbeans s doplňkem Android SDK.

5.1 AndroidManifest vytvořené aplikace

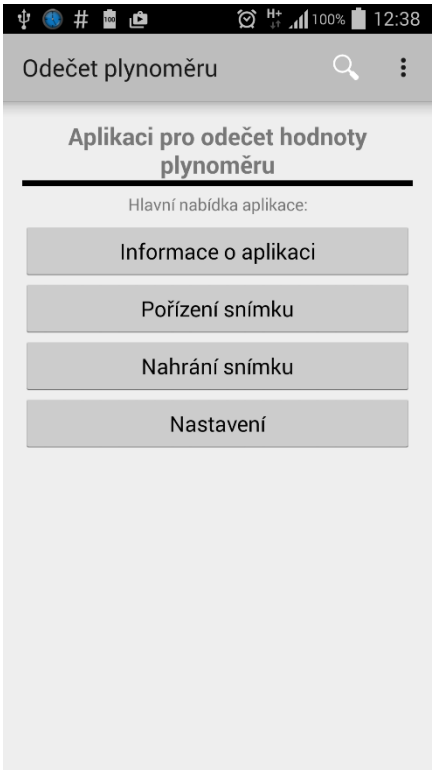
Význam souboru Manifest byl uvedený v kapitole *Základní části aplikace Android*. Důležitým parametrem jsou minSDkVersion a targetSdkVersion, kdy byla minimální hodnota nastavena na 15 a cílová na 21, pro podporu nových funkcí na cílovém zařízení Samsung Galaxy s4 s verzí systému Android 4.4 což odpovídá API 19. Dále bylo povoleno použití vestavěného fotoaparátu, jehož přítomnost je podmínkou pro spuštění aplikace a také bylo zpřístupněno externí úložiště pro možnost ukládání pořízených fotografií. V souboru AndroidManifest byla dále definované mateřská aktivita u všech aktivit, jenž jsou v aplikaci použity z důvodu umožnění rychlého návratu na domovskou obrazovku.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myfirstapp2"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-feature
        android:name="android.hardware.camera"
        android:required="true" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-sdk
        android:minSdkVersion="15"
        android:targetSdkVersion="21" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

Obrázek 9. Ukázka souboru AndroidManifest

5.2 Uživatelské rozhraní

Rozvržení objektů uživatelského rozhraní bylo vytvořeno v jazyce XML a každé vytvořené aktivitě přísluší jeden design vytvořený v tomto jazyce. Po spuštění příslušné aktivity je pak toto rozložení načteno metodou setContentView(), náhled hlavní obrazovky z editoru XML je k dispozici na obr 6.



Obrázek 10. Úvodní obrazovka aplikace

Okno obsahuje dvě textová pole (*TextView*), oddělovač černé barvy (*View*) a čtyři tlačítka (*Button*) umožňující přechod do dalších podřadných aktivit. Ve všech obrazovkách bylo použito lineární rozvržení. Pro účel pohodlného procházení aplikací byla implementována stavová lišta, rozšířením tříd pro možnost návratu na domovskou aktivitu. Obdobným způsobem bylo vytvořeno rozhraní ostatních aktivit.

5.2.1 Aktivita Nastavení

Aktivita s názvem *Settings* byla vytvořena pro budoucí potřebu ukládání klíčových údajů. Pro tento účel bylo použito API *SharedPreferences,* které umožňuje jednoduchou metodu čtení a zápisu klíčových hodnot. Tato metoda navíc umožňuje přístupu k údajům z dalších aktivit.

Obrázek 11. Spuštěná aktivita Nastavení

5.2.2 Aktivita Pořízení snímku

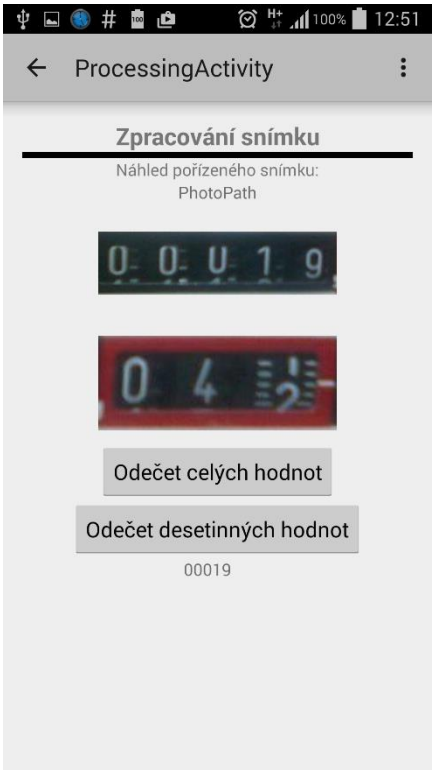
Pořízení snímku bylo realizováno metodou odeslání záměru *Intent* zabudované aplikaci fotoaparátu, kdy po odeslání požadavku pro pořízení snímku bude vytvořený název souboru, jenž zajišťuje unikátní identifikaci podle času pořízení, a který bude použitý pro uložení pořízené fotografie do fotogalerie. Pro ověření pořízení snímku je na aktivitě následně zobrazena miniatura pořízeného snímku. Viz obrázek 10. Pro možnost detekce uložených snímků byla vytvořena aktivita „Nahrání snímku“, jenž stejně jako aktivita pro pořízení snímku obsahuje volbu „Zpracování snímku“. Po úspěšném nahrání nebo vyfocení snímku může uživatel obrázek vyhodnotit a po stisknutí volby *Zpracování snímku* dojde ke spuštění aktivity *Image processing* a zároveň dojde k odeslání reference na vyšetřovaný soubor.

Obrázek 12. Spuštěná aktivita Pořízení snímku

5.2.3 Aktivity Zpracování obrazu

Aktivita zpracování je nejdůležitější aktivitou vytvořené aplikace. Bezprostředně po inicializaci aktivity *Processing activity* dojde v rámci metody životního cyklu *onCreate* k volání implementované funkce *showFields(),* jenž slouží k nalezení polí s hledanými číselnými údaji. Pro spolehlivou segmentaci vyšetřované scény bylo potřeba zajistit narovnání obrazu. Po provedení rešerše nebyla nalezena žádná spolehlivá metoda, která by umožňovala nalézt úhel natočení vyšetřované scény. Po důkladné analýze dostupných algoritmů jsem se rozhodl sestavit výpočetně nenáročnou a spolehlivou metodu pomocí analýzy histogramu úhlů natočení Houghových linií. Metoda spočívá v nalezení dvou nejvýraznějších směrů (vodorovný a svislý směr). Aplikace Houghovy

transformace na ne reálných snímcích je prakticky nereálná vzhledem k vyskytujícímu se zašumění. Z tohoto důvodu byly snímky předzpracovány aplikací posloupností Gausova filtru, Bilaterálního filtru, Adaptivního prahování a Cannyho detekcí hran. Analýzou Houghových linií bylo možné dopočítat korekční úhel natočení. Následujícím krokem bylo nalezení číselného údaje. Po provedení průzkumu běžných plynoměrů byl zjištěn fakt, že hodnoty desetinných míst jsou zvýrazněny červenou barvou. Segmentace obrazu byla realizována převedením obrazu do HSV roviny, ze které lze stanovit hledanou oblast. Provedením morfologické operace a následné ho vyhodnocení nalezených kontur bylo možné stanovit přesnou polohu vyšetřovaných údajů. Údaje desetinného a celočíselného místa byly analyzovány v samostatných instancích. Číselné údaje byly segmentovány pomocí detekce jasových složek. Po provedení další detekce kontúr bylo možné vyseparovat jednotlivé číselné údaje. Rozpoznání samotných čísel bylo realizováno nástrojem Tesseract ocr. Vzorky byly pro lepší detekci klíčových charakteristik prahovány jasovou technikou.



Obrázek 13. Ukázka aplikace s rozpoznáním celočíselného údaje

6 ZÁVĚR

Výsledkem práce je představení klíčových vlastností systému Android, jenž umožňuje vytvoření autonomního systému pro detekci údajů plynoměru. V práci jsou diskutovány různé metody zpracování obrazu, které se více méně uplatní pro řešení dané problematiky. Vytvořený systém je schopen kvalitní detekce pole hodnot i při špatných podmínkách focení. Aplikované metody zpracování obrazu se projevily jako slabší při detekci neúplných čísel, kdy OCR není schopno rozpoznávat neúplná čísla. Možné zlepšení aplikace spočívá v dokonalejší extrakci klíčových charakteristik.

Literatura

1. Počítačové vidění. *Wikipedia.* [Online] 22. 3 2015. http://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%A9_vid%C4%9Bn%C3%AD.

2. Smoothing Images. *http://docs.opencv.org/.* [Online] 25. 4 2015. http://docs.opencv.org/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html.

3. Richard O. Duda, Peter E. Hart. *Use of the Hough Transformation to Detect Lines and Curves in Pictures.* místo neznámé : Artificial Intelligence Center, 1971.

4. Hough Line Transform. *http://docs.opencv.org/.* [Online] 2014. [Citace: 10. 4 2015.] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html.

5. Marvan, Filip. Android jaký je. *diit.cz.* [Online] 27. Červenec 2011. [Citace: 4. Leden 2015.] http://diit.cz/clanek/android-jaky-je.

6. Samsung's Touchwiz Nature UX 2.0 vs. HTC Sense 5.0: Which is Better? *ANDROIDPIT.* [Online] 13. Květen 2013. [Citace: 5. Leden 2015.] http://www.androidpit.com/samsung-s-touchwiz-nature-ux-2-0-vs-htc-sense-5-0.

7. Application Fundamentals. *Android Developers.* [Online] [Citace: 4. Leden 2015.] http://developer.android.com/guide/components/fundamentals.html.

8. Česká dokumentace vývoje aplikací pro Android. *wiki.wzk.cz.* [Online] 5. 11 2012. [Citace: 4. Leden 2015.] http://wiki.wzk.cz/index.php/%C4%8Cesk%C3%A1_dokumentace_v%C3%BDvoje_aplikac%C3%AD_pro_Android.

9. Starting an Activity. *Android Developers.* [Online] [Citace: 4. Leden 2015.] http://developer.android.com/training/basics/activity-lifecycle/starting.html#lifecycle-states.

10. Learn About Java Technology. *www.java.com.* [Online] Oracle. [Citace: 2015. 2 20.] https://www.java.com/en/about/.

11. ADT Plugin Release Notes. *http://developer.android.com/.* [Online] Google. [Citace: 15. 1 2015.] http://developer.android.com/tools/sdk/eclipse-adt.html.

12. Smith, Ray. *An Overview of the Tesseract OCR Engine.* [pdf.] místo neznámé : Google Inc, 2007. 0-7695-2822-8/07.

13. Android Architecture – The Key Concepts of Android OS. *android-app-market.com.* [Online] 17. Únor 2012. [Citace: 4. Leden 2015.] http://www.android-app-market.com/android-architecture.html.

Seznam příloh