
# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
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** — **Bc. Michal Wertheim**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Peter Honec, Ph.D.**
SUPERVISOR


**BRNO 2016**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

***Student:*** Bc. Michal Wertheim ***ID:*** 140272

***Ročník:*** 2 ***Akademický rok:*** 2015/16

**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 souvisejících 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:*** 8.2.2016 ***Termín odevzdání:*** 16.5.2016

***Vedoucí práce:*** Ing. Peter Honec, Ph.D.

***Konzultant diplomové práce:***

**doc. Ing. Václav Jirsík, CSc.**, *předseda oborové rady*

## 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, Open CV, 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, 2016. 53 s. 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: **16. května 2016** ……..

podpis autora

## Poděkování

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

V Brně dne: **16. května 2016** ……

podpis autora

# Obsah

# ÚVOD

Cílem diplomové práce je vytvoření aplikace pro zařízení Android, která bude využívat integrovanou kameru/fotoaparát k získávání obrazu. Výstupem je aplikace pro mobilní telefony, která umí skenovat plynoměr v domácnosti a analyzovat informaci množství spotřebovaného plynu a identifikátor plynoměru. Aplikace funguje na chytrých mobilní telefonech s platformou Android OS a obsahuje intuitivní grafické uživatelské rozhraní pro snadné ovládání i nezaběhlým uživatelům v oblasti chytrých telefonů.

Práce se zabývá vyšetřováním vhodných metod pro detekci údajů plynoměru a jejich implementací. Dále pojednává o optimálním využití algoritmů obrazu pro rychlé a spolehlivé zpracování obrazu. Čímž se může odlišit od podobných řešení. Detekce a vyhodnocování snímané scény probíhá autonomně.

Práce může sloužit jako informační zdroj problematiky strojového vidění nezasvěceným inženýrům anebo jako šablona pro implementaci na jiné indikátory elektroměrů nebo průtokoměrů. Z marketingového hlediska má aplikace na analýzu plynoměru velký význam pro odečítání a snadné archivování dat o spotřebovaném plynu v domácnostech a bude aktuální do doby, než budou lokální měřiče připojeny přímo k informačním sítím.

V práci jsou dále zmíněny možnosti a výhody uvažované platformy Android, chod a struktura operačního systému a vývoj aplikací pro OS Android. Popis a analýza problematiky indikátoru plynoměru, báze informací týkající se strojového vidění a popis implementace aplikace s řešením problému v jazyce Java.

# 1 ANDROID

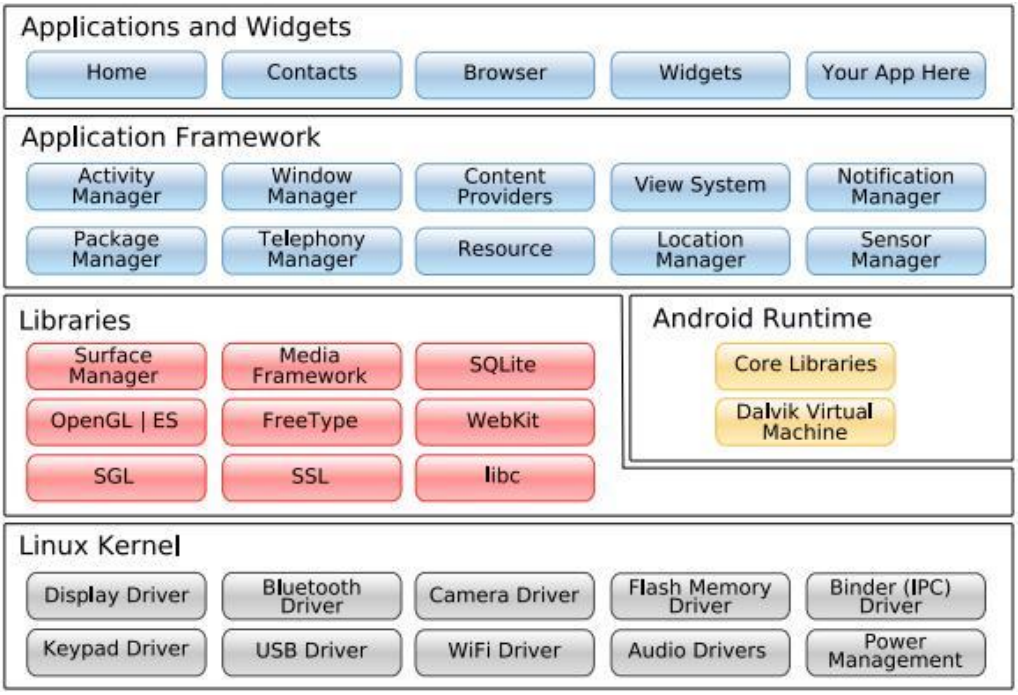
Open source operační systém pro přenosná zařízení jako jsou chytré telefony a tablety založený na Linuxovém jádře vznikl v roce 2003 se založením stejnojmenní společnosti Android Inc. Andy Rubin, Rich Miner, Nick Sears a Chris White se zaměřili na vytvoření chytřejšího telefonu, který splní nároky náročného uživatele. Do širšího povědomí se systém dostal v roce 2005, kdy byla společnost odkoupena Googlem, což zároveň podpořilo zrychlení vývoje. Po uvedení iPhone v roce 2007 bylo založeno konsorcium Open Handset Alliance (mimo Google s účastí firem Nvidia, Samsung, LG, HTC, Motorola, Intel, Qualcomm, Ebay, T-Mobile, Telefonica a další), jež představilo operační systém Android založený na otevřených standardech. První mobilní telefon se systémem Android se dostal do prodeje na konci roku 2008, jednalo se o T-Mobile G1 vyrobený firmou HTC. Android je upraven pro specifické užití 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. [1]


*Obrázek 1. Logo systému Android [2]*

## 1.1 Architektura systému Android

Komunikace systému Android probíhají prostřednictvím jednotného API, takto mohou aplikace přistupovat k funkcím operačního systému a funkcím zařízení (GPS modulu, akcelerometru, displeji a podobně). Virtuální mašina Dalvik VM, která je obdobou JAVA VM, zajišťuje běh aplikací. Aplikace jsou skutečně vyvíjeny v jazyce JAVA, avšak následně přeloženy pro virtuální mašinu, která je optimalizovaná pro běh na mobilních zařízeních.

Licencová politika, tedy GPL licence Linuxového jádra včetně provedených úprav a licence Apache 2.0 zbývající části operačního systému zpřístupnila zdrojový kód komukoliv a umožnila značnému rozšíření systému mezi výrobci telefonů.




*Obrázek 2. Grafické znázornění architektury systému Android [3]*

### 1.1.1 Charakteristické vlastnosti systému Android

Charakteristikou Androidu je podpora multitaskingu. Každá aplikace běží ve vlastní virtuální mašině, v samostatně odděleném procesu. Běh aplikace je tak systémem zajištěn i nadále, když uživatel pošle aplikaci na pozadí. Aplikace se mohou nacházet v několika různých stavech, které jsou přepínány dle potřeby uživatele, kapacity a dostupné operační paměti (jedná se o stavy uspané, běžící, nebo zastavené). Princip stavů měl fungovat bez nutnosti zásahů uživatele, avšak od verze Androidu 2.3 byl implementován správce úloh, aby výrobci zamezili nevítaným instalacím správce úloh třetích stran. Otevřenost systému je další významnou charakteristikou, která kupříkladu zpřístupněním grafického rozšíření uživatelského rozhraní umožnila výrobcům i koncovým zákazníkům lišit se od ostatních mobilních zařízení s totožným operačním systémem. Jako příklad uvádím srovnání rozhraní HTC Sense a TouchWiz od Samsungu, viz obrázek 3. [4]


*Obrázek 3. HTC Sensevlevo a TouchWiz vpravo [5]*

### 1.1.2 Linux Kernel

Počínaje nejnižší vrstvou architektury systému Android, jádrem systému je Linux kernel verze 2.6, které je obohaceno o specifické doplňky. Pro systém správy paměti, který je agresivnější s ohledem na zachování paměti (dat vybraných aplikací) se jedná o doplněk *Wakelocks*. Dalším významným doplňkem pro Android je ovladač Binder IPC, jež umožňuje vyšší Vrstvě API komunikaci se službami systému.

### 1.1.3 Knihovny

Nativní knihovna slouží zařízení ke zpracování různých typů dat. Knihovny jsou napsány v jazycích C a C++ a jsou specifické pro konkrétní hardware. Významné knihovny jmenovitě:

- Surface Manager - zajišťující přístup k subsystému displeje
- Media Framework - soubor kodeků umožňující záznam a přehrávání různých formátů médií
- SQLite - jedná se o databázový engine pro účely ukládání dat
- WebKit - slouží k zobrazení obsahu HTML
- OpenGL - vykreslování dvou a třídimenzionální grafiky na obrazovce.

### 1.1.4 Android Runtime

Android Runtime se sestává ze dvou částí - Dalvik Virtual Machine a knihovny Java Core. Dalvik VM je obdobou Java Virtual Machine použitou v zařízeních Android ke spouštění aplikací s optimalizací na nízkou spotřebu výpočetního výkonu a paměťových prostředků. Na rozdíl od Java VM souborů s příponou class spouští Dalvik VM soubory s koncovkou dex, které jsou vytvořeny kompilací class souborů poskytující vyšší účinnost

v prostředí s nižšími výpočetními prostředky. Dalvik VM umožňuje současné vytvoření několika různých instancí virtuální mašiny se zajištěním izolace, bezpečnosti, správy paměti s podporou vláknování. Application Framework

Jedná se o aplikacemi přímo integrovatelné bloky. Jejich úkolem je správa základních funkcí telefonu - správa hlasových hovorů, hospodaření s prostředky aj. Z pohledu vývojáře se jedná o základní nástroje pro tvorbu aplikací.

Application frameworks jmenovitě:

- Activity Manager - řízení životního cyklu aplikace
- Content Providers - správa sdílení dat mezi aplikacemi
- Telephony Manager - správa hlasových služeb, volán, pokud aplikace přistupuje k hovorům
- Location Manager - správa polohy zařízení za účasti vestavěného GPS modulu a mobilní sítě
- Resource Manager - správa různých typů prostředků užívaných aplikacemi

### 1.1.5 Applications

Jedná se o nejvyšší vrstvu architektury Androidu. Základní aplikace jsou již před instalovány na každém zařízení. Pro vývojáře aplikace je umožněno přistupovat k veškerým funkcím Androidu.

## 1.2 Základní části aplikace Android

Každá aplikace běžící na systému Android se sestává s částí popsaných v následujících podkapitolách.

### 1.2.1 Aktivity

Aktivita je tvořena jednou obrazovkou s uživatelským rozhraním. Pro ilustraci aplikace elektronické pošty se může skládat z aktivity pro náhled příchozí pošty, dále aktivity pro psaní zprávy a aktivity pro čtení vybrané zprávy. Aktivity pracují pohromadě s ohledem na komfort uživatele. Zároveň je každá aktivita na ostatních nezávislá. Pokud to aplikace povoluje, může být konkrétní aktivita volaná z jiné aplikace. Například aplikace fotoaparát může spustit aktivitu elektronické pošty - psaní nové zprávy, za účelem pohodlného sdílení fotografie.

### 1.2.2 Služby

Zde se jedná o komponentu, která běží na pozadí, vykonává dlouhotrvající operace nebo činnost vzdálených procesů. Služba neposkytuje uživatelské rozhraní. Příkladem služby je funkce přehrávat hudbu na pozadí, zatímco uživatel pracuje s jinou aplikací. Dalším využitím je například sběr dat prostřednictvím sítě, aniž by docházelo k blokaci interakce

uživatele s jinou aktivitou. Aktivita může službu spustit, nebo ji k sobě přiřadit, aby s ní mohla komunikovat.

### 1.2.3 Poskytovatelé obsahu

Správa sdíleného souboru aplikačních dat je úkolem Poskytovatele obsaku, ať se jedná o data uložena v souborovém systému, v databázi SQLite, na internetu, či jiném perzistentním uložišti, ke kterému může aplikace získat přístup. Prostřednictvím Poskytovatele obsahu mohou aplikace tato data požadovat, nebo dokonce jejich obsah měnit (pokut to poskytovatel obsahu povolí). Systém Andorid například spouští 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.

### 1.2.4 Přijímače vysílání

Broadcast receiver, tedy přijímač vysílání je komponentou, který reaguje na vysílání oznamování. Řada oznámení pochází ze systému – například oznámení vypnutí obrazovky, nízkého stavu baterie, pořízení obrázku. Aplikace může zahájit vysílání, aby dala ostatním aplikacím echo – například informaci o dokončeném stahování dat do zařízení. Aplikace jsou tak srozuměny, že jsou data připravena k použití. Vysílání nepoužívá uživatelské rozhraní, ovšem může vytvořit oznámení ve stavové liště jako notifikaci pro uživatele, ve chvíli kdy událost vysílání nastane. Obvykle je příjem vysílání bránou k jiným komponentám, kdy je možnost zajistit vykonání určité služby na základě vysílané události. Přijímač vysílání se implementuje jako podtřída BroadcastReceiver a jednotlivé vysílání je doručeno jako objekt *Intent* (záměr). [6]

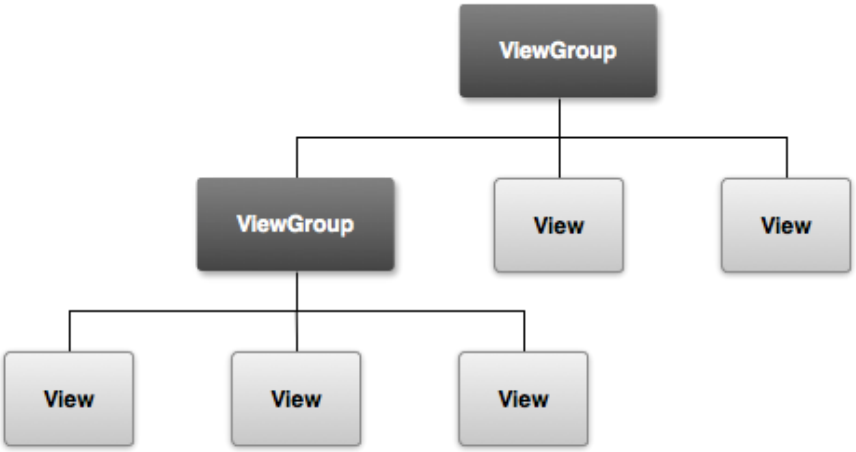
### 1.2.5 Další komponenty systému Android

Vedle výše zmíněných existují ještě další komponenty, které usnadňují vývoj aplikací:

- Layouts – obecná deklarace grafického uživatelského rozhraní obrazovek aplikace definovaná v jazyce XML
- Manifest – soubor v kořenovém adresáři projektu obsahující základní inforamce o aplikaci. Nejčastěji definuje základní předpoklady pro běh aplikace, jako minimální úrovně API, dále oprávnění aplikace k přístupu k systémovým komponentám (např. zabudovaný modul fotoaparátu)
- View a ViewGroup – jsou objekty, představující grafické uživatelské rozhraní aplikace jak znázorňuje obrázek níže. View objekty představují UI widgety jako

například textová pole (EditText), nebo tlačítka (Button). Objekty ViewGroup jsou neviditelné kontejnery, definující seskupení potomků View.

- Resources – zdroje uloženy v podadresářích aplikace (res/, obsahují ilustrace, texty, jazyky a jejich parametrů.



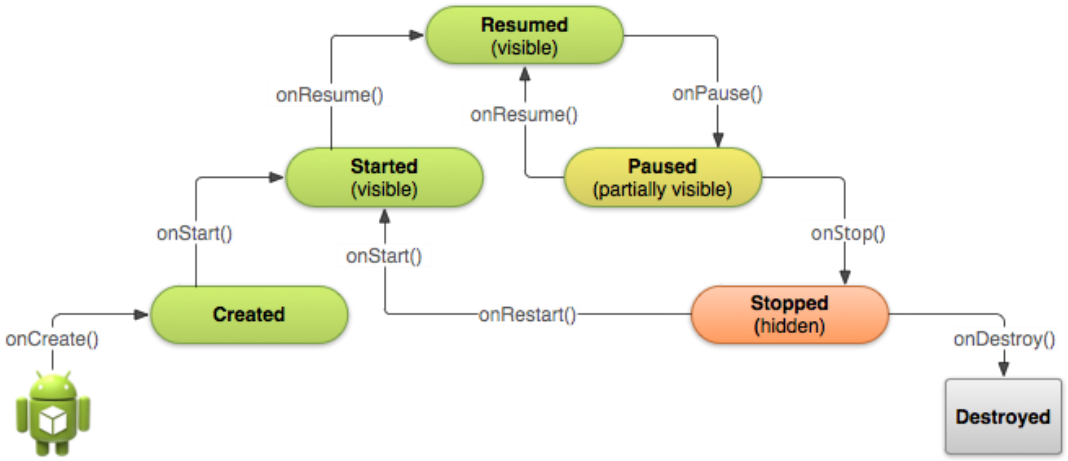

- *Obrázek 4. Ilustrace objektů ViewGroup a potomků View [7]*

### 1.2.6 Správa aplikace

Pokud po spuštění aplikace uživatel přepne na jinou, nebo jen vypne podsvícení displeje, instance aktivity vlastní aplikace přechází mezi jinými stavy v rámci svého životního cyklu. Po prvním spuštění aktivita přechází do popředí systému a získává uživatelovu pozornost. V průběhu procesu ovšem systém android volá řadu metod životního cyklu aktivity, proto je nutno mít dopředu nakonfigurovány komponenty a grafická uživatelská rozhraní.

### 1.2.7 Životní cyklus aktivity

V průběhu trvání aktivity jsou systémem sady metod životního cyklu volány v sekvenci, připomínající stupňovitou pyramidu jak znázorňuje obrázek níže. Každá fáze životního cyklu aktivity představuje samostatný krok na diagramu. Ve chvíli vytvoření nové instance aktivity, každé volání metod posouvá stav aktivity o jeden krok směrem k vrcholu diagramu. Na vrcholu diagramu je stav *Resumed*, kdy aktivita běží na popředí a uživatel s ní může komunikovat. Po opuštění aktivity systém volá metody posouvající stav aktivity v diagramu směrem dolů, za účelem zničení aktivity.




*Obrázek 5. Grafické znázornění životního cyklu aktivity [8]*

### 1.2.8 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 jinou 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. [8]

# 2 SOFTWAROVÉ VYBAVENÍ POTŘEBNÉ PRO PRÁCI

## 2.1 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 VirtualMachine). 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 DevelopmentKit) pro konkrétní platformu a jako IDE (Integrated Development-Environment) se nejčastěji používají nástroje: Netbeans, Eclipse a nově také Android Studio. [9]

## 2.2 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 DevelopmentTools). ADT umožňuje vytvořit navíc i aplikační UI, přidávat balíky založené na Android Framework API, debugovat aplikace pomocí SDK tools (Standard DevelopmentKit) a také exportovat apk soubory.

K instalaci ADT je nutná Eclipse a také Android SDK. 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. [10]

## 2.3 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 rámciBSD open-source licence. Knihovna podporuje široké spektrum platforem. 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.

# 3 VÝVOJ APLIKACE

V době započetí činnosti na diplomové práci, byl Eclipse IDE s doplňkem Android DevelopmentTools (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 nahrazen z pozice primárního vývojového nástroje Eclipse 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.

## 3.1 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, jež jsou v aplikaci použity z důvodu umožnění rychlého návratu na domovskou obrazovku.

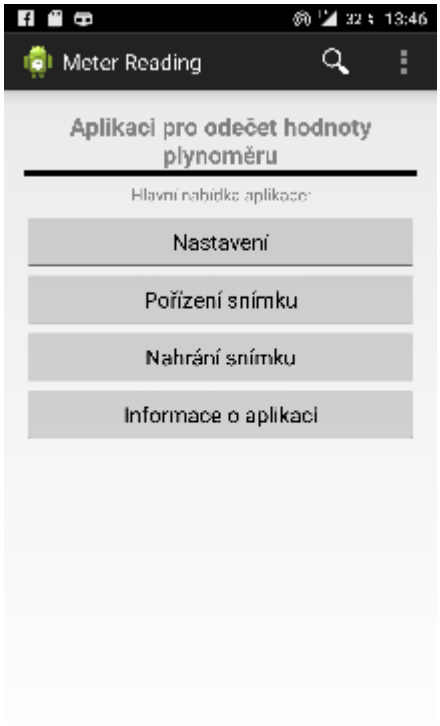
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myfirstapp2"
    android:versionCode="1"
    android:versionName="1.0" >

    <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 6. Ukázka souboru AndroidManifest*

## 3.2 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 7. 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.

### 3.2.1 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.

### 3.2.2 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.

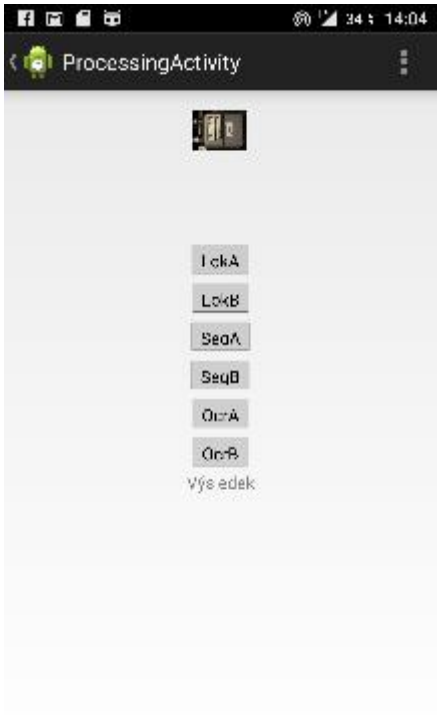

*Obrázek 8. Spuštěná aktivita Pořízení snímku*

### 3.2.3 Aktivita Zpracování

Aktivita zpracování je nejdůležitější aktivitou vytvořené aplikace. Bezprostředně po inicializaci aktivity *Processingactivity* dojde v rámci metody životního cyklu *onCreate*k volání implementované funkce *setInput ()*, jenž slouží k zobrazení obrázku z předchozí aktivity. V novém okně je nabídka metod, jež aplikace prování se vstupním obrazem.

Metody byly uvedeny ve zkratkách vzhledem k prostorovým možnostem aktivity a představuj:

- LokA: lokalizaci číselníku první variantou.
- LokB: lokalizaci číselníku druhou variantou.
- SegA: Segmentace číslic první metodou.
- SegB: Segmentace číslic druhou metodou.
- OcrA: Rozpoznávání číslic první metodou.
- OcrB: Rozpoznávání číslic druhou metodou.




*Obrázek 9. Ukázka aplikace s rozpoznáním celočíselného údaje*

# 4 ZPRACOVNÍ OBRAZU

## 4.1 Předmět zpracování

V této podkapitole popíši předměty a vstupy, které byly východisky pro zpracování obrazu.

### 4.1.1 Plynoměry

Plynoměr je specializovaný průtokoměr používaný k měření objemu proudícího množství přírodního plynu a propanu. Plynoměry jsou používány v obytných domácnostech nebo průmyslu, kde se konzumuje plyn. Plyn je mnohem náročnější na měření než kapaliny, protože měření je mnohem více ovlivňováno teplotou a tlakem. Plynoměr měří definovaný objem bez ohledu na velikost tlaku a množství proudící skrz plynoměr. Teplota a tlak musí být následně kompenzována, aby mohla být zjištěna skutečné množství plynu proudící plynoměrem.

**Indikační zařízení**

Jakýkoliv plynoměr může být osazen širokou škálou indikátorů. Nejběžnější indikátory ručičkové a digitální. V české republice jsou používány zejména ty digitální.

**Digitální indikátor**

Na obrázku je zobrazen digitální indikátor plynoměru používaný v domácnostech. Hlavní Informace, které chceme získat z indikátoru plynoměru je naměřený objem, jednotku a identifikační číslo plynoměru znázorněný čárovým kódem. Aplikace pro čtení z číselníku plynoměru by měla tyto hodnoty bezpečně rozpoznat a dále by měla odolávat nepříznivým podmínkám, jako jsou nepříznivá intenzita světla, špatný snímaný úhel, odlesky a matoucí informace.

Malá intenzita světla má za následek nižší kontrast a větší šum a způsobuje komplikace při detekcí hran což je u algoritmů na rozpoznávání čísel klíčové. Indikátor plynoměru kromě množství spotřebovaného plynu a identifikátoru obsahuje texty a čísla a popřípadě další čárové kódy, které chceme v aplikaci při rozpoznávání eliminovat. Aplikace by také měla reagovat na různé typy plynoměrů s odlišnými tvary, velikostí a designovým řešením indikátoru.


*Obrázek 10 Design plynoměrů*

### 4.1.2 Čárové kódy

Užití čárových kódů patří mezi nejpopulárnější metody automatického zadávání dat. Jedná se o vzor tvořený rovnoběžnými pruhy a mezerami různé šíři uspořádaný ve specifické předem definované struktuře, jehož účelem je reprezentovat čísla, písmena, či znaky. Informace v čárovém kódu je obsažena v relativní tlouště a vzdálenosti jednotlivých proužků a mezer a do použitelné formy je konvertována za použití skeneru. Čárové kódy racionalizovaly proces zadávání dat, ty lze zadávat rychleji a za vyšší přesnosti, než při ručním vkládání. Byly navrženy pro strojové čtení a jejich rozšíření je zásluhou nízké citlivosti na chyby při vstupu dat. S mírou chybovosti jedna ku dvěma milionům jsou považovány za nejefektivnější metodu zadávání dat. Implementace čárových kódů má krátkou dobu návratnosti a stala se tak nejpopulárnější metodou záznamu a přenosu dat. Mezi klady lze jmenovat jednoduchosti tisku a rozmanitost metod značení, jazykovou nezávislost, nízkou chybovost, bezkontaktní čtení. Uplatnění lze nalézt ve všech oblastech podnikání od správy materiálu po správu výrobků a zboží. V současné době dochází již také ke správě informací identifikací dokumentů. [11]

Standardní čárové kódy jsou čteny zdrojem světla, pohybem čtecí hlavy směrem ke kódu, či přiblížením kódu k fixované čtecí hlavě. Zdrojové světlo je absorbováno černými pruhy a odráženo mezerami mezi nimi ve specifické sekvenci, která je určena krajními elementy signalizující start a stop. Snímač detekuje odrážené světlo z mezer a produkuje silný výstup pro mezeru a slabý výstup pro černý pruh, kde bylo světlo pohlceno. Doba vysokých a nízkých signálů indikuje šíři pruhů a mezer. Referenční šíře zvaná „x“ dimenze je nejužším elementem kódu. Pro následné dekódování je potřeba určit symboliku čárových kódů. Na jejím základě je čárový kód přeložen na vlastní obsah a informace přenesena k dalšímu zpracování. [11]

K nejpoužívanějším symbolikám čárových kódů patří:

- Code 39 a Code 39 Mod 43
- U.P.C. A
- UPC E0 a UPC E1
- EAN 13 a EAN 8
- Code 93
- Interleaved 2/5 a Interleaved 2/5 Mod 10
- Code 128
- Codabar
- MSI [12]

### 4.1.3 Symbolika Code 128 [13]

V aplikaci byla implementována schopnost čtení čárových kódů dle symboliky Code 128, která je v tuzemsku použita pro většinu plynoměrů. V následující podkapitole popíši její specifikaci a postup dekódování.

Symbolika Code 128 umožňuje kódovat 128 znaků ASCII tabulky, kontrola parity každého kódovaného znaku spolu s kontrolou cheksum hodnoty představuje efektivní možnost kódování relativně velkého množství dat na malém prostoru.

**Struktura symbolu**

Předpokládejme, že pruh značíme číslem 1 a mezeru číslem 0. Šířka jednoho elementu je dimenzí „x“ (definovaná nejužším elementem kódu). Níže uvedený obrázek tedy přeložíme jako černý pruh, černý pruh, mezera, černý pruh, tedy 1101.


*Obrázek 11 Segment čárového kódu [13]*

Paritu jednotlivých bloků kódu vyjadřuje jejich konstantní délka 11 x, přičemž se skládá ze 3 pruhů a 3 mezer. Tabulka níže je ukázkou kódování některých znaků. Jak je patrno, symbolika pracuje se třemi sadami znaků, přičemž znaky START s hodnotou 103, 104, 105 umožňují přepínat mezi jednotlivými sadami. Sada B doplňuje set o malá písmena, sada C reprezentuje číselné hodnoty od 00 do 99.

| **Hodnota** | **Reprezentující ve znakové sadě:** | | | **Kódování** |
|---|---|---|---|---|
| | **Sada A** | **Sada B** | **Sada C** | |
| 103 | START A | START A | START A | 11010000100 |
| 104 | START B | START B | START B | 11010010000 |
| 105 | START C | START C | START C | 11010011100 |
| - | STOP | STOP | STOP | 11000111010 |
| 33 | A | A | 33 | 10100011000 |
| 34 | B | B | 34 | 10001011000 |
| 65 | SOH | a | 65 | 10010110000 |
| 67 | ETX | c | 67 | 10000101100 |
| 99 | Code C | Code C | 99 | 10111011110 |

*Tabulka 1 Ukázka dekódovací tabulky [13]*

### Struktura čárového kódu symboliky Code 128

Předmětný čárový kód se sestává z úvodní tiché zóny, kódu start, datového obsahu, kódu stop, ukončovacího pruhu a koncové tiché zóny.

0. Úvodní tichá zóna
1. kódu Start s hodnotou (103 - 105) nastavuje znakovou sadu kódu, může být rovněž použit v datovém obsahu jako přepínač znakové sady.
2. Datový obsah představují kódové bloky jednotlivých znaků zprávy. Na konci datového obsahuje je navíc blok představující checksum hodnotu.
3. Stop kód 11000111010
4. Ukončovací pruh s kódem 11
5. Konečná tichá zóna

### Checksum kontrola

Hodnota Checksum je vypočteno na základě váženého součtu hodnot kódových bloků počínaje znakem Start až do konce Datového obsahu. Kód výsledná hodnoty je zároveň připojen na konec datového obsahu před Stop kód.

Uvažujeme-li váhu znaku Start rovno jedné a následně váhu každé pozice symbolu datového obsahu rovno dané pozici v datovém obsahu, hodnota checksum je vypočtena dle vzorce:

$$MODUS\ (\frac{Sh + \sum_{n=1}^{n} Dn.n}{103})$$

Kde Sh představuje hodnotu kódu Start

Dn představuje hodnotu bloku datového obsahu na pozici n

n počet pozic datového obsahu

*Vztah 1 Výpočet hodnoty checksum*

Pro ilustraci uvádím výpočet hodnoty checksum pro čarový kód:


*Obrázek 12 Čárový kód použitý v příkladu [13]*

| **Symbol** | **Start B** | **H** | **I** | **Sada C** | **34** | **56** | **78** |
|---|---|---|---|---|---|---|---|
| **Hodnota** | 104 | 40 | 41 | 99 | 34 | 56 | 78 |
| **Pozice symbolu** | - | 1 | 2 | 3 | 4 | 5 | 6 |
| **Výpočet** | 103 | 40 * 1 | 41 * 2 | 99 * 3 | 34 * 4 | 56 * 5 | 78 * 6 |
| **Sčítanec** | 103 | 40 | 82 | 297 | 136 | 280 | 468 |

*Tabulka 2 Výpořet checksum [13]*

Celočíselný zbytek podílu součtu všech sčítanců (104 + 40 + 82 + 297 + 136 + 280 + 468 = 1407) a hodnoty 103 (1407 / 103 = 13) činí 68. Symbol pro zjištěnou hodnotu checksum (10000100110) je umístěn na konec datového obsahu. Popis kódování uvedeného čárového kódu vypadá následovně:


*Obrázek 13 Dekódování jednotlivých polí [13]*

1. kód Start B: 11010010000
2. znak „H“ kódován jako: 11000101000
3. znak „I“ kódován jako: 11000100010
4. změna na znakovou sadu C: 10111011110
5. číslo "34" kódováno jako: 10001011000
6. číslo "56" kódováno jako: 11100010110
7. číslo "78" kódováno jako: 11000010100

8. hodnota checksum "68" kódována jako: 10000100110
9. kód Stop: 11000111010
10. ukončovací pruh se symbolem: 11

## 4.2 Počítačové vidění

Zpracování obrazu je formou 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. [14]

## 4.3 Vybrané metody počítačového vidění

V následující kapitole se věnuji metodám zpracování obrazu, jež byly použity k řešení problémových oblastí.

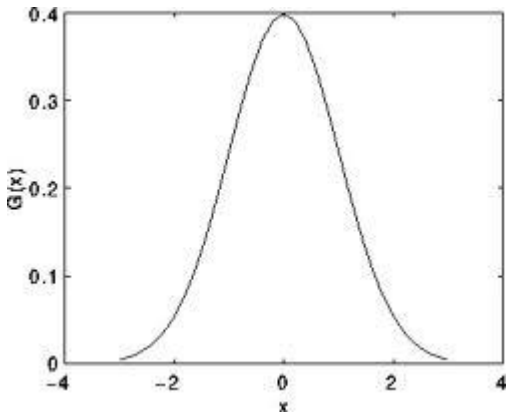
### 4.3.1 Segmentace obrazu

Představuje skupinu postupů k dosažení automatického separaci požadované scény, na oblasti se společnými oblastmi. Nejběžnější metoda segmentace obrazu je práhování,

jež představuje oddělení obrazu na dvě části s úrovní jasu vyšší než je jedna konkrétní hodnota a část s jasem nižším. Adaptivní práhování představuje pokročilou techniku prahování, umožňující stanovit práh v závislosti na lokálních parametrech obrazu. Metoda segmentace, jež 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.

### 4.3.2 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, což činí detekci obtížnější.




*Obrázek 14. 1D obraz, kde má bod ležící uprostřed největší váhu. Váha sousedních bodů směrem od středu obrazu klesá. [15]*

### 4.3.3 Houghova transformace

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. [16]

### 4.3.4 Houghova transformace přímek

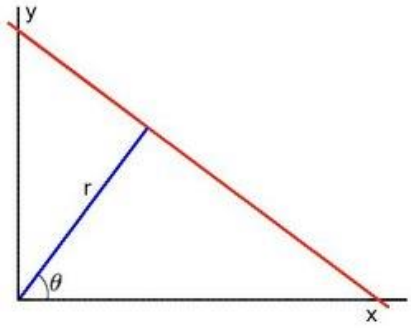
Při detekci přímek v obraze HT vychází za normálového tvaru analytického zápisu přímky, který je nezávislý na orientaci os souřadnic.

$$r = x\cos\theta + y\sin\theta$$

Kde r představuje velikost normály od počátku a

$\theta$ představuje velikost úhlu vůči ose x.

*Vztah 2 Normálový tvar analytického zápisu přímky*



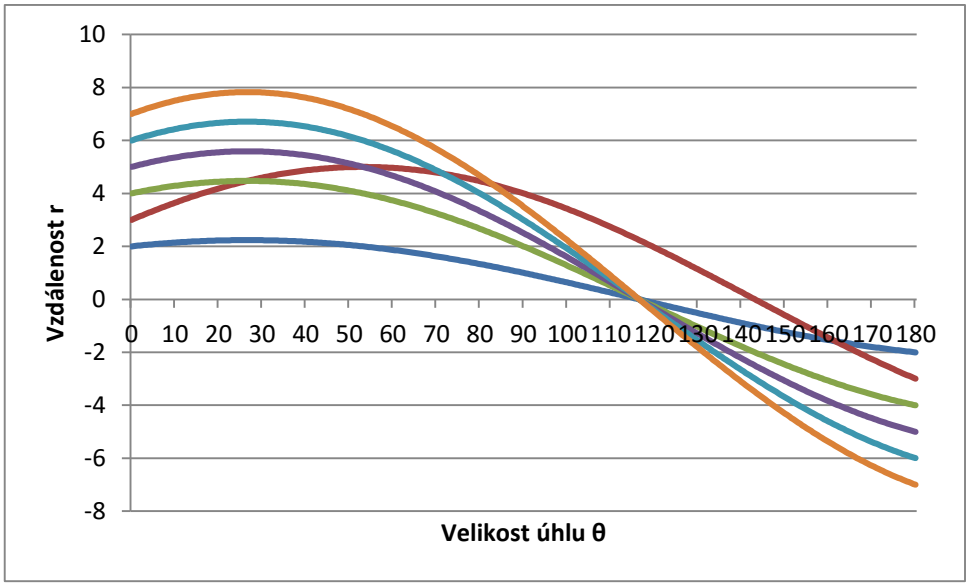

*Obrázek 15 Parametry normálového tvaru rovnice přímky [17]*

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:

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

*Tabulka 3 Zvolené body*




*Obrázek 16. 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.

### 4.3.5 Morfologické operace [18]

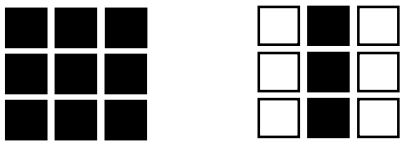
Morfologická filtrace obrazu pracuje s obrazy jako s bodovými množinami a používané postupy se dosti liší od dříve popisovaných principů konvoluce. Pro její popis se obvykle používá teorie množin a topologie, my se zde však pokusíme přiblížit vše srozumitelnějšími prostředky Booleovy algebry.

Při morfologických transformacích nepracujeme s konvolučním jádrem, ale s tzv. strukturním elementem a výsledný pixel je výsledkem (logické) operace mezi množinou mřížky strukturního elementu a množinou bodů zdrojového obrazu, ležících pod body strukturního elementu. Morfologické transformace lze provádět i s barevnými obrazy, nejdříve se ale zaměříme na operace s obrazy binárními.

Mezi základní operace binární morfologie patří:

- dilatace
- eroze

## Binární morfologie



*Obrázek 17 Strukturní elementy*

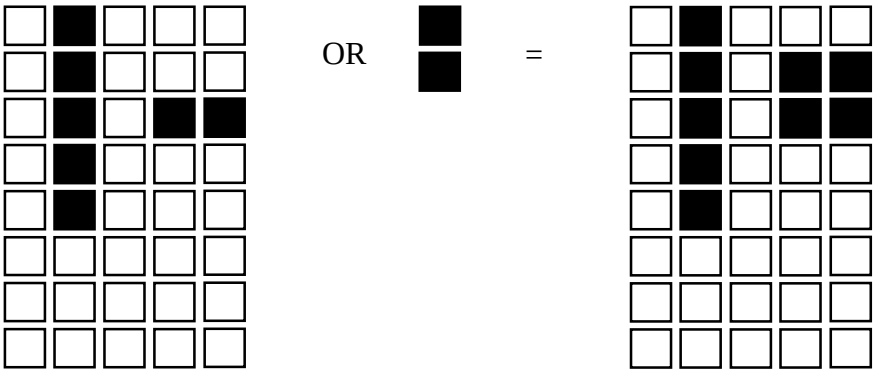
Binární morfologie pracuje s binárními obrazy (tj. se čtvercovými diskrétními bodovými mřížkami). Na začátku tedy již máme vstupní binární obraz, na kterém jsou jednotlivé černé objekty na bílém pozadí. Morfologické transformace pak mění tvar nebo strukturu (objekty se zvětšují, zmenšují, spojují, rozdělují nebo tříští na další objekty atd.) objektů. Můžeme je využít např. k odstranění nežádoucí informace z obrazu, která vzniká např. při prahování (o kterém již víme, že je značně problematickou operací).

Při morfologických transformacích budeme provádět logické operace mezi hodnotami buněk strukturního elementu a pixely s pozicemi příslušnými těmto buňkám. Strukturní elementy mohou vypadat jako na obr.

### Dilatace

Dilatace eliminuje izolované díry v objektech a rozšiřuje obrysy objektů na úkor okolního pozadí. Jestliže se pod jakoukoliv jedničkovou buňkou strukturního elementu nachází černý pixel, pak výsledkem operace je černý pixel.

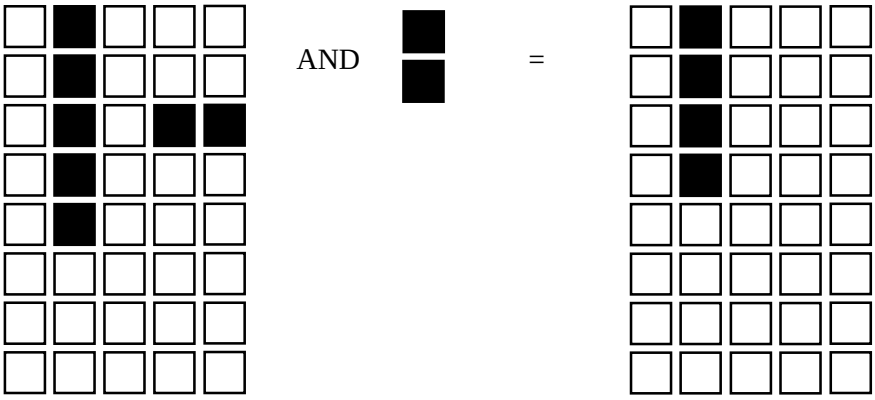
Dilatace je znázorněná následujícím vztahem, kde X je obraz, B je strukturním elementem.




*Obrázek 18 Grafické znázornění Dilatace*

### Eroze

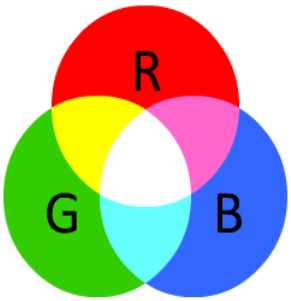
Eroze eliminuje izolované pixely na pozadí a ubírá obrysy objektů. Černý pixel je výsledkem operace jen tehdy, když jsou černé pixely současně pod všemi jedničkovými buňkami strukturního elementu, ve všech ostatních případech je výsledkem bílý pixel.




*Obrázek 19 Grafické znázornění Eroze*

### 4.3.6 Barevné prostory [19]

**RGB**




*Obrázek 20 Spektrum RGB [19]*

Barevný prostor RGB odpovídá fyziologii vnímání lidského oka, které má na tyto tři základní barvy speciální receptory na sítnici oka – tzv. čípky. Tyto tři základní barvy spektra lze snímat a zobrazovat v různé intenzi-tě. Jejich kombinace pak utvářejí všechny ostatní barevné odstíny, od černé (nulová intenzita všech tří barev) až po bílou (maximální intenzita všech tří barev).

Pro každou barevnou složku se rozlišuje 256 stupňů intenzity barvy, tato informace se ukládá do 8 bitů. Pro všechny tři barvy dostáváme 2563, což je 16 777 216 barevných odstínů v celkem 24 bitech (= 3 byty) na obrazový bod (pixel). Ne všechna výstupní obrazová zařízení jsou schopna takové množství barev současně zobrazit, proto někdy bývá počet barev před vykreslením snižován. Ovšem tak, aby lidské oko zaznamenalo co nejmenší ztrátu kvality obrazu. Barvy modelu RGB se uvádějí i v celočíselném rozsahu 0–255, kde 0 je 0 % (barevná složka není zastoupena) a hodnotě 255 odpovídá 100 % (barevná složka nabývá největší intenzity).

**HSV**

Barevné modely HSV a HSL se nejvíce přibližují systému používání barev malíři, neboť nové odstíny se vytvářejí přidáváním bílé (vznikají nádechy) nebo černé (vznikají odstíny) do základních spektrálních barev. Oba prostory definují barvu pomocí trojice složek, které však nepředstavují základní barvy. Základními parametry jsou:

- barevný tón (H – hue) – reprezentuje převládající spektrální barvu (barvy duhy) v rozsahu od 0 do 360 stupňů (bývá zobrazována do kruhu),
- sytost (S – saturation) – udává „čistotu“ barvy a bývá vyjádřena poměrem čisté barvy a bílé,
- jas (V – value) – vyjadřuje intenzitu barvy, stupeň zářivosti barvy (kolik světla odráží) a bývá vyjádřen poměrem čisté barvy a černé.

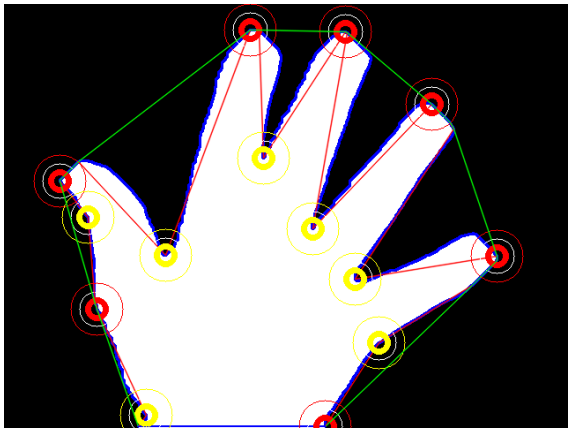
**Detekce hran**

Hrany jsou místa, kde se prudce mění jas a dají se využít na identifikaci důležitých oblastí (regionů) v obraze. Jsou vstupem mnoha algoritmů na detekci objektů v obraze, případně při segmentaci obrazu.

- Prudká změna jasu = hrana
- Lokální změna jasu = jasová hrana
- Globální změna = jasový hraniční segment
- Vlastnosti: velikost, směr

### 4.3.7 Kontury

Mezi hlavní úkoly počítačového vidění patří identifikace objektu v obraze. Co je pro člověka naprosto přirozené a nenáročné ovšem v počítačovém vidění patří k náročným úkonům, neboť neexistuje obecný algoritmus pro nalezení všech existujících typů objektů, který by navíc byl spolehlivý. Pokud ovšem je dáno, jaké konkrétní objekty v obraze hledáme a jejich geometrie je navíc jednoduchá (čísla, geometrické tvary, lidské tváře), pak lze tyto objekty za pomocí nalezených kontur identifikovat s vysokou spolehlivostí. Kontury jsou křivky spojující hraniční body objektu, mající například stejnou barvu či intenzitu. Pro lepší přesnost při hledání kontur objektů se využívá binárních obrazů, kde se uplatňuje prahová detekce hran. Právě tuto metodu používá funkce findContours, kterou obsahuje knihovna OpenCV.



*Obrázek 21 Modře vyznačené kontury ruky nalezené pomocí funkce findContour, knihovna OpenCV [20]*

**Funkce findContours**

Tato funkce slouží k nalezení kontur v binárním obraze, nalezené kontury vrací ve formě vektorů, ze kterých za pomocí vhodné selekce lze oddělit objekty od pozadí či šumu a za pomocí vzorů lze identifikovat konkrétní objekty (číslice).

[21]

## 4.4 Korekce pootočeného obrazu

Před obrazovou analýzou plynoměru, jako je detekce pole s údajem spotřebovaného množství plynu, segmentací samotných číslic nebo také detekcí čárového kódu s označením plynoměru, je velmi vhodné zajistit vodorovnou orientaci obrazu. Pořízení snímku ve vodorovné poloze je prakticky nemožné a označení plynoměru taktéž nemusí být orientováno stejně jako dominantní linie plynoměru. Z těchto důvodů je nutno provést autonomní korekci obrazu před příslušnou operací zpracování obrazu. V této práci byla implementována vlastní metoda pro korekci natočení, jež je založena na Houghovy transformaci.

Metodu se skládá z následujících kroků:

1. Převedení vstupního obrazu do stupňů šedi.
2. Zvýraznění hran s použitím Cannyho hranového detektoru.
3. Aplikace Houghovy transformace pro nalezení linií procházející detekovanými hranami.
4. Sestavení sloupcového grafu orientace nalezených linií ve stupních.
5. Úhel natočení odpovídá prvku histogramu s největší hodnotou.
6. Narovnání je realizováno nalezením středu vstupního obrazu, změny jeho orientace o odpovídající hodnotu natočení a oříznutí na původní rozměr.


*Obrázek 22 Fragment křivě vyfoceného plynoměru*


*Obrázek 23 Zvýraznění dominantních linií*


*Obrázek 24 Korekce natočení*

*Obrázek 25 Sloupcový diagram, jenž znáyorňuje četnost linií s příslušným úhlem svírajícím s vodorovným směrem*

## 4.5 Detekce pole hodnot plynoměru

V práci je potřeba jednoznačně určit pozici číselníku plynoměru. Hledání pole hodnot bylo založeno deduktivním oddělení číselníku a okolí na základě charakteristik číselníků, jenž jsou pro všechny běžné plynoměry společné.

Klíčové charakteristiky pro stanovení pozice pole hodnot plynoměru:

1. Pole hodnot plynoměru sestává ze dvou částí.
2. Celočíselná část číselníku představuje tmavé pole s pěti místním údajem v bílé barvě, desetinná část se nachází v těsné blízkosti celočíselného pole, obsahuje

bílý text s černým pozadím a je ohraničen červenou barvou, anebo obsahuje bílý text s červeným pozadím.

3. Barva v oblast nad i pod celočíselným i desetinným údajem je výrazně světlejší a většinou má bílou barvu.

Detekce pole hodnot plynoměru byla realizována dvěma metodami, které využívají zmíněné klíčové charakteristiky.

### 4.5.1 První verze nalezení číselníku

Běžné typy plynoměrů se mohu lišit v rozměrech, tvaru nebo designu, nicméně v okolí desetinného místa číselníku či mezi hodnotami v tomto poli se nachází červené značení, jež má obdélníkový tvar.

Postup nalezení číselníku v první verzi lze rozdělit do následujících kroků:

1. Nalezení úhlu natočení podle významných linií.
2. Korekce nakřivení pro zajištění horizontální orientace číselníku.
3. Nalezení shluků v obraze, jejichž chromatičnost odpovídá červené barvě.
4. Ohraničení nalezených segmentů scény.
5. Nalezení číselníku na základě třídění nalezených segmentů.

Při vycentrování obrazu (viz kapitola: 4.3.6) je nutno správně nastavit parametry pro hledání významných linií.


*Obrázek 26 Fragment vycentrovaného plynoměru*

Minimální délka linií byla pro správnou funkci stanovena jako osmina šířky obrazu a další parametr – maximální povolená délka mezi body nacházejícími se na jedné přímce jako sedmdesátina výšky obrazu. Po provedení korekce natočení plynoměru následuje nalezení červených míst v obraze. Segmentace obrazu na základě chromatičnosti je umožněna po převodu vstupního obrazu do roviny HSV. Barevný tón odpovídající barvě hledaného objektu se nachází ve dvou pásmech, a tak bylo potřeba provést segmentaci pro každý rozsah zvlášť a tyto obrazové výstupy pak sloučit s rovnocenným váhováním (převod do roviny HSV je realizován metodou *cvtColor* a váhování je implementováno funkcí *addWeighted*). Vzhledem k možným deformacím hledaného pole byla provedena morfologická operace (viz kapitola: 4.3.5) se strukturalizujícím elementem ve tvaru obdélníku.

*Obrázek 27 Porušené vymezení desetinné oblasti*

Takto upravený vstupní obraz je připraven k detekci kontur, jenž umožní lokalizovat a ohraničit všechny shluky obrazových bodů. Detekované kontury jsou posléze ohraničeny obdélníky s nejmenší plochou a tyto jsou dále filtrovány.

*Obrázek 28 Výsledný obraz po aplikovaci morfologické operace*

Dle proporcí desetinné oblasti je vybrán obdélník, jenž lokalizuje hledanou oblast. Celočíselná část číselníku je poté lokalizována dle proporcí desetinné oblasti.

### 4.5.2 Druhá verze nalezení číselníku

Předchozí metoda nemusí být dostatečně robustní, neboť v případě výskytu cizího tělesa se stejným barvovým podáním a proporcemi, jaké má desetinná část číselníku, nemusel by algoritmus pozici číselníku vyhodnotit korektně a bylo proto potřeba navrhnout alternativní metodu lokalizace číselníku. Tato metoda využívá navíc přechody barev černá a bílá v okolí číselníku.

Metoda sestává z následujících kroků:

1. Korekce zakřivení plynoměru.
2. Nalezení a ohraničení shluků v obraze, jejichž chromatičnost odpovídá červené barvě.
3. Nalezení horizontálních linií.
4. Výpočet jasového gradientu v okolí horizontálních linií.
5. Detekce číselníku na základě nalezení linií s největším a nejmenším jasovým gradientem v rámci ohraničené oblasti s červenou barvou.

Korekce natočení plynoměru a nalezení potenciálních umístění desetinné části údaje plynoměru bylo provedeno stejným způsobem jako v předchozí metodě. Vycentrovaný obraz byl opět předzpracován převodem do stupňů šedi a filtrován Cannyho hranovým detektorem a po aplikaci Houghovy transformace byly nalezeny všechny významné vodorovné hrany. V oblasti každé detekované kontury (obdélníkové ohraničení objektu červené barvy) byla sledována přítomnost vodorovných přímek a pro každou z těchto přímek byla

vypočtena diference hodnoty jasu v okolí přímky. Po převodu vstupního obrazu do prostoru „HSV“ je hodnota jasu pro každý pixel uvedena ve složce „V“. Absolutně bílá barva má hodnotu „V“ odpovídající dvěma stům padesáti pěti. Z tohoto důvodu má přímka nacházející se na přechodu bílá -černá nejvyšší hodnotu diference a přímka ležící na přechodu černá –bílá má pak diferenci nejnižší. V rámci každé detekované kontury byla nalezena dvojce přímek s největším rozdílem diferencí, a pokud byla vzdálenost těchto přímek větší než šířka příslušné kontury, tak byla považována jako horní a spodní hrana číselníku. Druhá metoda je robustnější, ale za cenu výrazně vyšší výpočetní náročnosti.


*Obrázek 29 Vyznačení výsledku lokalizace číselníků pro různé plynoměry.*

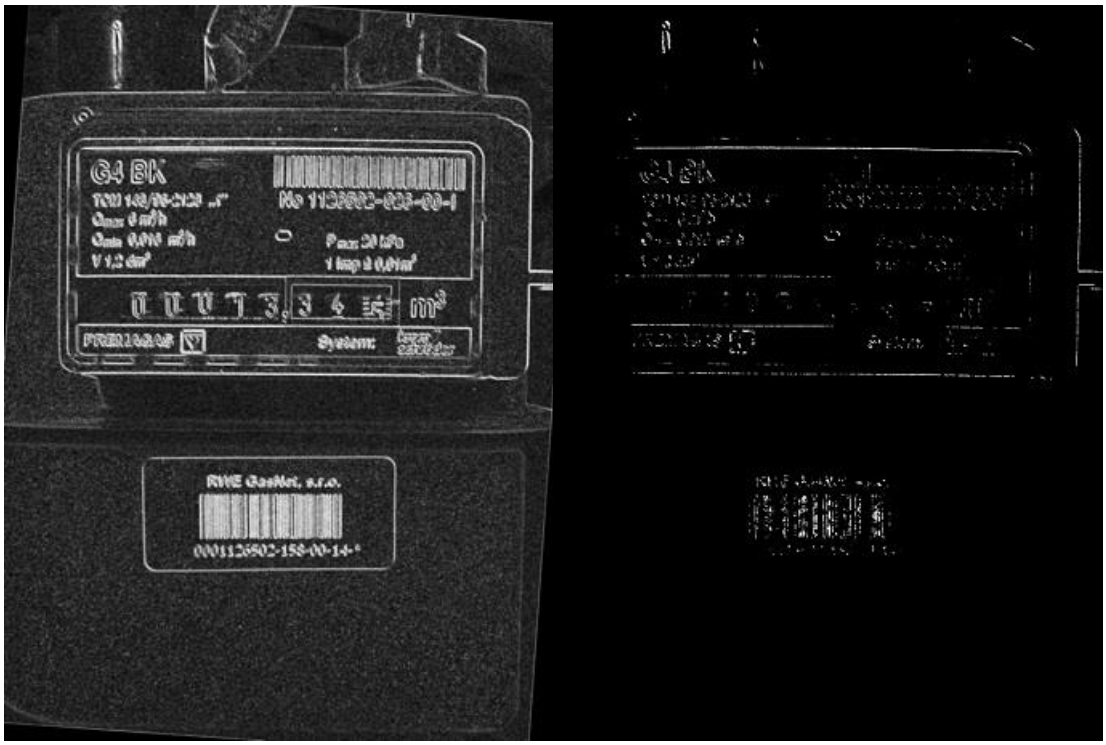
Druhá metoda nalezení číselníku je násobně pomalejší, nicméně její úspěšnost byla 91%.

### 4.5.3 Lokalizace čárového kódu [22]

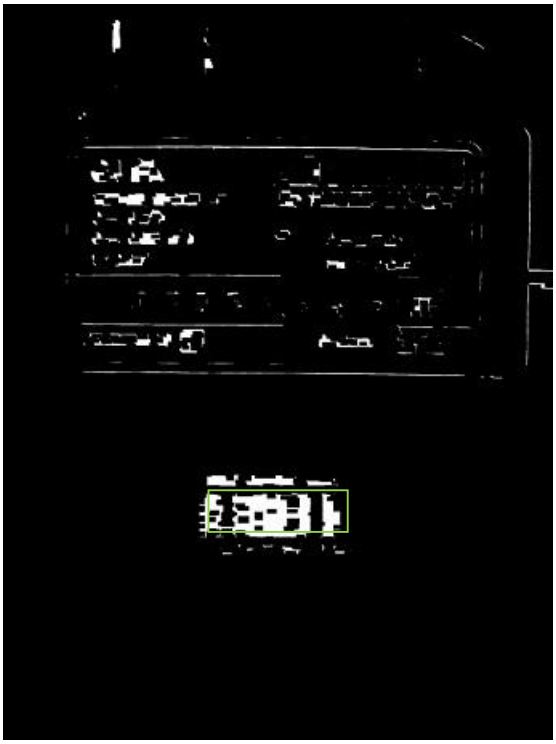
Pro lokalizaci čárového kódu v obraze bylo užito metody vyhledávání hran pomocí *Gradientu*.

1. Po převedení vstupního obrazu do stupňů šedi byla pro každý pixel vypočítaná absolutní hodnota rozdílu gradientu podle osy x a osy y. Cílem bylo znázornit oblasti s vysokým horizontálním gradientem a nízkým vertikálním gradientem – tyto charakterizují symbol čárového kódu ve vodorovné poloze.
2. Po aplikaci rozmazání, bylo na rozmazaný obraz aplikováno prahování, kdy bylo pixelům gradientového obrazu s hodnotou větší než 225 nastavena hodnota 255, ostatním nulová hodnota.
3. Pro odstranění vertikálních mezer z oblasti čárového kódu a usnadnění detekce požadované oblasti, byly dále aplikovány morfologické metody dilatace a eroze s nastavením morfologického kernelu tvaru obdélníku šířky 21 a výšky 7 pixelů.
4. Posledním krokem je vyříznutí symbolu čarového kódu z původního obrazu.


*Obrázek 30 Výchozí obraz pro zpracování a vytvoření gradientového obrazu v kroku 1*




*Obrázek 31 Rozmazání (vlevo) a aplikace práhování kroku 2*



*Obrázek 32 Upravený obraz po 3. kroku s vyznačením identifikovatelné oblasti*

### 4.5.4 Segmentace číslic plynoměru

Segmentace číslic představuje nalezení údaje plynoměru z pole hodnot, jehož úspěšná lokalizace byla popsána v předchozích řádcích. Číselník plynoměru obsahuje pět celočíselných míst a tři místa desetinná. Mechanismus plynoměru údaje mění ve vertikálním směru a z toho důvodu nemusí být čísla v jedné rovině. Korekci natočení již provádět není potřeba, neboť je pole s údaji orientováno stejně, jako dominantní linie na plynoměru. Nicméně pole s čísly je před segmentací potřeba předzpracovat. K segmentaci číslic byly vypracovány dvě metody.


*Obrázek 33 Lokalizované pole s údaji plynoměru*

### 4.5.5 Segmentace číslic plynoměrů založená na detekci kontur

Vstupem pro následující metodu segmentace číslic je barevný obraz pole hodnot plynoměru.

Postup se skládá z následujících kroků:

1. Odstranění červené barvy.
2. Předzpracování obrazu.
3. Provedení morfologické operace.

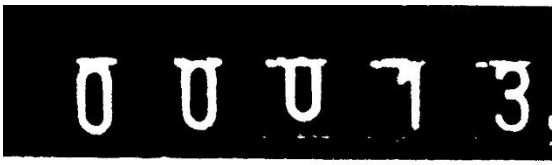
4. Detekce a vyhodnocení kontur.
5. Stanovení pozice číslic.

Metoda byla navržena tak, aby uměla vyhodnotit jak celočíselnou tak desetinnou část údaje plynoměru. Z toho důvodu bylo v prvním kroku provedeno potlačení červené barvy – všechny pixely v rovině *RGB* byly extrahovány, *R* složky veškerých obrazových bodů.

*Obrázek 34 Barevný obraz na vstupu*

*Obrázek 35 Zobrazení po potlačení červené složky dat obrazu*

Předzpracování obrazu bylo realizováno aplikací bilaterálního filtru, jenž zanechá obraz dostatečně ostrý, následovalo vytvoření binárního obrazu práhováním metodou OTSU, jenž umožňuje prahovat bimodální obraz.

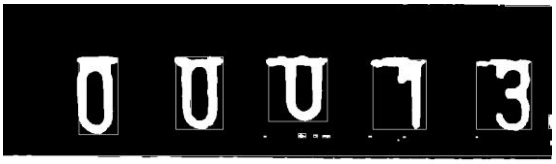


*Obrázek 36 Binární obraz číselníku*

Na takto připraveném obraze byla provedena morfologická operace se strukturalizujícím elementem obdélníku, který má rozměry proporčně odvozeny od výšky vyšetřovaného pole s údaji a poměr stran odpovídal poměru šířky a výšku čísel.

*Obrázek 37 Zvýraznění hledané informace morfologickou operací*

Po provedení shlukování významných objektů v obraze, byla provedena detekce kontur. Nalezené kontury byly ohraničeny nejmenším obdélníkem každá zvlášť, z množiny nalezených obdélníku byly odstraněny ty, jejichž obsah nesplňoval požadované minimum a také byly odstraněny obdélníky vnitřních křivek číslic (nejčastěji v číslici nula).



*Obrázek 38 Ohraničení kontur*

Zbývající obdélníky opisující detekované kontury byly seřazeny vůči vodorovné ose a na základě jejich souřadnic byly z pruhovaného obrazu segmentovány nalezené číslice.

*Obrázek 39 Segmentované číslice*

Od hledaných segmentů není možné jednoznačně separovat rušivá tělesa, atak byla vytvořena přesnější metoda.

### 4.5.6 Segmentace číslic pomocí horizontální a vertikální projekce

Segmentaci číslic bylo potřeba navrhnout tak, aby byly jejím výstupem normalizované segmenty číslic vhodní k dalšímu zpracování. Zostření ani adaptivní práhování se neosvědčilo, neboť do scény zanášelo nežádoucí šum. Vzhledem k jednoznačnému oddělení desetinného a celočíselného byla segmentace realizována funkcemi zvlášť pro obě části údaje k dosažení vyšší přesnosti.

Postup metody lze rozdělit do následujících kroků:

1. Předzpracování obrazu.
2. Vytvoření binárního obrazu scény.
3. Sestrojen vertikální projekce.
4. Analýza dat získaných po provedení vertikální projekce.
5. Vymezení pozice hledaných čísel v horizontální rovině.
6. Aplikace morfologické operace k určení pozic číslic ve vertikálním směru.
7. Vytvoření středového výseku ke stanovení přibližných pozic hledaných čísel.
8. Hledání, setřídění a slučování kontur.
9. Vytvoření výseku jednotlivých číslic.

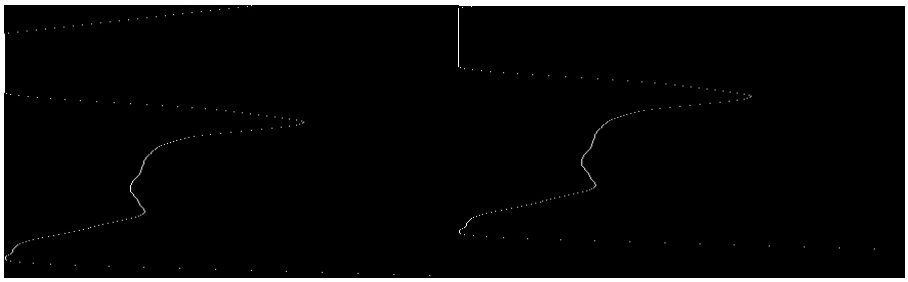
Segmentace číslic byla realizována analýzou binárního obrazu číselníku. Nejlepší výsledky podávala sekvence bilaterální filtrace a prahování metodou OTSU, jejíž výstupem je binární obraz s minimálním množstvím šumu.

*Obrázek 40 Binární obraz číselníku*

Ve výstupním obraze jsou kromě hledaných čísel zřetelné cizí elementy způsobeny nečistotou, odlesky či šumem na obrázku. Vzhledem ke skutečnosti, že se hledané čísla nachází z většiny své výšky v jedné rovině, bylo rozhodnuto o vytvoření horizontálního průmětu obrazu.

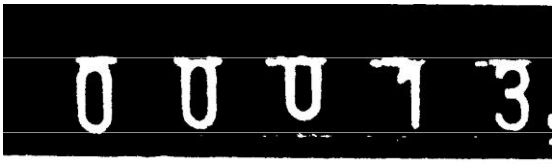
*Obrázek 41 Horizontální projekce číselníku*

Z analýzy horizontálního průmětu číselníku vyplývá, že se vodorovný výsek číslic nachází v okolí střední hodnoty sloupcového grafu.



*Obrázek 42 Sestavení dopředného a zpětného klouzavého průměru*

Jednoznačné lokalizování spodní linie vymezující hledaná čísla byla realizována analýzou klouzavého průměru ve směru svrchu dolů. Analýza počíná v prostřední části grafu a po nalezení prvku, jehož hodnota klesne o hodnotu větší, než procento šířky projekce je možné stanovit pozici spodní linie k segmentaci znaků. Obdobným způsobem probíhá lokalizace svrchní linie k určení výřezu segmentů, kdy byl však vypočten klouzavý průměr ve směru zespodu nahoru. Analýza znovu započala v prostřední části diagramu (ve vertikálním směru) a první prvek, jehož hodnota klesla pod stanovenou mez odpovídala ypsilonové souřadnici přímky, jež ohraničuje segmenty číslic z vrchní strany.



*Obrázek 43 Horizontální vymezení číslic*

Výsledek segmentace v horizontálním směru je patrný na binárním obraze. Vzhledem ke skutečnosti, že se na analyzované scéně mohou vyskytovat odlesky, které mají prakticky totožné charakteristiky jako hledané čísla, není možné jednoznačně stanovit pozici vertikálních linií k segmentaci číslic. Na diagramu vertikální projekce lze vidět, že umístění čísla jedna není možné deterministicky oddělit od rušivých elementů.

*Obrázek 44 Vertikální projekce binárního obrazu číselníku*

Další možností je analýza obrazu po provedení výseku v horizontální rovině, nicméně ve vertikální projekci pozici tohoto výseku není možné jednoznačně stanovit. Po převedení horizontální projekce do vodorovného sloupcového diagramu je poloha výseku přibližně patrná, nicméně k jednoznačnému určení polohy výseku nepostačuje.

*Obrázek 45 Kombinace vertikální a horizontální projekce číselníku*

Další možností je provedení výseku ve vertikálním směru původního binárního obrazu, nicméně není možné stanovit šířku výseku ke správnému posouzení středu číslic.

*Obrázek 46 Středový výsek číselníku*

Řešením předcházejícího problému bylo aplikování morfologické operace na ohraničeném obraze, přičemž byly nejlepší výsledky dosaženy po použití elipsoidního strukturalizujícího elementu (vzhledem k přítomnosti oblých tvarů v číslech) jehož výška musela být větší než šířka a byla odvozena od výšky výseku.

*Obrázek 47 Aplikace morfologické operace s obdélníkovým strukturalizujícím elementem*

Po provedení morfologické operace byl vytvořen výsek prostřední třetiny obrazu ve vertikálním směru s následnou detekcí kontur.

*Obrázek 48 Provedení středového výseku po aplikaci morfologické operace*

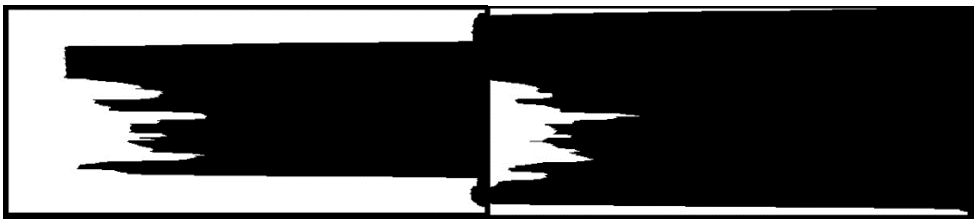
Detekované kontury byly seřazeny dle umístění na ose x. Pokud byly ohraničené kontury příliš blízko sebe (vzdálenost sousední kontury byla menší, než polovina vzdálenosti sousední kontury z opačné strany), byly považovány za kontury nacházející se v oblasti jednoho znaku.

*Obrázek 49 Vymezení a vytřídění detekovaných kontur*

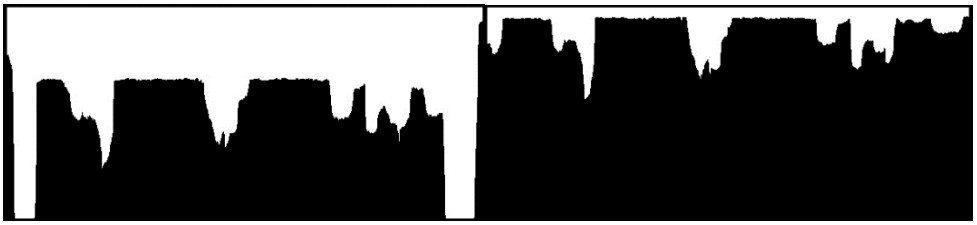
Číslice byly segmentovány na základě vertikálních pozic jednotlivých sloučených kontur a linií, jež ohraničují hledaná čísla v horizontálním směru. Popsané řešení je implementováno ve funkci *FindNumbers*.

*Obrázek 50 Výseky segmentů číslic*

Desetinné oblast číselníku byla vyhodnocena alternativní funkcí *FindNumbersIntegral,* jenž v části předzpracování obrazu navíc eliminuje červenou barvu jednoduchým rozložením každého pixelu obrazu na složky R, G a B, přičemž byla hodnota složky R nastavená na nulu.



*Obrázek 51 Srovnání vertikální projekce bez odstranění červené barvy a s odstraněním*



*Obrázek 52 Porovnání vertikální projekce*

Robustnější metodou segmentace číslic bylo možné dosáhnout úspěšnost 89%.

## 4.6 Rozpoznání znaků

### 4.6.1 Tesseract OCR

Tesseract je open-source OCR (Opticalcharacterrecognition) engine. Pracuje s binárním obrázkem na vstupu, který převádí na čistý text. Převod probíhá v několika fázích. První

z nich je connectedcomponentanalysis , 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 adaptiveclassifieru, který díky tomu může lépe rozpoznávat další slova na stránce. Pro vylepšení výsledků je průchod opakován ještě jednou.

Hlavními částmi procesu rozpoznávání textu jsou blobfiltering 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 baselinefitting, což umožňuje detekci zaoblených řádku, které vznikají při skenování stran. Pokud je to možné, snaží se identifikovat fixedpitch a porportionalwords (non-fixedpitch). Když najde fixedpitch, 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 takovém 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 characterclassifieru 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. [23]

Rozpoznávání číslic pomocí frameworku Tesseract OCR je implementováno v rámci v rámci volby OcrA. Úspěšnost klasifikace číslic byla však poměrně nízká- 60%. Z toho důvodu byla uvažována následující metoda rozpoznávání číslic.

### 4.6.2 Neuronové sítě [24]

Neuronové sítě jsou inspirovány biologickými neuronovými sítěmi. Tato vlastnost určitým způsobem předurčuje, že uměle vytvořené neuronové sítě by měly být schopny, z hlediska základních principů, se chovat stejně nebo alespoň podobně jako jejich biologické vzory. Je zřejmé, že vytvoření umělého lidského mozku se všemi jeho schopnostmi je věc jen velmi těžce řešitelná ať už z hlediska kvantity jeho neuronů či jejich způsobu propojení, chování jednotlivých typů neuronů apod. Nicméně skýtá se tu šance simulovat alespoň některé funkce lidského myšlení a tyto pak implementovat.

Neuronové sítě využívají distribuované, paralelní zpracování informace při provádění výpočtů. Jinými slovy ukládání, zpracování a předávání informace probíhá prostřednictvím celé neuronové sítě spíše než pomocí určitých paměťových míst. Tedy paměť a zpracování informace v neuronové síti je ve své přirozené podstatě spíše globální než lokální.

Znalosti jsou ukládány především prostřednictvím síly vazeb mezi jednotlivými neurony. Vazby mezi neurony vedouc ke "správné odpovědi" jsou posilovány a naopak, vazby vedoucí k "špatné odpovědi" jsouoslaboványpomocíopakovanéexpozicepříkladůpopisujícíchproblémovýprostor.

Učení je základní a podstatná vlastnost neuronových sítí. Tento fakt zjevně vyjadřuje základní rozdíl mezi dosud běžným použitím počítačů a použitím prostředků na bázi neuronových sítí. Jestliže jsme doposud veškeré své úsilí při tvorbě uživatelských programů soustředili na vytvoření algoritmů, které transformují vstupní množinu dat na množinu dat výstupních, pak neuronové sítě již tuto náročnou fázi nepotřebují. Jakým způsobem se budou vstupní data transformovat na data výstupní, určuje právě fáze učení založená na již dříve uvedené expozici vzorků (příkladů) popisující řešenou problematiku – trénovací množina. Odpadá tedy nutnost algoritmizace úlohy, která je nahrazena předložením trénovací množiny neuronové síti a jejím učením.

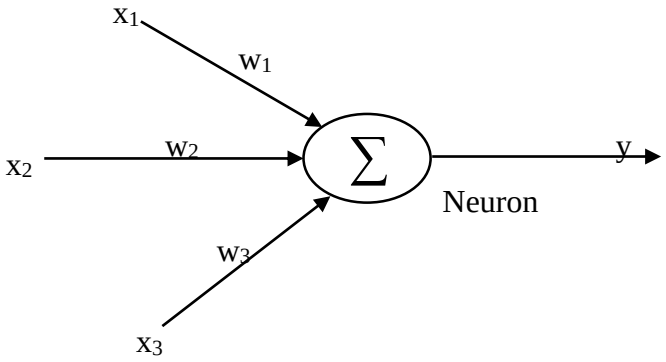
**Neuron**

Uměle vytvořený neuron je dán svým biologickým vzorem a tvoří jakousi základní "výpočetní jednotku" složitějšího komplexu – neuronové sítě.

$$y = f\left(\sum_{i=0}^{n} w_i\, x_i\right)$$

*Vztah 3 Vyjádření neuronu*

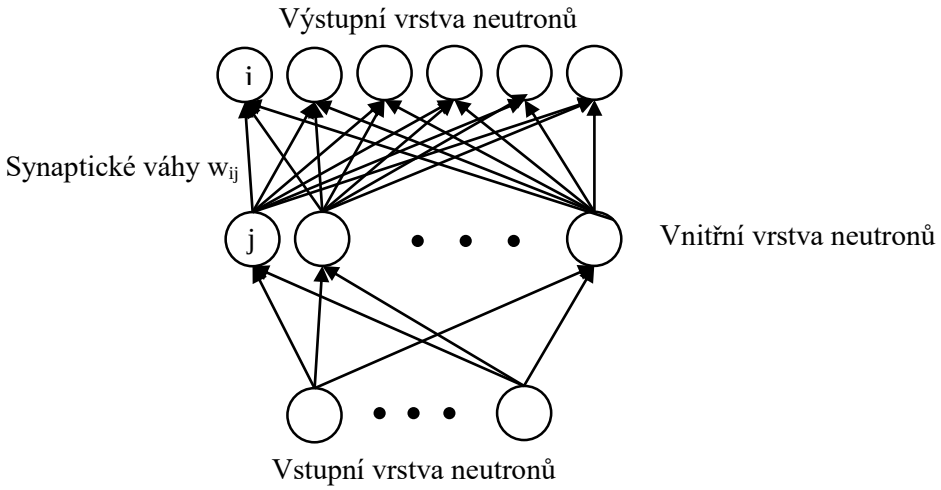
Obsahuje $n$ vstupů, které jsou přes příslušné váhy $w$ přivedeny na vstup neuronu, stejně tak je přiveden takzvaný práh neuronu (bias) přes $w0$. Vnitřní stav neuronu $z$ je dán sumou vstupů. Výstupní hodnota neuronu je dána přenosovou funkcí $f$, standardně se používá sigmoida. Takto popsaný neuron se označuje jako perceptron.




*Obrázek 53 Schematické znázornění neuronu*

**Vícevrstvé sítě**

Pravděpodobně nejrozšířenější způsob propojení spojitých perceptronů jsou tzv. vícevrstvé sítě, jejichž topologie je následující:




*Obrázek 54 Schéma vícevrstvé sítě*

Z uvedeného obrázku vyplývá, že neuronová síť je tvořena minimálně třemi vrstvami neuronů: vstupní, výstupní a alespoň jednou vnitřní vrstvou. Vždy mezi dvěma sousedními vrstvami se pak nachází tzv. úplné propojení neuronů, tedy každý neuron nižší vrstvy je spojen se všemi neurony vrstvy vyšší. Jakým způsobem je informace zpracována v takové neuronové síti? Výklad započneme tím jednodušším, tedy dopředným šířením (feedforward) signálu:

1. Nejprve jsou excitovány na odpovídající úroveň (v rozmezí 0 až 1) neurony vstupní vrstvy.
2. Tyto excitace jsou pomocí vazeb přivedeny k následující vrstvě a upraveny (zesíleny či zeslabeny) pomocí synaptických vah.
3. Každý neuron této vyšší vrstvy provede sumaci upravených signálů od neuronů nižší vrstvy a je excitován na úroveň danou svou aktivační funkcí (vztah 1).
4. Tento proces probíhá přes všechny vnitřní vrstvy až k vrstvě výstupní, kde pak získáme excitační stavy všech jejích neuronů.

V podstatě jsme tímto způsobem získali odezvu neuronové sítě na vstupní podnět daný excitací neuronů vstupní vrstvy. Takovým způsobem vlastně probíhá šíření signálů i v biologickém systému, kde vstupní vrstva může být tvořena např. zrakovými buňkami a ve výstupní vrstvě mozku jsou pak identifikovány jednotlivé objekty sledování. Otázkou zůstává to nejdůležitější, jakým způsobem jsou stanoveny ony synaptické váhy vedoucí

ke korektní odezvě na vstupní signál. Proces stanovení synaptických vah je opět spjat s pojmem učení - adaptace – neuronové sítě.

Neuronová síť byla implementována pomocí frameworku Neuroph. Množství neuronů ve vstupní vrstvě odpovídá profilovému vektoru číslic o velikosti 600, neboť byla vstupní data normalizována na rozměr 20x30 obrazových bodů. Počet neuronů ve vnitřní vrstvě byl stanoven na hodnotu 40. Počet neuronů ve výstupní vrstvě odpovídal počtu klasifikovaných tříd deseti číslic. Neuronová síť byla trénována počtem generovaných a náhodně zašuměných číslic odpovídajících písmu použitém na plynoměrech o počtu 700, doplněných o dalších 300 číslic odvozených z číslice reálných číselníků. S uvedenou konfigurací sítě byla dosažena přesnost klasifikace 83%.

# 5 ZÁVĚR

Účelem této práce bylo nasbírat velké množství znalostí v oboru strojového vidění a OS Android a uplatnit ho v podobě aplikace pro mobilní telefony, která odečítá spotřebovaný objem plynu a identifikátor plynoměru.

Výsledek jsem splnil v plném rozsahu zadání. Přiložená aplikace umí rozpoznávat informace z indikačního zařízení plynoměru s úspěšností téměř 85%. Návazností na tuto práci by mohla být zlepšení efektivity čtení ze snímku nebo vyhodnocení obrazu v reálném čase už během zaměřování plynoměru.

Přiložený projekt aplikace obsahuje příkladný program s několika přídavnými větvemi řešení, kde každá větev vyhodnocuje výsledek jinými algoritmy.

# BIBLIOGRAFIE

1. Jak se vyvíjel operační systém Android? Napříč jeho historií až do současnosti. *Android Market.* [Online] [Citace: 5. Duben 2015.] http://androidmarket.cz/android/jak-se-vyvijel-operacni-system-android-napric-jeho-historii-az-do-soucasnosti-1-dil/.

2. The Story Behind The Android Logo. *trendblog.net/.* [Online] [Citace: 15. Duben 2015.] http://trendblog.net/the-story-of-the-android-logo/.

3. Android System Architecture. *Kebomix Blog.* [Online] [Citace: 15. Duben 2015.] https://kebomix.wordpress.com/2010/08/17/android-system-architecture/.

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

5. 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.

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

7. 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.

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

9. Learn About Java Technology. *www.java.com.* [Online] Oracle. [Citace: 2016. Únor 20.] https://www.java.com/en/about/.

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

11. BAR CODE BASICS. *Pointil Systems.* [Online] [Citace: 8. Květen 2016.] http://pointil.gdom.net/wp-content/uploads/2012/08/BAR-CODE-BASICS.pdf.

12. Čárové kódy (teorie). *Automatická identifikace GABEN.* [Online] [Citace: 8. Květen 2016.] http://www.gaben.cz/cz/faq/carove-kody-teorie.

13. CODE 128 SYMBOLOGY. *BARCODE ISLAND.* [Online] [Citace: 8. Květen 2016.] http://www.barcodeisland.com/code128.phtml.

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

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

16. Richard O. Duda, Peter E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. [Online] 4 1971. [Citace: 15. Duben 2015.] http://www.ai.sri.com/pubs/files/tn036-duda71.pdf.

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

18. Morphology. *HIPR Image Processing Learning Resources.* [Online] [Citace: 01. Květen 2016.] http://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm.

19. D'Silva, Pasquale. A Little About Color: HSV vs. RGB. *Kirupa.com.* [Online] [Citace: 10. Duben 2016.] https://www.kirupa.com/design/little_about_color_hsv_rgb.htm.

20. *https://dcmmachinevision.files.wordpress.com.* [Online] [Citace: 10. Květen 2016.] https://dcmmachinevision.files.wordpress.com/2011/02/hull_p.png.

21. Structural Analysis and Shape Descriptors. *OpenCV 3.1.0 Open Source Computer Vision.* [Online] [Citace: 10. Květen 2016.] http://docs.opencv.org/3.1.0/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a&gsc.tab=0.

22. Detecting Barcodes in Images with Python and OpenCV. *PyImageSearch.* [Online] [Citace: 5. Květen 2016.] http://www.pyimagesearch.com/2014/11/24/detecting-barcodes-images-python-opencv/.

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

24. prof . Ing. Ivo Vondrák, CSc. Neuronové sítě. [Online] [Citace: 5. Květen 2016.] http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf.

25. 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 zkratek

| | |
|---|---|
| ADT | Android DevelopmentTools |
| API | Application Programming Interface |
| DP | Diplomová práce |
| HT | Houghova transformace |
| HSV | Hue, Saturation, Value |
| HSL | Hue, Saturation, Lightness, |
| HTML | HyperText Markup Language |
| IDE | Integrated Development-Environment |
| Java ME | Java Micro Edition |
| Java SE | Java Standard Edition |
| JDK | Java DevelopmentKit |
| JVM | Java Virtual Machine |
| OCR | Opticalcharacterrecognition |
| OpenCV | Open Source Computer Vision |
| RGB | Red Green Blue |
| SDK | Standard DevelopmentKit |
| UI | User Interface |
| VM | Virtual Machine |
| XML | Extensible Markup Language |

# Seznam příloh

Příloha 1. Manuál ke zprovoznění aplikace.

Příloha 2. Zdrojový kód.

Příloha 3. CD/DVD