Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Automatické rozpoznávání stavu elektroměru z fotografie

Diplomová práce

Bc. Ondřej Hanzlík

Vedoucí práce: Ing. Miroslav Skrbek, Ph.D.

České Budějovice 2015

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL MAGISTERSKÉ PRÁCE

Student: Bc. Ondřej Hanzlík .....
(jméno, příjmení, tituly)

Obor – zaměření studia: Aplikovaná informatika .....

Katedra: Ústav aplikované informatiky

Školitel: Ing. Miroslav Skrbek, Ph.D. .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma magisterské práce: Automatické rozpoznávání stavu elektroměru z fotografie

Cíle práce :

Seznamte se s existujícími dílčími řešeními pro rozpoznávání stavu elektroměru z fotografie. Proveďte rešerši relevantních algoritmů pro zpracování obrazu a dostupných knihoven podporujících tyto algoritmy jak na počítačích PC, tak mobilních zařízeních. Navrhněte, implementujte systém pro automatické rozpoznávání stavu elektroměru, který převede spolehlivě obraz z fotoaparátu mobilního zařízení do textové podoby. Uvažujte možnost rozpoznání přímo na mobilním zařízení v místě pořízení fotografie i zpracování obrazové informace na PC odděleně od pořízení fotografie. Vhodnými testy vyhodnoťte chybovost a použitelnost realizovaného systému. Rozsah implementace upřesněte po dohodě s vedoucím práce.

Základní doporučená literatura : dodá vedoucí práce

Financování práce : .....

Vedoucí práce : Ing. Miroslav Skrbek, Ph.D. ..... podpis : .....

U externích vedoucích fakultní garant práce..... podpis : .....

Garant oboru mag. studia ..... podpis : .....

Vedoucí katedry ..... podpis .....

Případný souhlas vedoucího ústavu AV ..... podpis : .....

V Českých Budějovicích dne 24. 1. 2013

Převzal/a dne 31. 1. 2013 podpis : .....

Bibliografické údaje

Hanzlík Ondřej, 2015: Automatické rozpoznávání stavu elektroměru z fotografie.

[Automatic recognition of the electrometer status from picture, Mgr. Thesis, in Czech.] – 68 p., Faculty of Science, Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace:

Práce se zabývá problematikou rozpoznávání stavu elektroměru ze snímaného obrazu. Konkrétně jde o snímání elektroměru fotoaparátem mobilního telefonu. Na snímaném obraze je detekována plocha s číselníkem elektroměru a na té jsou následně detekována jednotlivá čísla. Ta jsou rozpoznávána za pomoci neuronové sítě. Pro získání informací z obrazu elektroměru, díky kterým dokážeme jeho stav rozpoznat, je využito technik segmentace obrazu. Pro klasifikaci výstupů segmentace je využito klasifikačních nástrojů, konkrétně vektorového stroje (SVM) a neuronové sítě. Pro řešení problematiky segmentace obrazu je použita knihovna OpenCV, stejně tak jako k implementaci vektorového stroje. Číslice na číselníku elektroměru jsou klasifikovány pomocí neuronové sítě, která byla vlastnoručně implementována. Celá aplikace pro rozpoznání je na platformě Android. Součástí práce je i vytvoření desktopové aplikace, která slouží pro testování neuronové sítě a vytváření jejích modelů. Práce současně popisuje, jak ukládat potřebná data získávaná v průběhu rozpoznávání, která jsou využívaná pro práci s neuronovou sítí. Součástí práce je spuštěný web, který bude rozvíjen pro možnost zapojení se do dalšího vývoje systému. Na webu je dostupný veřejný repozitář se zdrojovými kódy vytvořenými při implementaci.

Annotation:

This thesis deals with problems of recognition of an electrometer´s state from sensing image. It is tangibly about electrometer´s scanning by a mobile phone´s camera. There is a surface with an electrometer´s dial which is detected and on this surface the particular numbers are detected consequently. The numbers are recognized via neural network. For more information from this image there are used some techniques of image segmentation to check the status. For the classification of the segmentation´s outputs are used classification tools, especially a support vector machine (SVM) and neural networks. Problems of image segmentations are solved by using OpenCV library. OpenCV is used for the implementation of the vector machine either. Application is on Android platform. Part of the thesis is concerned in a creation of a desktop application which is instrumental towards testing of neural network. The thesis also describes how to save the necessary data gathering in the course of the recognition which are used for working with neural network. The part of the thesis also deals with running web which will be evolved for the opportunity to participate in the further

development of the system. There is available a public repository with source codes created during implementation.

Keywords:

Recognition, segmentation, OpenCV, neural network, SVM, vector machine, preprocessing, Android, Java

Klíčová slova:

Rozpoznávání, segmentace, OpenCV, neuronová síť, SVM, vektorový stroj, předzpracování, Android, Java

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.
Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové, a to v nezkrácené podobě, elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne 24. 4. 2015

Podpis: .....

Bc. Ondřej Hanzlík

Poděkování

Děkuji Ing. Miroslavu Skrbkovi, Ph.D. za pomoc při zpracování a řešení diplomové práce.

Obsah

Úvod

Tématem této práce je problematika rozpoznání stavu elektroměru za pomoci snímacího zařízení. Předpokládané snímací zařízení je fotoaparát mobilního telefonu. Aby bylo možné rozpoznání stavu elektroměru provést, je třeba zanalyzovat skutečnosti týkající se procesu odečtu elektroměru. Je tedy třeba se seznámit se samotnými variantami typů elektroměrů. Existují dva základní typy elektroměrů, které nás budou zajímat, co se odečtu týče. Prvním typem jsou analogové elektroměry a druhým jsou digitální elektroměry. Tyto elektroměry pak mají různý vzhled, což je třeba také zahrnout do analýzy, jelikož tento fakt hraje v návrhu systému zásadní roli. V práci jsou popsány jednotlivé druhy elektroměrů v závislosti na vlastnostech místa, informujícím o stavu elektroměru - číselníku.

Fakt, že pro snímání elektroměru se počítá s využitím fotoaparátu mobilního zařízení, je třeba brát v potaz při samotném rozhodování o povaze systému, který bude problematiku práce řešit. V práci je zhodnocen charakter samotného systému v závislosti na možnostech, kterými lze rozpoznávání stavu elektroměru realizovat právě s využitím fotoaparátu mobilního telefonu.

Pro specifikaci povahy systému uvážíme dvě hlavní možnosti rozpoznání stavu elektroměru z fotografie. První možností je rozpoznání přímo na mobilním zařízení v místě pořízení fotografie, a druhou možností je zpracování obrazové informace na PC, odděleně od místa pořízení fotografie. Oba tyto směry mají ve svém charakteru společnou většinu technik, které je třeba aplikovat pro možnost realizace rozpoznání od počátku procesu až do finálního konce. Pro vytvoření kompletní implementace je třeba znát i postup, jak zjišťování stavu elektroměru v praxi probíhá a přizpůsobit tomuto vývoj aplikace. Před vlastním návrhem systému jsou tedy zanalyzovány i potřeby těch, kdo odečty elektroměrů provádí a další nutné aspekty, které do návrhu systému vstupují a ovlivňují jeho charakter a funkčnost.

Dále následuje analýza a popis jednotlivých kroků a techniky, které lze pro realizaci celistvého systému brát v potaz. Jsou zde popsány relevantní algoritmy pro zpracování obrazu a dostupné knihovny podporující tyto algoritmy jak na počítačích PC, tak na mobilních zařízeních. Současně jsou popsány možnosti provedení rozpoznání číslic

na snímku s číselníkem elektroměru a možnosti jejich převodu z obrazové do textové podoby.

Součástí práce je návrh a implementace systému pro automatické rozpoznání stavu elektroměru, který převede spolehlivě obraz z fotoaparátu mobilního zařízení do textové podoby**.** Tento systém je vhodnými testy otestován a je vyhodnocena jeho chybovost, stejně jako zhodnocena jeho použitelnost pro potřeby automatizovaného zjišťování stavu elektroměru za pomoci fotoaparátu mobilního zařízení.

1.1 Cíle práce

1. Analýza kritérií a aspektů týkajících se návrhu použitelného řešení problematiky. V závislosti na tom provedení vyhodnocení charakteru a povahy systému.

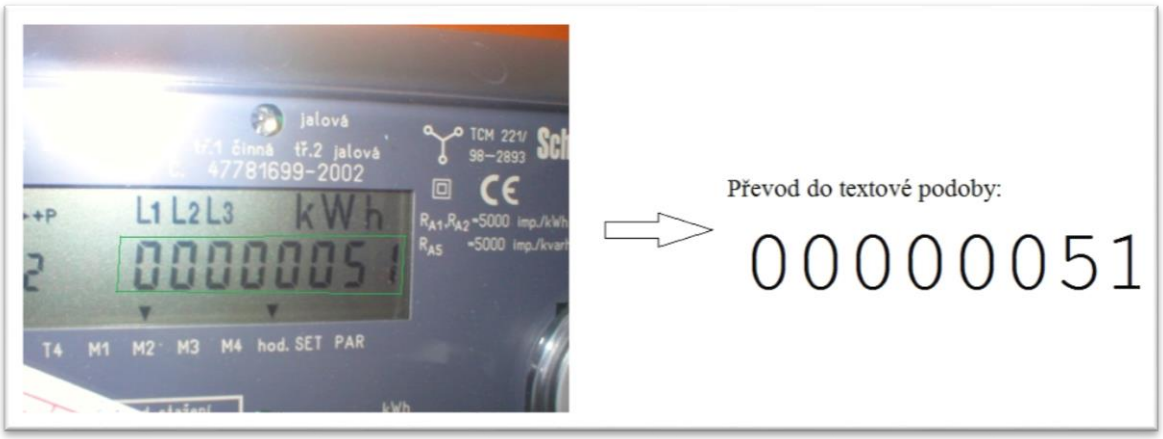
2. Seznámení se s existujícími dílčími řešeními pro rozpoznávání stavu elektroměru z fotografie

3. Provedení rešerše relevantních algoritmů pro zpracování obrazu a dostupných knihoven podporujících tyto algoritmy jak na počítačích PC, tak na mobilních zařízeních

4. Návrh a implementace systému pro automatické rozpoznávání stavu elektroměru, který převede spolehlivě obraz údaje o stavu elektroměru z fotoaparátu mobilního zařízení do textové podoby

5. Vyhodnocení chybovosti a použitelnosti realizovaného systému vhodnými testy

Pro ucelení představy o požadavku na základní funkčnost systému je na obrázku 1 demonstrován hlavní cíl implementované aplikace a to je převod obrazu elektroměru do textového údaje s hodnotou informující o stavu elektroměru.



Obrázek 1 – převod obrázku elektroměru do textové podoby stavu elektroměru

2 Teorie

Aby bylo docíleno finálního požadovaného stavu systému, jehož výstupem je textová hodnota stavu elektroměru, je třeba zpracovat veškeré aspekty zmíněné v úvodu práce. V práci je tedy popsána analytická část, vedoucí k popsání návrhu charakteru systému a jeho funkcionalit. Následuje popis řešení a způsobu samotné implementace. V teoretické části budou popsány základní techniky, ze kterých se vychází při vytváření systému a popsáno jejich konkrétní využití, které bylo často zjištěno až při implementaci. Začněme popisem existujících dílčích řešení využitelných pro navrhovaný systém.

2.1 Existující dílčí řešení

Dosud se nepodařilo nalézt žádnou práci, zabývající se konkrétně rozpoznáváním stavu elektroměru z obrazu. Tématika zasahující do oblasti rozpoznávání stavu elektroměrů, která již byla v některých pracích popsána, je například realizována v práci Rozpoznávání naměřených hodnot [1]. Práce se zabývá pouze detekcí číselníků

elektroměrů z fotografie. Další práce využitelné nebo týkající se našeho navrhovaného systému je Rozpoznávání SPZ z jednoho snímku [21]. Zde jsou diskutovány možnosti rozpoznání čísel z SPZ, což je částečně podobný problém jako rozpoznávání čísel z číselníku elektroměru. Problematika rozpoznávání stavu elektroměru je také do značné míry podobná problematice OCR všeobecně. Lze se tedy inspirovat z literatury zabývající se i tímto tématem. Jednou z prací, kde tuto inspiraci můžeme hledat je Zpracování a rozpoznávání obrazu [20].

Můžeme však říci, že návrh a realizace systému pro automatické rozpoznávání stavu elektroměru fotoaparátem mobilního telefonu bude nutné vytvořit od začátku, bez návaznosti na jiné řešení.

2.2 Specifikace

2.2.1 Typy elektroměrů

Jak již bylo řečeno, existují dva hlavní typy elektroměrů. Jsou to elektroměry digitální a analogové. Na obrázku 2 můžeme vidět ukázku digitálního elektroměru. Na obrázku 3 je zobrazen analogový elektroměr.

Obrázek 2 – digitální elektroměr

Obrázek 3 – analogový elektroměr

Elektroměry se sice dělí pouze na tyto dva základní typy, nicméně pro potřeby řešení problematiky rozpoznávání stavu elektroměru nás bude zajímat i další dělení, a to podle vzhledu číselníku elektroměru. Z hlediska počítačového vidění a vůbec možnosti

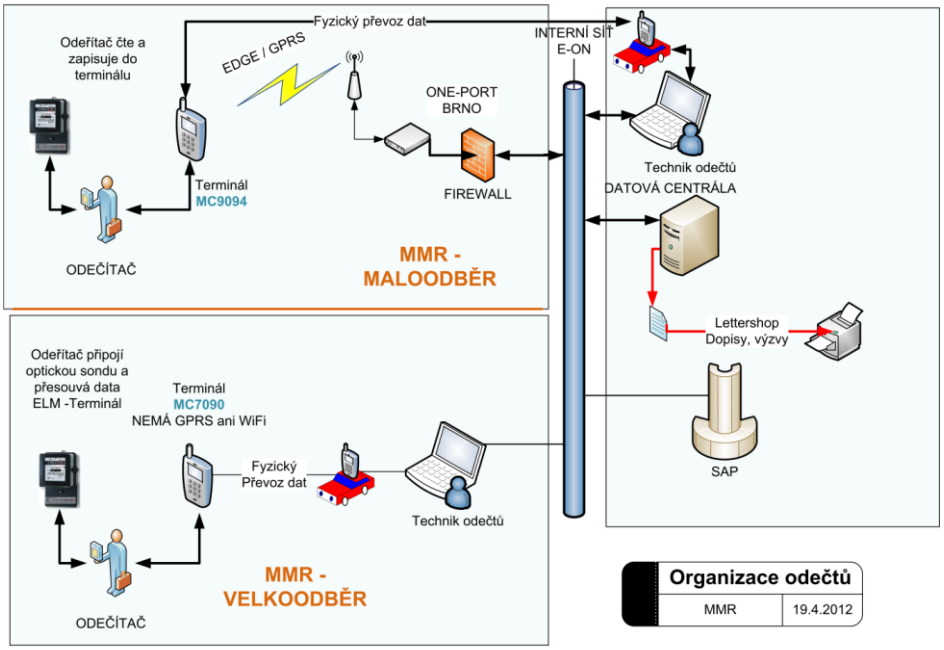
detekce určité části obrazu pomocí počítače nás totiž nezajímá ani tak reálné dělení dle typu, ale právě dělení podle obrazových charakteristik plochy, na které chceme provádět zjištění hodnot. Z dostupných fotografií [příloha 1] je vidět, že různorodost plochy, která nás na obraze zajímá je nejvýraznější u analogových elektroměrů. Údaje o stavu jsou uvnitř černé plochy (často tvaru obdélníku), kde poslední číslice je vyznačená červenou barvou. Konkrétních typů vzhledů této plochy a celého elektroměru je u analogových elektroměrů více, převážně pro ně však platí předchozí popsané. Dále je třeba brát v potaz fakt, že existují jednosazbové a více sazbové elektroměry, což znamená, že musí být umožněno sejmout stav z obou těchto typů zařízení. Na obrázku 4 můžeme vidět ukázku rozdílu vzhledu analogových elektroměrů.

Obrázek 4 – rozdílný vzhled analogových elektroměrů

Vzhled digitálních elektroměrů je méně rozmanitý, než vzhled analogových elektroměrů. Samotná část elektroměru s údajem o stavu, ale u obou typů vizuálně stejná s drobnými rozdíly. Jde o šedozeleně zabarvenou plochu s černými, digitálními číslicemi (sedmisegmentové) a dalšími informativními údaji. Důležitý je však rozdíl ve velikosti čísel u jednotlivých typů elektroměrů co se poměru výšky a šířky týče. Důvod zvýšené důležitosti tohoto rozdílu je zmíněn v kapitole 4.5 Segmentace.

2.2.2 Provádění odečtu v praxi

Proces provádění odečtu začíná zadáním seznamu elektroměrů, které je třeba obejít a provést odečet. Toto zadání je odesláno na terminál, jímž je pracovník provádějící odečet vybaven. Poté pracovník obchází elektroměry podle seznamu z terminálu a zadává do terminálu hodnoty, které čte na elektroměru. Hodnoty přiřazené elektroměrům se pak synchronizují se serverem, který má odečty na starost. Jak vypadá provádění odečtu, je zobrazeno na obrázku 5.



Obrázek 5 – proces provádění odečtu

Vzhledem k tomu, za jakých podmínek provádění odečtu probíhá, není možné uvažovat využití připojení k internetu v místě odečtu, a to hlavně z toho důvodu, že v různých oblastech, kde je třeba odečet elektroměru provést, není mobilní signál, a tím pádem ani možnost mobilního připojení k internetu. Možnost rozpoznávání na vzdáleném serveru může být tedy vyloučena.

Pro naši aplikaci z tohoto vyplývá, že musí umožňovat zadání čísla elektroměru, na kterém je právě prováděn odečet, a po rozpoznání údaje o stavu k číslu elektroměru přiřadit rozpoznanou hodnotu schválenou pracovníkem.

2.2.3 Charakteristika procesu rozpoznání

Pro rozpoznání číslic, které tvoří údaj o stavu elektroměru, je nejprve třeba vytvořit vhodný snímek elektroměru. Rozpoznání pak může proběhnout přímo v mobilním zařízení, později v počítači po nahrání sejmutých fotografií, nebo online, za využití serveru, určenému k rozpoznávání. Možnost online rozpoznávání bylo diskutováno v kapitole 2.2.2 Provádění odečtu v praxi, týkající se popisu provádění odečtu.

Možností, jak přistoupit k snímání elektroměru, je více. Je možné sejmout fotografii elektroměru, a na jejím základě rozpoznávat, nebo můžeme snímat elektroměr kamerou fotoaparátu, a tím získat sadu snímků využívaných pro rozpoznání. Přístup kdy elektroměr vyfotografujeme, je však limitující kvůli zvýšené pravděpodobnosti jednorázového sejmutí fotografie, která je nevhodná pro dosažení úspěšného procesu rozpoznávání. Tento fakt byl v bakalářské práci Rozpoznávání naměřených hodnot [1] ověřen na sadě fotografií pořízených při reálném provádění odečtu. Při pokusu vytvořit jednotný algoritmus, za pomoci něhož by bylo možno z pořízených fotografií rozpoznat stav elektroměru, bylo zjištěno, že tento přístup není dostatečně úspěšný, a to i z důvodu, že údaj o stavu není z některých fotografií zjistitelný ani lidským okem.

V případě rozpoznávání z pořízené fotografie bychom museli opakovat fotografování tak dlouho, dokud by se nepodařilo sejmout fotografii, z níž by rozpoznání proběhlo úspěšně, což není uživatelsky přívětivý přístup. Abychom se tomuto vyhnuli, je vhodné využít možnost rozpoznávání ne z jednorázově sejmuté fotografie, ale přímo z obrazu snímaného kamerou. Tím odstraníme nutnost fotografování a umožníme lepší zpětnou vazbu od uživatele (pracovníka provádějícího odečet). Z toho přístupu vyplývá, že můžeme vyřadit zmiňovanou možnost rozpoznání stavu z fotografie, po pozdějším nahrání do počítače.

Nejlepším řešením se tedy jeví provádět rozpoznávání přímo na mobilním zařízení z obrazu snímaného v reálném čase. Tento přístup má jednu, již zmiňovanou, nespornou výhodu, která je takřka nutností pro reálné využití systému pro provádění odečtu. Touto výhodou je zpětná vazba od uživatele. Při provádění odečtu dojde k situacím, kdy není

možné vytvořit dostatečně kvalitní snímek elektroměru. Některé číslice nejsou jednoznačně identifikovatelné (např. při přechodu jedné číslice na další u analogových elektroměrů, kdy je kolečko s čísly natočeno tak, že napůl ukazuje jednu číslici a napůl druhou), nebo nejsou dostatečně vhodné viditelnostní podmínky. Další situace, která může nastat, je špatné rozpoznání číslic. I přes to, že dosáhneme kvalitního snímku elektroměru, se může stát, že některé číslice jsou špatně rozpoznány. Reálně není možné dosáhnout stoprocentně bezchybného systému a tak možnost zakročit do procesu rozpoznání je velice důležitá. Uživatel musí mít možnost sám ručně potvrdit správnost rozpoznání, nebo opravit chybně klasifikované číslice. Nejlépe lze tuto možnost realizovat za využití přístupu, kdy rozpoznávání provádíme přímo na místě, kdy je uživatel u elektroměru a může nejsnáze a nejlépe ověřit správnost rozpoznání a provést případnou opravu chyb.

2.2.4 Charakter a funkcionality systému

Aplikace, která bude implementována, musí splňovat podmínky diskutované v předchozích kapitolách. Aplikace musí tedy splňovat následující funkcionality

- Zadat, zda elektroměr, který aktuálně rozpoznáváme, je digitální nebo analogový.
- K naměřenému údaji mít možnost zvolit, zda jde o hodnotu vysokého, nebo nízkého tarifu.
- V případě, že provádíme odečet elektroměru, který má jiný vzhled, než předpokládaný, musí být umožněno zadat stav elektroměru ručně
- Zadat číslo elektroměru, na kterém je prováděn odečet
- Využití fotoaparátu ke snímání v reálném čase
- Spuštění rozpoznávání ve chvíli, kdy máme snímačem zaměřenou plochu elektroměru s údaji o stavu
- Zobrazení výsledků rozpoznání a možná editace jednotlivých číslic, které jsou výsledkem rozpoznání
- Uložení výsledku rozpoznání k danému číslu elektroměru

Jelikož proces rozpoznávání je klíčovou částí systému a skládá se z mnoha kroků, je třeba specifikovat jeho jednotlivé části. Možných způsobů realizace rozpoznávání je spousta. Následující část práce tedy vymezuje směr a popis algoritmů a postupů, které povedou k návrhu rozpoznávací čísti systému.

2.2.5 Průběh proces rozpoznávání

Rozpoznávání probíhá pomocí dvou hlavních částí. První je detekce jednotlivých číslic tvořících informaci o stavu elektroměru, druhou částí je pak rozpoznání (predikce) těchto detekovaných číslic. Abychom mohli z jednotlivých separovaných obrázků čísel, tvořících údaj o hodnotě stavu elektroměru, rozpoznat jejich textovou reprezentaci, je třeba získat jejich charakteristiky, pomocí kterých pak proběhne predikce toho, jaká číslice je na každém obrázku vykreslena. Podobnou predikcí číslic i znaků se zabývá bakalářská práce Systém pro rozpoznávání znaků [2]. Jednotlivé kroky, kterými je proces rozpoznávání realizován, jsou tedy následující:

1. Vhodně zachycený snímek plochy elektroměru s údajem o stavu fotoaparátem mobilního zařízení (vhodný vstup)
2. Předzpracování obrazu pro následnou detekci plochy s údajem o stavu
3. Detekce a vyříznutí plochy s údajem o stavu elektroměru, čímž dojde k odstranění okolního obrazu, který nepotřebujeme
4. Detekce jednotlivých znaků z upraveného snímku – rozdělení textu po číslicích a vyříznutí jednotlivých separovaných číslic
5. Získání charakteristik číslic pro následnou možnost predikce znaků na obrázku
6. Rozpoznání znaků (číslic) na základě jejich charakteristik a uspořádáním výsledků ve správném pořadí získání textové podoby stavu elektroměru (výstup procesu)

Techniky, kterými lze řešit problematiku, týkající se výše shrnutých bodů, spadají do odvětví počítačového vidění, segmentace, klasifikace a rozpoznávání. Zbývající část této kapitoly se tedy zabývá jejich popisem a s nimi spojenými relevantními algoritmy a postupy.

Zjištění polohy číselníku, případně konkrétních číslic na číselníku je typický segmentační problém. Potřebujeme z obrazu vyselektovat pouze plochu, která nás zajímá. Stejně tak pak předzpracování a separace jednotlivých číslic číselníku. Kapitola 2.3 Segmentace a vybrané segmentační algoritmy se zabývá právě těmito segmentačními technikami.

Při segmentaci obrazu může být dosaženo toho, že z obrazu budou získány údaje (části obrazu), které nejsou ty, jež jsme si představovali. Konkrétně v našem systému můžeme za číselník elektroměru označit takovou část obrázku, která svými vlastnostmi odpovídá podmínkám segmentace, ale ve skutečnosti to není námi požadovaný údaj, který jsme chtěli získat. Tento stav může být způsoben jak využitím nevhodných segmentačních technik, tak vlastnostmi zpracovávaného snímku, což souvisí i s prostředím, v jakém byl snímek pořízen (světelnostní podmínky, úhel snímání, materiál ochranného povrchu číselníku, elektroměr, atp.). Pro zlepšení úspěšnosti získávání námi požadovaných údajů z obrázku, můžeme využít klasifikačního nástroje, díky kterému budeme moci rozlišit, zda výstupy segmentace jsou námi požadované, nebo nikoli. Takovouto klasifikaci je možné provést pomocí tzv. podpůrného vektorového stroje (SVM - support vector machine), jehož popisem se zabývá kapitola 2.4 Support Vector Machine (SVM).

Rozpoznání číslic z vyseparovaných obrázků je další klasifikační problém. K jeho řešení lze využít neuronové sítě (NN - neural networks), jak již bylo dříve zmíněno. Z obrázků číslic je třeba získat atributy obrázku, se kterými bude dále neuronová síť pracovat. Neuronové sítě jsou popsány v kapitole 2.5 Neuronová síť (NN).

2.3 Segmentace obrazu a vybrané segmentační algoritmy

Segmentace obrazu je proces, při němž dochází k rozdělení obrazu na různé segmenty – části obrazu. Díky tomuto procesu máme možnost získat z obrazu ty informace, které požadujeme, změnit reprezentaci obrazu, umožnit snadnější analýzu obrazu. Můžeme tak například získat všechny body v určité části obrazu s určitou charakteristikou, jako je například barva nebo jas.

Segmentačních technik a algoritmů je spousta a jejich různou kombinací můžeme vytvářet nekončící množství segmentačních postupů. Vymezíme tedy segmentační

techniky relevantních pro náš systém. Potřebujeme nalézt polohu číselníku na snímku elektroměru, vyseparovat z něj číslice a získat atributy obrázků. Následuje tedy shrnutí knihoven podporujících segmentační algoritmy, popis všeobecných segmentačních algoritmů a poté algoritmů relevantních pro každou segmentační část systému.

2.3.1 Knihovny počítačového vidění

Knihovny implementují segmentační algoritmy a techniky jsou využitelné pro řešení naší problematiky. Pro přehled si některé z nich představíme.

- BoofCV – open source, jazyk Java, navrženo pro realtimové zpracování obrazu, vydána pod licencí Apache
- STAIR Vision Library (SVL) – kódové označení lasik, původně vyvinutá pro Stanford AI Robot Project, podpora počítačového vidění, strojového učení, pravděpodobnostních grafických modelů
- VLFeat – open source, implementuje oblíbené algoritmy počítačového vidění, specializovaná na porozumění obrazu, extrakci lokálních příznaků a hledání shody (pattern matching)
- FastCV – orientováno na využití pro mobilní zařízení, podpora mnoha platforem, implementuje techniky rozpoznání gest, rozšířené reality, detekce obličeje, sledování a rozpoznávání objektu (tracking and recognition),
- SimpleCV – open source, jazyk python, zaměřeno na umožnění využívání technik počítačového vidění bez předchozích znalostí bitové hloubky, formátů, barevných prostorů, atd.
- OpenCV – BSD licence, jazyky C/C++/Python/Java, komunita 47000 uživatelů, počet stažení přes 9000000.

K poslední zmiňované knihovně, OpenCV, je vhodné zmínit ještě pár faktů. Knihovna je napsána v jazyce C/C++ a má SDK pro platformu Android. Stejně tak poskytuje wrapper pro platformu Java a syntaxe je blízce podobná zápisu v jazyce C++. Má otevřený zdrojový kód a jednu z největších komunit v porovnání s jinými knihovnami.

Je stále aktivně vyvíjena a aktualizována. Knihovna je celosvětově uznávaná a využívaná pro řešení problematiky týkající se počítačového vidění.

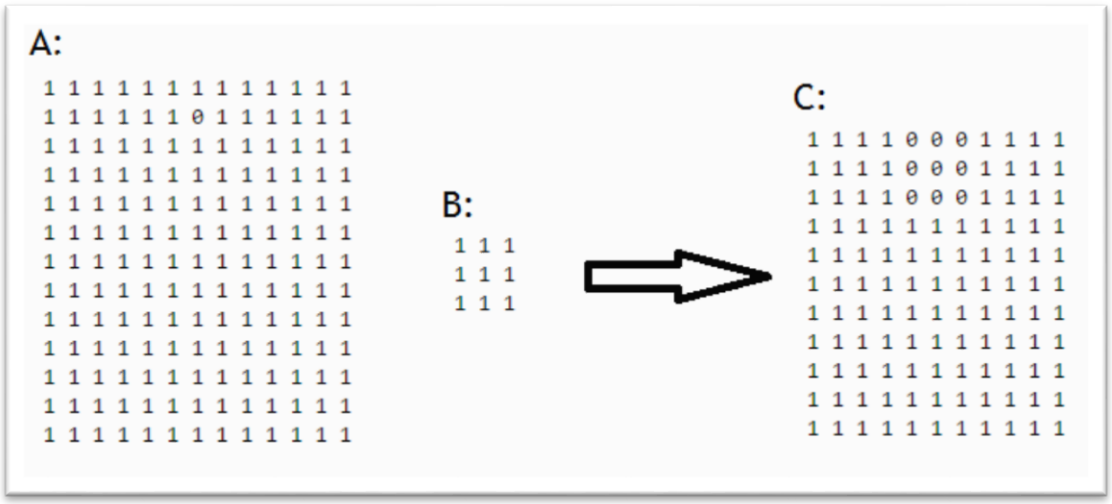
2.3.2 Relevantní algoritmy

2.3.2.1 Eroze a dilatace (Erode and Dilate)

Eroze je jednou ze dvou základních operací v morfologickém zpracování obrazu, na nichž jsou všechny ostatní morfologické operace založeny. Druhou základní operací je dilatace.

Eroze binárního obrázku A strukturním elementem B je definovaná jako $A \ominus B = \{z \in E | B_z \subseteq A\}$. Dilatace A pomocí B je pak definována jako $A \oplus B = \bigcup_{b \in B} A_b$.

Pro lepší představu funkce eroze můžeme vidět obrázek 6 kde A je matice 13x13 a B je matice 3x3. Erozí A maticí B vznikne matice C velikosti 11x11.



Obrázek 6 – ukázka operace eroze

S morfologickými operacemi eroze a dilatace úzce souvisejí operace otevření a uzavření (opening, closing). Otevření je provedeno za využití operace eroze a následné aplikování operace dilatace. Uzavření je zas definováno jako aplikování operace

dilatace následující operací eroze. Otevření A za použití strukturálního elementu B je tedy definováno, jako $A \circ B = (A \ominus B) \oplus B$, kde $\oplus$ značí operaci dilatace a $\ominus$ operaci eroze. Uzavření A za použití strukturálního elementu B je pak definováno, jako $A \bullet B = (A \oplus B) \ominus B$, kde $\oplus$ opět značí operaci dilatace $\ominus$ operaci eroze.

Uzavření společně s otevřením slouží k morfologickému odstranění šumu v obraze. Otevření odstraňuje malé objekty a uzavření odstraňuje malé díry.

2.3.2.2 Hledání kontur

Výsledkem hledání kontur v obraze je seznam prostorů, jež jsou definovány určitou křivkou a tvoří tak uzavřené oblasti, nalezené v různých částech obrazu. Kontura je seznam bodů, který představuje určité zakřivení v obraze - křivku. Jedním ze způsobů reprezentace kontury je sekvencí bodů, kde každý bod nese informaci o umístění dalšího bodu na křivce.

Díky metodě hledání kontur jsme schopni najít v obraze spojité plochy, které se liší od dalších ploch, které se vyskytují v jejich okolí.

2.3.2.3 Adaptivní prahování (Adaptive thresholding)

Prahování je způsob, jak segmentovat objekty od jejich pozadí. Jestliže je pozadí relativně jednotné, to znamená, na jedné části obrázku není zásadně barevně a jasově jiné než na jiné části obrázku, je možné využít globální prahovou hodnotu, za pomoci níž lze prahování provést. Dojde tak k binarizaci obrazu na základě stanoveného prahu a tím odlišení objektů od pozadí. V případě, že jsou však v intenzitě pozadí velké rozdíly, je vhodné využít adaptivního prahování. To využívá prahové adaptivní funkce, která počítá prahové hodnoty v oblastech bloků určité definované velikosti. Pro každý blok je pak spočítána prahová hodnota, pomocí průměru z místního okolí bodu v bloku.

2.3.2.4 Houghova transformace

Houghova transformace je metoda hledání příznaků v obraze. Často je využívána pro detekci čar nebo kruhů v obraze, může však být generalizována pro hledání libovolného tvaru. V obrazu jsou nalezeny body, které vyhovují definici hledaného tvaru pomocí Houghovy transformace, čímž můžeme nalézt pozici požadovaných objektů v obraze.

Houghova trasformace byla vyvinuta k reprezentaci čar a rovinných křivek, čehoš je dosaženo pomocí transformace z Kartézského souřadnicového systému do polárního systému. Parametrické vyjádření křivky je $\rho = x\cos\theta + y\sin\theta$. Výhodou této metody je, že není příliš citlivá na šum a porušená data [19].

2.3.2.5 Cannyho detektor hran pomocí Otsunovi prahovací hodnoty

Cannyho hranový detektor je algoritmus zahrnující několik kroků pro získání co nejlepšího výsledku při detekci hran v dvourozměrném diskrétním obraze. Doporučený postup při provádění Cannyho detekce hran je eliminace šumu (gausovským filtrem), určení gradientu, nalezení lokálních maxim a eliminace nevýznamných hran, což je možné provést pomocí prahování. Při provádění prahování je možné využít Otsunův algoritmus pro výpočet prahové hodnoty. Otsunova prahovací metoda je zaměřena na výpočet prahu takovým způsobem, aby hodnota šumy popředí a rozložení pozadí byla minimální. Výpočet hodnoty prahu v obraze probíhá za pomoci výpočtu $\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$, kde váhy ω_i jsou pravděpodobnosti dvou tříd oddělené prahem t a σ_i^2 jsou rozptyly těchto tříd. To znamená, že algoritmus předpokládá, že obraz obsahuje dvě třídy pixelů (pixely v popředí a pixely v pozadí).

2.3.2.6 SWT - Stroke Width Transform

Využití algoritmu SWT je popsáno v práci, zaměřené na detekci textu touto technikou [14]. Na základě gradientu a šířky mezi hranami nalezených objektů v obraze lze detekovat objekty, které reprezentují čísla a písmena.

2.3.3 Možné způsoby detekce číselníku

2.3.3.1 Detekce vertikálních hran

Částečně by se dal problém řešit pomocí postupu zmíněného v bakalářské práci [3], jejíž součástí je demonstrace využití segmentace pro účely detekce plochy, na níž se vyskytuje SPZ. Jde o využití detekce hran za pomoci operátoru pro detekci vertikálních hran. Určení detekování místa s SPZ proběhne na základě specifikovaného množství vertikálních hran, které když se v obraze vyskytují, můžeme oblast označit za oblast s SPZ.

2.3.3.2 Top – hat

Top – hat transformace je operace, která extrahuje drobné prvky a detaily z předloženého snímku. Existují dva typy této transformace. White top – hat transformace, která je definována jako rozdíl mezi vstupním obrazem a jeho otevřením pomocí strukturálního elementu a Black top – hat transformace, která je definována jako rozdíl mezi uzavřením vstupního obrazu a jeho originálem. Top – hat transformace se používá pro segmentaci popředí nebo pozadí obrazu.

Matematická definice white top – hat pro *f* je dána $T_w(f) = f - f \circ b$. Black top – hat pro *f* je definován jako $T_b(f) = f \bullet b - f$.

2.3.3.3 Využití Houghovy transformace

Za pomoci Houghovy transformace můžeme provést detekci číselníku tím způsobem, že detekujeme čáry v obraze, čímž budou detekovány čáry na horní a spodní hraně číselníku elektroměru a na jeho bočních hranách. Průsečíky těchto čar pak detekují číselník elektroměru.

Další možností je generalizovaná Houghova transformace (GHT), metoda kdy na základě matematické specifikace tvaru hledáme v obraze požadovaný tvar. Takto můžeme vyhledávat libovolný tvar v obraze nezávisle na barvě, jeho velikosti i rotaci. Můžeme tak nadefinovat tvar plochy s údajem o stavu elektroměru a tímto způsobem ji detekovat.

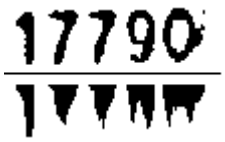
2.3.3.4 Template matching

Nalézt hledaný obrazec v obrazu lze i pomocí metody zvané template matching. Metoda probíhá tak, že zadáme template, podle kterého chceme provádět porovnání, zda je podobný obraz ve snímku, ve kterém hledáme část odpovídající templatu. Jako template můžeme zadat obrázek číselníku a poté co nalezneme ve snímku obrazec, který odpovídá templatu, který jsme zadali, označíme tuto oblast za oblast, na níž se vyskytuje číselník.

Template je vhodné upravit algoritmy pro úpravu obrazu a následně upravit i obraz, na kterém template matching provádíme.

2.3.4 Možné způsoby separace čísel

Číslice lze separovat za využití vertikálního histogramu, kdy za jeho pomoci stanovíme hranice mezi jednotlivými čísly, a následně je tak můžeme separovat. Jak vypadá použití této metody, je naznačeno na obrázku 7.



Obrázek 7 – histogram vertikální projekce

Další možností je **využití kontur** v obraze. Podobného efektu, jako má předchozí popsaná metoda, je možné dosáhnout za využití hledání kontur v obraze. V případě vhodně předzpracovaného obrazu můžeme v obraze najít kontury, vytřídit ty, které pravděpodobně můžou být číslice, odstranit malé kontury a poté separovat číslice díky pozici nalezených kontur v obraze. Metoda stojí na předpokladu, že je možné nalézt pro každé číslo na obrázku jednu konturu, která mu přísluší.

2.3.5 Získání atributů pro neuronovou síť

Pro rozpoznání číslic z obrázků číslic je nejprve třeba získat takové atributy obrázku, podle kterých může proběhnout následné rozpoznání. Zmíníme tři způsoby, kterými lze získat požadované atributy.

Prvním způsobem je **rozdělení pásma**. Této metody bylo využito v bakalářské práci Systém pro rozpoznávání tištěného písma [2]. Obrázek číslice se rozdělí na určitý počet řádků a sloupců, jejichž průniky tvoří čtverce nebo obdélníky. Díky počtu černých pixelů v černobílém obraze v každém čtverci nebo obdélníku, pomocí nichž je obraz rozdělen, získáme potřebné atributy.

Druhý způsob získání atributů je **metoda průsečíků**, kdy se v obraze hledají průsečíky pomocí projekce horizontálního a vertikálního histogramu. Z obrázku číslic můžeme získat zmíněné histogramy a využít je tím způsobem, že budou zapsány za sebou a vytvoří tak jednu sadu hodnot. Tyto hodnoty pak dělíme šířkou, respektive výškou obrázku, z nichž byly získány a dokončíme tak vytvoření sady atributů. Metoda je využívána i komerčními programy pro zpracování textu [20].

Poslední zmíněnou metodou je **strukturová analýza**. Zde je popisována hlavně geometrická a topologická struktura znaku. Hledají se geometrické prvky, jako jsou úsečky, oblouky a koncové body. Tato metoda je výhodná z hlediska možnosti invariance vůči transformaci, měřítku, či rotaci.

2.4 Support Vector Machine (SVM)

Vektorový stroj je diskriminativní klasifikátor, formálně definovaný separační nadrovinou. Jinými slovy, po předložení pojmenovaných trénovacích dat (trénovací množiny), SVM umožňuje klasifikaci nových předložených nepojmenovaných vzorků, vytvářením optimální nadroviny, za pomoci níž jsou data klasifikována. Optimální nadrovina je taková, že body leží v opačných poloprostorech a jejich vzájemná vzdálenost od roviny je co největší, což znamená, že body v každém prostoru leží co nejdále od roviny. SVM takto rozděluje data do dvou tříd, takže nadrovina je dána lineární funkcí.

Součástí SVM je technika jádrové transformace prostoru dat do prostoru vyšších dimenzí. Tím je možné převést lineárně neseparovatelnou úlohu na úlohu lineárně separovatelnou, na které je možné nalézt rozdělující nadrovinu. Díky tomu můžeme SVM využít i pro klasifikaci podle více tříd, nikoli pouze dle dvou.

Výhodou této techniky je nízká výpočetní náročnost oproti jiným klasifikačním technikám.

V případě lineárního vektorového stroje je proces učení dat $\mathcal{D}$, sadě bodů n popsán jako $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^{n}$.

2.5 Neuronová síť (NN)

Neuronová síť je výpočetní model využívaný v umělé inteligenci, strojovém učení a dataminingu. Lze za její pomoci klasifikovat a rozpoznávat různá data. Výchozím vzorem neuronové sítě je chování biologických struktur, jako jsou nervová spojení. Odtud také název neuronová síť. Neuronová síť se skládá s umělých neuronů, které jsou do značné míry modelovány podle jejich reálných biologických předobrazů. Neurony jsou v síti vzájemně propojeny a předávají si mezi sebou signály, které transformují podle určitých přenosových funkcí. Každý neuron v síti má určitý počet vstupů, který může být vždy jiný dle potřeb použití, avšak je jenom jeden výstup. Neuronové sítě nalézají uplatnění v odvětvích týkajících se rozpoznání, klasifikace, komprese obrazu nebo zvuku, předpovídání (predikce) vývoje časových řad či simulování chování nervových soustav živých organismů. Neuronových sítí je mnoho typů a implementují různé algoritmy pro dosažení cíle, ke kterému jsou určeny. Existují SOM sítě, sítě složené z RBF neuronů, sítě typu ADALINE, Kohonenovi samo organizující se sítě, Hopfieldovi sítě, rekurentní sítě, konvoluční sítě, perceptronové sítě, Neocognitron – neuronová síť, používaná za účelem rozpoznání ručně psaných znaků inspirované vnímáním obrazu částí kočičího mozku a mnoho dalších.

V současné době se dostává zvýšené popularitě tzv. konvolučním neuronovým sítím v kombinaci s metodami deep learningu. Za pomoci těchto technologií se v relativně krátké době, při potřebě relativně malého výkonu pro klasifikaci, rozpoznávají objekty

snímané pomocí fotoaparátů mobilních telefonů a podobně. V nedávné době byl pro tyto účely vydán framework s názvem DeepBelief framework, který je určený přímo pro využití těchto technik na mobilních telefonech.

Jak je vidět, existuje velké množství těchto sítí, přičemž z nich často využívanou sítí je perceptronová síť, kterou si blíže popíšeme.

2.5.1 Perceptron

Samotný perceptron je nejjednodušším modelem dopředné neuronové sítě, kterou sám o sobě tvoří. Původní využití perceptronu bylo pro množinách, které jsou lineárně separovatelné. Díky pozdějšímu použití ve vícevrstvých neuronových sítích se možnost použití perceptronu rozšířila. Perceptron funguje tím způsobem, že jako vstupy přijme vektor (sadu vstupů) a jeho výstup je definován jako

$$y = S(\sum_{i=1}^{N} w_i x_i + \Theta) \qquad 1.1$$

Přičemž S je sigmoidální výstupní (může být i skoková) funkce, w jsou váhy, x značí vstupy a y je samotný výstup neuronu.

Perceptrony je možné spojovat, a vytvořit tak perceptronovou neuronovou síť s různým počtem vrstev.

2.5.2 Vícevrstvá perceptronová síť

Vícevrstvá neuronová síť je model dopředné sítě, který převádí sadu vstupů sítě na vhodné výstupy. Skládá se z více vrstev o určitém počtu neuronů, přičemž každý neuron každé vrstvy je spojen s každým neuronem další vrstvy. Kromě vstupní vrstvy sítě, je každý uzel v síti tvořen neuronem s nelineární aktivační funkcí. Uzly vstupní sítě slouží pro možnost přiložení vstupů na síť. Vícevrstvá perceptronová síť využívá algoritmu backpropagation pro umožnění jejího učení. Takovýto model sítě umožňuje klasifikaci lineárně neseparabilních problémů.

2.5.3 Backpropagation

Algoritmus backpropagation je algoritmus používaný při učení perceptronové sítě. Metoda, kterou využívá, je nazývána metoda učení s učitelem, což je odvozeno od nutnosti klasifikace dat použitých pro trénování neuronové sítě. Trénování probíhá za pomoci trénovací množiny, která je tvořená sadou vstupů sítě a tzv. labelem – požadovaným výstupem sítě pro předložené vstupy. Díky sadě vzorků tvořících trénovací množinu může být síť naučena. Další množinou je trénovací množina, která se tvoří stejným způsobem, ale slouží k otestování neučené neuronové sítě.

Základní vlastností algoritmu backpropagation je zpětná propagace chyby do vnitřních vrstev sítě. Jednotlivé kroky algoritmu jsou následující:

1. Náhodně inicializujeme váhy v neuronové síti
2. Vezmeme pár vstup-výstup z trénovací množiny dat a část vstup přiložíme na vstup sítě
3. Vypočteme výstupy sítě
4. Porovnáme výstup sítě s částí výstup z páru vstup, výstup (vypočteme odchylku)
5. Na základe odchylky upravíme o kousek váhy sítě (získáme o něco lepší řešení)
6. Pokud jsme ještě nevyčerpali celou trénovací množinu, pokračujeme bodem 2.
7. Pokud je průměrná chyba pres celou trénovací množinu vyšší, než požadované kritérium, pokračujeme bodem 2
7. Sít je naučena, konec

(převzato z Neuronové sítě a výpočetní inteligence [15])

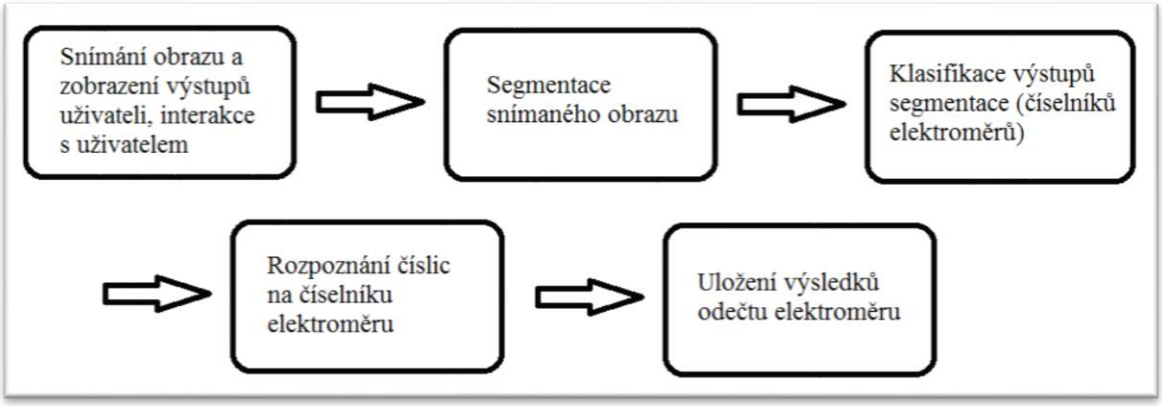
3 Návrh řešení

V souladu s popsanými nároky na funkcionality aplikace a popisem algoritmů a technik zmíněných v kapitole 2 Teorie, můžeme navrhnout systém pro rozpoznávání. Je třeba specifikovat konkrétní způsob implementace a použité vývojové nástroje, zvolit využívané algoritmy a zdroj, odkud jsou využívány. Návrhem grafického rozhraní

a ovládání aplikace dosáhneme specifikace, podle které můžeme provést implementaci aplikace. V této kapitole je tedy popsán návrh jejích jednotlivých částí.

3.1 Funkční bloky aplikace

Samotná aplikace se bude skládat ze čtyř hlavních funkčních bloků. Těmi jsou snímání obrazu a zobrazení výstupů uživateli, segmentace snímaného obrazu, klasifikace výstupů segmentace (číselníků elektroměrů), rozpoznání číslic a uložení výsledků odečtu elektroměru. Vizualizace těchto hlavních funkčních částí aplikace je vyobrazena na obrázku 8.



Obrázek 8 – blokové schéma aplikace

3.2 Způsob implementace

Počáteční vývoj aplikace bude prováděn pro počítač, čímž bude usnadněno ladění nastavení a fungování aplikace a hledání přijatelných hodnot parametrů jednotlivých algoritmů. Následně bude implementována verze pro mobilní zařízení, kde potřebné hodnoty parametrů a nastavení budou přizpůsobeny mobilnímu telefonu, za účelem dosažení co nejlepších výsledků rozpoznávání na mobilním zařízení. Takto přizpůsobené jádro mobilní aplikace bude připojeno ke grafickému rozhraní, umožňujícímu uživateli interakci s rozpoznávacím procesem.

Z důvodu složitosti a časové náročnosti vlastní implementace segmentačních i ostatních algoritmů je vhodné využít existujících implementací, to znamená knihovny

a frameworků. Navíc tento přístup mívá tu výhodu, že hotová řešení jsou funkční, otestována a správně implementována.

3.3 Implementační nástroje

Systém bude implementován v programovacím jazyce Java a bude možné ho spustit jak na běžném počítači, tak na mobilním zařízení. Pro implementaci mobilní aplikace bude zdrojový kód jemně upraven a přizpůsoben platformě Android. Výběr této platformy je vhodný kvůli minimální nutnosti úpravy kódu při implementaci pro mobilní zařízení a současně, jak je popsáno v kapitole 3.5 Segmentace, knihovna zvolená pro implementaci segmentační části, OpenCV, podporuje platformu Android i desktopovou Javu. Neméně zásadní argument je ten, že mobilní telefon dostupný pro vývoj aplikace je LG Google Nexus 5 s operačním systémem Android. Jako vývojové prostředí je zvolen Eclipse ve verzi Juno (verze 4.2), s podporou vývoje pro platformu Android. Bližší informace o vývojovém prostředí a jeho nastavení jsou popsány také v kapitole 4.1 Příprava pro vývoj. Vývoj aplikace bude probíhat pod operačním systémem Microsoft Windows ve verzi 8.1.

3.4 Vymezení aplikace

Způsob průběhu rozpoznávání zvolený pro realizaci odečtu je rozpoznávání v reálném čase pomocí ideálních snímků získaných při snímání elektroměru fotoaparátem mobilního telefonu. Při vyhodnocení rozpoznání stavu elektroměru se rozpoznávání pozastaví, uživateli se zobrazí výsledek s možností uložení a editace a opětovného spuštění rozpoznávání. Aplikace pro mobilní telefon je, jak již bylo zmíněno, vyvíjena na LG Google Nexus 5 s operačním systémem Android ve verzi 5.0.1 (Lollipop). Aplikace podporuje Android API od levelu 14, tzn. Android 4.0 (Android Ice Cream Sandwich).

Data využívaná při vývoji a testování aplikace jsou přiložena v příloze 1. Jsou to stejná data, které byla využita pro bakalářskou práci Rozpoznávání naměřených hodnot [1].

Problematika řešená ve zmíněné bakalářské práci je do značné míry shodná s částí s jednou z problematik této diplomové práce.

Pro vývoj aplikace uvažujeme v potaz dva typy elektroměrů – jeden analogový a jeden digitální, s tím, aby byl systém připraven pro případnou implementaci podpory rozpoznávání dalších typů elektroměrů. Cílem práce není implementovat rozpoznávání pro všechny typy elektroměrů, ale spíše ukázat směr a dát možnost pro případné rozšíření podpory podobných typů elektroměrů. Ukázka vizuálního typu podporovaných elektroměrů, vymezených pro vývoj mobilní aplikace, je znázorněna na obrázku 9.

Obrázek 9 – ukázka vizuálních typů podporovaných elektroměrů

3.5 Segmentace

Algoritmy pro segmentaci obrazu budou implementovány za pomoci knihovny OpenCV, která je pro potřeby systému ideální, jak vyplývá z kapitoly 2.3.1 Knihovny počítačového vidění. Podporuje zvolené platformy pro implementaci, čímž je minimalizován počet a náročnost úprav nutných při konverzi kódu z aplikace pro PC do aplikace pro Android. Zásadním faktem je také to, že téměř všechny algoritmy a techniky popsané v kapitole 2. Teorie, jsou v knihovně již implementovány, nebo je velice snadné je za její pomoci implementovat. Nutný je jejich správný výběr a kombinace a samozřejmě vhodné nastavení jejich parametrů.

Pro každý vizuální typ číselníku elektroměru je vhodné použít relevantní typ segmentace, jelikož detekce číselníku i separace číslic je závislá na vizuálních vlastnostech daného elektroměru. Aplikace bude tedy mít pro každý jeden typ číselníku

modul, a bude umožňovat přidání dalšího modulu pro případ budoucího rozšiřování systému.

3.5.1 Detekce číselníku (plochy s údajem o stavu elektroměru)

Jelikož navrhovaný systém uvažuje moduly pro digitální a analogové elektroměry, zvolíme segmentační metody pro každý z těchto typů elektroměrů.

3.5.1.1 Analogové

Prvotním záměrem, jak detekovat číselník analogového elektroměru, bylo využití Houghovy transformace (kapitola 2.3.3.3 Využití Houghovy transformace), na jejímž základě následně proběhne hledání čtverců. Technika se však po implementaci ukázala jako nevhodná pro tento typ elektroměrů a byla nahrazena metodou využívající hledání shluků vertikálních hran (kapitola 2.3.3.1 Detekce vertikálních hran). Ovšem i druhá technika se po implementaci ukázala jako nevhodná, pro analogový typ elektroměrů. Blíže jsou tyto skutečnosti popsány v kapitole 4.5.1.1 Detekce číselníku.

V úvahu byla brána ještě řešení, která jsou založena na předpokladu, že číselník analogového elektroměru obsahuje červenou barvu, nebo že čísla na číselníku černé barvy jsou bílá. Takový způsob segmentace je zajisté také využitelný, dokonce byl i částečně rozpracován, avšak z logiky věci není zrovna vhodný. Omezovali bychom se zaprvé na segmentaci závislé na barvě a tím bychom ještě více snížili univerzálnost segmentační části systému. Navíc analogový elektroměr má červené pole v číselníku často různého tvaru a velikosti. Zkrátka by tento přístup mohl přinést více škody než užitku.

Jako finální zvolený způsob implementace byla tedy zvolena technika založená na využití hledání kontur v obraze, pomocí nichž je následně za pomoci kontroly rozměrů každé kontury provedena detekce možných částí obrazu reprezentujících číselník.

3.5.1.2 Digitální

U digitálních elektroměrů se dostáváme k problému, že čísla, která potřebujeme z číselníku rozpoznávat, nejsou jedinými čísly nebo znaky na číselníku. Kdybychom tak postupovali stejně jako u analogových elektroměrů, dostali bychom se do stavu, kdy máme vyříznutý obrázek číselníku, ale v něm je obsaženo více údajů než jen ty, které potřebujeme. To nás vrací opět na začátek procesu segmentace, protože potřebujeme z obrazu získat pouze čísla tvořící údaj o stavu elektroměru. Metodu hledání kontur a detekci možných částí obrazu, které jsou kandidáty na číselník, využitou u analogových elektroměrů, tak není možné použít.

Na rozdíl od analogových elektroměrů je ale dobře využitelná metoda hledání shluků. Čísla na číselníku digitálního elektroměru mají do značné míry podobné vlastnosti, jako čísla na SPZ uvedená v práci Zpracování a rozpoznání obrazu [3] kde tato technika byla úspěšně využita. Čísla na číselníku jsme tak schopni detekovat i bez předchozí detekce polohy celého číselníku, jehož polohu v našem systému stejně nepotřebujeme, protože nás z číselníku zajímá jen číselná hodnota stavu elektroměru. V kapitole 4.5.2.1 Detekce čísel na číselníku, je popsána implementace detekce číslic digitálního elektroměru za přispění techniky inspirované metodou top-hat (kapitola 2.3.3.2 Top – hat).

3.5.2 Separace číslic

Efektivní a vhodná technika pro separaci číslic z detekované plochy číselníku je využití hledání kontur. Jsme pomoci ní schopni dosáhnut toho samého, jako pomocí horizontálního histogramu, navíc můžeme odfiltrovat další části obrazu, které jsou pravděpodobně šum nebo jiné pro nás nadbytečné části obrazu, kterých se chceme zbavit. Tento přístup je využit pro oba typy elektroměrů.

3.5.3 Získání atributů

Získání atributů z obrázků čísel pro využití neuronovou sítí můžeme nechat pro oba typy elektroměrů taktéž shodné. Podstatné je čísla nejprve transformovat do určité velikosti a z nich pak získávat sadu atributů. Kdybychom toto neudělali, nemusíme

zaprvé získat pro všechna čísla stejný počet atributů, což je při použití atributů jako vstupů neuronové sítě nutné. Za druhé by získané atributy nemusely pro jednotlivé obrázky mít informační hodnotu stejného typu.

Pro získání atributů tedy obrázek převedeme do normované velikosti a následně na něj aplikujeme metodu rozdělení pásma, která byla v bakalářské práci Rozpoznávání tištěného písma [2] úspěšně využita a otestována.

3.6 Klasifikace

Techniky klasifikace využijeme v navrhované aplikaci ke dvěma věcem. První bude snížení četnosti chybné segmentace a druhou pak rozpoznávání číslic na separovaných obrázcích čísel.

3.6.1 SVM

Vektorový stroj použijeme pro klasifikaci detekovaných číselníků segmentací. Určíme tak, jestli detekovaná část snímku, která je kandidátem na číselník, je opravdu číselníkem. Vektorový stroj tak vlastně bude třídit detekované části obrazu podle toho, jak klasifikuje, že se jedná o část obrazu, na němž je číselník, nebo nejedná.

Pro implementaci SVM využijeme knihovny OpenCV, která jeho použití umožňuje.

3.6.2 Neuronová síť

Neuronovou síť použijeme pro rozpoznávání separovaných číslic z číselníku. Využijeme vícevrstvé perceptronové sítě (kapitola 2.5.2 Vícevrstvá perceptronová síť) využívající algoritmus backpropagation (kapitola 2.5.3 Backpropagation). Tento typ sítě byl využit i v práci Systém pro rozpoznávání tištěného písma [2] a ukázal se, jako úspěšný. Ve zmíněné práci byla neuronová síť implementována pomocí programu RapidMiner. Cílem navrhované mobilní aplikace je vytvořit co nejvíce nezávislý, celistvý systém, který si vystačí s výpočetním výkonem i paměťovou dispozicí mobilního telefonu. Pro implementaci neuronové sítě tak volíme možnost vlastní

implementace. Neuronová síť bude implementována v programovacím jazyce Java a vytvořena na desktopovém PC, následně integrována do mobilní aplikace.

3.7 Návrh aplikace na platformě Android

Jak již bylo zmíněno, pro vývoj mobilní aplikace je třeba mít správně nakonfigurované vývojové prostředí, zprovoznit knihovnu OpenCV na platformu Android, a pro případné potřeby (i co se navazujícího vývoje aplikace týče) připravit mobilní aplikaci na možnost použití programovacího jazyka C++. Aplikaci tedy bude podporovat JNI (Java Native Interface), což je způsob, jakým v prostředí Android dokáže Java kód interagovat s C++ prostředím. Možnost vývoje v programovacím jazyce C/C++ otevírá další možnosti do budoucna, co se výkonnosti i rozšíření implementace týče.

3.7.1 Popis funkčních celků

Mobilní aplikace samozřejmě musí umožňovat ovládání a interakci s uživatelem pomocí grafických komponent. Musí být umožněno snímání fotoaparátem mobilního telefonu a zobrazení snímaného obrazu, informování uživatele o průběhu rozpoznávání, ovládání průběhu snímání elektroměru, včetně možnosti uložení nebo nového sejmutí stavu elektroměru.

Jednotlivé funkční celky aplikace tedy jsou:

- GUI s podporou snímání fotoaparátem a interakcí uživatele
- Segmentační část využívající obraz snímaný fotoaparátem pro detekci číselníku a separaci čísel z číselníku
- Klasifikační část pro klasifikaci, zda výsledek segmentace číselníku je opravdu číselník, nebo ne
- Klasifikační část pro rozpoznávání textové hodnoty číslic z obrázků číslic
- Sestavení a vyhodnocení výsledku sejmuté hodnoty elektroměru

Jednotlivé celky spolu interagují následovně. Fotoaparátem je snímán obraz, ten je pomocí GUI zobrazen uživateli a zpřístupněn pro segmentaci. Segmentační část

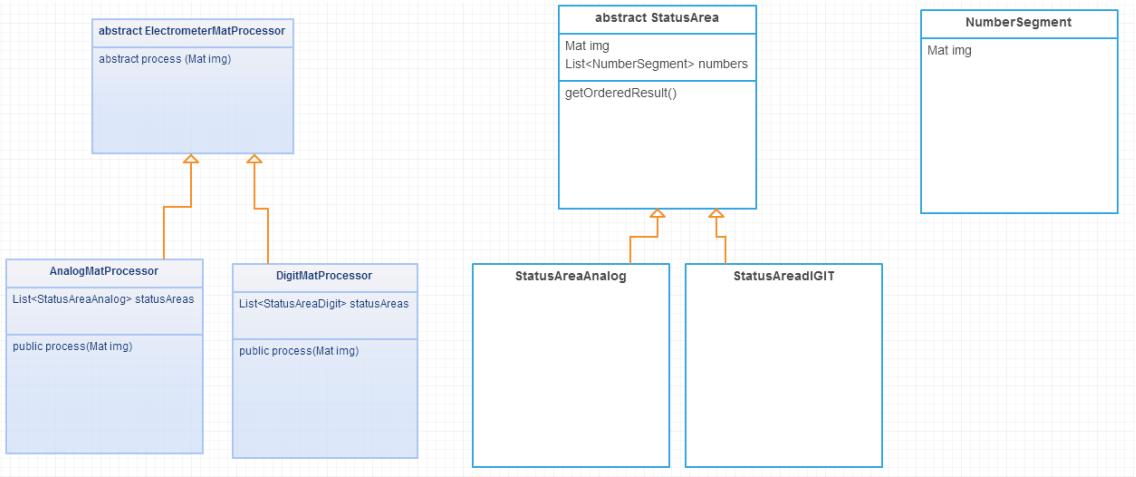
aplikace detekuje kandidáty na číselník a ti jsou předloženi SVM klasifikátoru, který odfiltruje výseky snímků, které nebyly vyhodnoceny jako obrázky číselníků. Z pozitivně klasifikovaných obrázků číselníků jsou segmentační částí aplikace vyseparovány jednotlivá čísla, která jsou předána klasifikační části pro rozpoznávání číslic pomocí neuronové sítě a výsledky jednotlivých rozpoznaných číslic jsou uchovány. V poslední části aplikace je pak provedeno sestavení celkového výsledku rozpoznání a umožněno uživateli pomocí GUI dokončit proces odečtu elektroměru.

3.7.2 Struktura aplikace a návrh tříd

S ohledem na využití programovacího jazyka Java, který je objektově orientovaný, definujeme strukturu aplikace. Abychom dosáhli modulárnosti segmentační části aplikace a umožnili tak snadné přidávání podpory dalších typů elektroměrů, vytvoříme třídu zastřešující tyto moduly a každý specifický modul z ní bude zděděn. V našem případě budou existovat dvě třídy, jedna pro analogové a druhá pro digitální elektroměry. Obě tyto třídy budou dědit zmiňovanou nadřazenou třídu představující abstrakci segmentačních modulů. Ta bude dále definovat metodu pro zpracování obrazu a jeho segmentaci, kterou je nutné implementovat v každém modulu zajišťujícím segmentaci snímku z elektroměru.

Dále je třeba modelovat reálné vlastnosti elektroměrů, to znamená, že vytvoříme třídu reprezentující číselník elektroměru, která bude obsahovat kolekci jednotlivých čísel elektroměru. Pro čísla bude vytvořena další třída, ve které se bude uchovávat separovaný obrázek čísla z číselníku elektroměru, výsledek rozpoznané hodnoty čísla na obrázku a pozice čísla na číselníku. Obrázek 10 znázorňuje popsaný

návrh tříd aplikace.



Obrázek 10 – návrh tříd aplikace

3.7.3 GUI a interakce s uživatelem

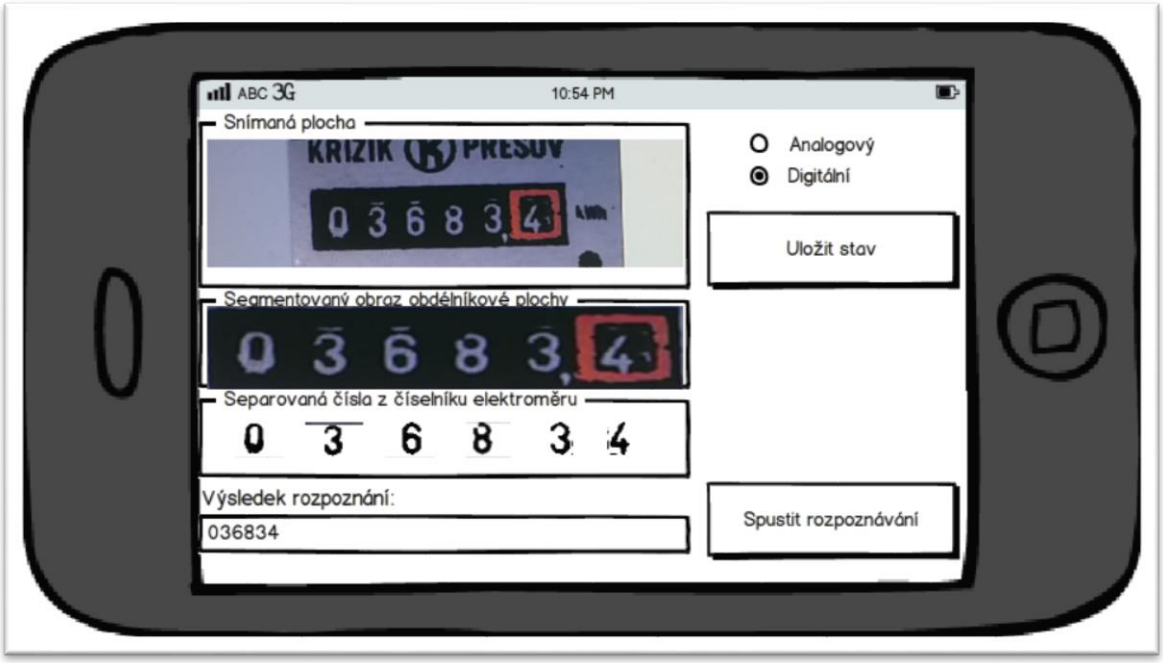
Při procesu rozpoznávání se v reálném čase bude zobrazovat obraz snímaný fotoaparátem a podstatné výstupy získané segmentačním procesem a následným provedením rozpoznání získaných číslic. Aby se aplikace jevila pro uživatele deterministická, pozastavíme proces rozpoznávání ve chvíli, kdy dojde k vyhodnocení získání výsledku snímání. Proces musí být možno opětovně spustit na žádost uživatele, abychom v případě potřeby mohli provést nové snímání. Uživateli také umožníme přepnutí segmentačního modulu – určení, jaký typ elektroměru je právě snímán. Dále je třeba umožnit editaci a uložení potřebných hodnot.

Po spuštění aplikace se uživateli zobrazí GUI aplikace, která bude mít následující prvky:

- okénko s obrazem snímaným fotoaparátem, v němž se v případě detekce číselníku bude jeho detekce indikovat
- prostor, v němž se při detekci číselníku zobrazí vyseknutý obrázek číselníku
- obrázky separovaných čísel z detekovaného číselníku, které se zobrazí pod obrázkem detekovaného číselníku
- textová podoba rozpoznaných čísel na číselníku s možností jejich editace
- tlačítko pro spuštění rozpoznávání ze snímaného obrazu

- tlačítko pro vyvolání okna pro možnost uložení údajů

Návrh GUI v podobě wireframu můžeme vidět na obrázku 11.



Obrázek 11 – návrh GUI

Po seznámení se s problematikou, popisu využívaných algoritmů, specifikaci aplikace a jejím návrhu, můžeme přistoupit k samotné implementaci.

4 Implementace

4.1 Příprava pro vývoj

Na začátku implementace je třeba nastavit vývojové prostředí tak, abychom byli schopni využívat knihovnu OpenCV, nasadit kompilovaný kód na mobilní zařízení s podporou knihovny OpenCV a inicializovat knihovnu. V příloze 2 nalezneme popis nastavení vývojového prostředí Eclipse, včetně programů, balíků a doplňků nutných pro vývoj. Po nahrání aplikace do mobilního telefonu dojde automaticky k instalaci

OpenCV knihovny do mobilního telefonu. Tato instalace je pro běh aplikace vyžadovaná.

4.2 Třídy a jejich popis

Třída reprezentující zpracování obrazu snímaného elektroměru se jmenuje `ElektrometerImageMatProcessor`. Tato třída je abstraktní a dědí od ní ostatní třídy, které představují konkrétní typ elektroměru. V našem případě od této třídy dědí třídy `AnalogElProcessor` a `DigitElProcessor`. Toto je podstatné pro případné rozšiřování aplikace o podporu dalších typů elektroměrů. `AnalogElProcessor` a `DigitElProcessor` vlastně představují segmentační moduly pro systémem podporované elektroměry. Třída `ElektrometerImageMatProcessor` implementuje abstraktní metodu `public abstract Mat process(Mat input)`. Datový typ `Mat` v prostředí knihovny OpenCV představuje maticovou reprezentaci obrázku. Každý segmentační modul aplikace tedy musí implementovat metodu `process(Mat input)`, která se stará o zpracování obrazu, předaného parametrem `input`. Třída `ElektrometerImageMatProcessor` má ještě třídní proměnnou pro uchování aktuálně zpracovávaného obrázku, proměnnou pro nastavení cesty ke složce s daty, které aplikace vyžaduje pro svůj běh a proměnné s aktuálně používanými klasifikačními modely, to znamená s model neuronové sítě a SVM. Tyto popsané třídy, vlastnosti a metody jsou minimální nutné aspekty, jejichž znalost je nezbytná pro modifikaci aplikace a porozumění její funkčnosti. Konkrétní názvy proměnných budou zmíněny dále, vždy v části, která se jich týká.

Třída `ElektrometerImageMatProcessor` ještě implementuje další metody pro práci datovými sadami, které využívá jak SVM, tak neuronová síť. K jejich popisu se také dostaneme. Znalost těchto metod je neméně důležitá pro schopnost ovládání systému jako celku a pochopení návaznosti jednotlivých funkcionalit.

4.3 GUI

V prostředí aplikací pro platformu Android je při spuštění aplikace využita třída typu `Activity`, nebo její rozšíření. Tato třída vyžaduje implementaci metody `onCreate()`, která je volána po spuštění aplikace. Životní cyklus aplikace pro Android a jeho popis pro třídu typu `Activity` lze nalézt v Android dokumentaci [7]. Je tedy implementována třída `OpencvLoader`, využívaná po spuštění aplikace. V této třídě existuje proměnná, obsahující aktuální segmentační modul aplikace. Tato proměnná je tedy typu `ElektrometerImageMatProcessor`, její název je `elMatProcess`. Díky této proměnné jsme schopni volit modul, kterým se bude provádět segmentace a ten za běhu aplikace umožnit uživateli změnit.

Ve složce res/layout najdeme soubor `aktivity_opcv.xml`, který je využíván pro vytvoření GUI po spuštění aplikace. Můžeme v něm nalézt definici všech komponent GUI. Implementované GUI vychází z popsaného návrhu v kapitole 3.7.3 GUI a interakce s uživatelem. Za zmínku stojí zmínit jakým způsobem je indikováno, jaký segmentační modul je aktuálně použit. Pro analogové elektroměry je pozadí části GUI, kde se zobrazuje výsek číselníku elektroměru černé, u digitálních elektroměrů pak šedozelené. Tyto barvy zhruba odpovídají barvám pozadí číselníku na elektroměrech. Aby bylo uživateli umožněno segmentační modul změnit při běhu aplikace, součást GUI je tlačítko typu `Switch`, sloužící jako přepínač. Při přepnutí dojde ke změně z analogového segmentačního modulu na digitální a naopak. Tato změna je realizována vytvořením nové instance třídy představující segmentační modul. Tato instance je nastavena do proměnné `elMatProcess` pomocí níž aplikace volá metodu `process(Mat input)`.

Aby uživatel mohl ovlivnit, jaký je vstup do procesu rozpoznání realizovaný parametrem `input`, musíme mu zobrazit fotoaparátem snímaný obraz. K tomu není využito pro Android standardního přístupu k aktivaci fotoaparátu [8], ale je využito rozhraní z knihovny OpenCV, připravené pro účely zpracování obrazu ze snímaného obrazu fotoaparátem mobilního telefonu. Rozhraní se jmenuje `CvCameraViewListener2` a aktivita `OpencvLoader` ho implementuje. Toto rozhraní vyžaduje implementaci metody `onCameraFrame(CvCameraViewFrame inputFrame)`,

která je volána vždy, když je fotoaparátem každý sejmut jeden snímek (frame). V této metodě zbývá zavolat námi implementovanou metodu `process(Mat input)` z aktuálního segmentačního modulu a segmentace snímaného obrazu se tak začne realizovat. Díky rozhraní `CvCameraViewListener2` lze využít předpřipravené komponenty `JavaCameraView`, která je definovaná v `aktivity_opcv.xml` a zobrazuje v GUI uživateli obraz snímaný fotoaparátem telefonu. Abychom efektivně odfiltrovali ze snímané plochy co největší část obrazu, který nás nezajímá, a tedy docílili zaměření číselníku, je ze snímaného obrazu zobrazován pouze výsek o velikosti, která poměrem odpovídá zhruba poměru výšky a šířky číselníku elektroměru. Uživatel tak na display telefonu vidí obdélník, do kterého je třeba číselník zaměřit a v tomto vymezeném obdélníku dále probíhá detekce display a další části segmentace. Teoreticky je tak implementováno, jakým způsobem se docílí segmentace obrazu pro určitý typ elektroměru, prakticky je třeba zajistit ještě inicializaci knihovny OpenCV, jejíž implementované algoritmy, datové typy, atd. by jinak nešly volat.

4.4 Inicializace OpenCV

Knihovna OpenCV je inicializována pomocí callbacku, konkrétně vytvořením objektu typu `BaseLoaderCallback`, při jehož inicializaci také aktivujeme zobrazování snímání komponentou `JavaCameraView`. Pro dokončení inicializace je pak v metodě `onResume()` zavolána metoda `OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_9, this, mLoaderCallback)`, kde `mLoaderCallback` je objekt definující zmiňovaný callback. Díky tomu je inicializace OpenCV dokončena a jsme připraveni na její využívání pro segmentaci a další účely systému. Způsoby a bližší informace o inicializaci OpenCV jsou opět k dispozici v OpenCV dokumentaci [9].

4.5 Segmentace – zpracování obrazu

Matice uložená v parametru `input`, reprezentující snímaný obraz, je zpracována za pomoci segmentačních algoritmů. Popis interpretace obrazu za pomoci datového typu `Mat` můžeme nalézt v dokumentaci knihovny OpenCV [9]. Pro každý typ elektroměru je

využito jiných segmentačních algoritmů, rozdělíme tedy popis segmentace pro analogové a digitální elektroměry.

4.5.1 Analogové elektroměry

4.5.1.1 Detekce číselníku

Provedená implementace pro získání obrázku číselníku ze snímaného obrazu využívala Houghovu transformaci. Konkrétně pomocí ní byly detekovány přímky v obraze. Myšlenka detekce byla založená na tom, že nalezneme přímku horní a spodní hrany analogového číselníku elektroměru, poté provedeme opětovné hledání přímek v obraze, ale s „jemněji“ nastavenými parametry a tím získáme přímky pravé a levé hrany číselníku. Detekcí průniků přímek nalezených při prvním použití Houghovy transformace s přímkami nalezenými při druhém použití Houghovy transformace a ověřením, jestli úhel svíraný těmito přímkami je mezi 85° a 95°, získáme obrys detekovaných obdélníků v obraze. Nalezení požadované části obrazu, číselníku, bylo díky této metodě úspěšné, avšak velmi výpočetně náročné a tak pro mobilní zařízení nepraktické. Jako hlavní problém se však ukázalo, že na mnoha dostupných datech tato metoda nefunguje, a to z důvodu častých odlesků na snímcích elektroměru a z důvodu, že při vizuálně nerovné hraně číselníku nedojde k detekci přímky, přičemž na mnoha obrázcích jsou tyto nerovnosti viditelné. Houghova transformace pro detekci přímek v obraze lze pomocí OpenCV aplikovat voláním metody `Imgproc.HoughLines()` taktéž popsané v dokumentaci OpenCV [9].

Detekce za pomoci hledání červeného místa v obraze byla vyzkoušena za využití metody `Core.inRange()`, která je pro tyto účely připravena. Pomocí metody odfiltrujeme z obrazu všechny barvy, které nevyhovují rozsahu barev v obraze definovaným v barevném modelu HSV. Určíme tedy maximální a minimální hodnotu barvy, kterou akceptujeme, tím je definován rozsah barev, a body obrazu nevyhovující tomuto rozsahu jsou odfiltrovány. Tímto způsobem bylo nalezeno požadované místo v obraze, místo v jehož okolí je číselník, avšak na mnoha fotografiích elektroměrů je červená barva na číselníku různého odstínu a jasu a tak detekovaná plocha má často různé parametry a někdy připomíná shluk rozsypaných bodů, podobných dalším bodům

v obraze, které se v něm po segmentaci občas vyskytují. Ukázalo se tak, že metoda založená na hledání určité barvy není dostatečně univerzální.

Finální implementace tedy využívá hledání kontur v obraze. Kontury vyhledáme pomocí metody `Imgproc.findContours()`. Vlastnost číselníku analogových elektroměrů je totiž ta, že tvoří uzavřenou stejnobarevnou plochu, což znamená, že jedna z nalezených kontur je i kontura číselníku, který hledáme. Abychom odfiltrovali ostatní kontury, je třeba nalézt charakteristiku specifickou pro konturu číselníku a odstranit ostatní kontury, které nesplňují tuto charakteristiku. Byla zvolena charakteristika poměru velikosti šířky a výšky kontury, což je vlastně reálný poměr číselníku. Metoda `boolean verifyStatusAreaSizes(RotatedRect rect)` implementovaná v segmentační třídě analogových elektroměrů ověří, zda zmiňovanou charakteristiku kontura splňuje. Parametrem metody je typu `RotatedRect`, což je obdélníkový obrys kontury. Metodou odfiltrujeme všechny kontury, které nesplňují charakteristiku kontury číselníku, a tím docílíme získání sady kontur (v ideálním případě jedna pro jeden číselník na snímku), díky kterým jsme schopni vytvořit výsek číselníku. Z originálu obrazu vysekneme tu část, která odpovídá obdélníkovému obrysu kontury (`RotatedRect`) a narovnáme obraz vodorovně, abychom odstranili případnou rotaci způsobenou snímáním. Obraz číselníku je pak uložen v objektu třídy `StatusAreaAnalog`. Pro získání kandidátů na číselník byla implementována metoda `getAreaCandidates(Mat eyeRect)`.

Výsledná implementace není závislá na barvě číselníku, pouze na tom, aby byla barva pro celý číselník stejná, respektive aby číselník byl alespoň kouskem propojen stejnou barvou, bez celkového přerušení jinou barvou.

Toto jsou základní informace pro orientaci v implementované technice segmentace obrazu za účelem detekce číselníku analogového elektroměru. Konkrétní implementaci lze vidět v přiloženém zdrojovém kódu aplikace [příloha 3].

Kandidáti na číselník elektroměru jsou klasifikováni pomocí SVM. Popis implementace SVM je v kapitole [4.6 SVM].

4.5.1.2 Detekce a separace číslic

Pro separaci číslic z detekovaného obrázku číselníku je třeba detekovat jednotlivá čísla na obrázku, zbavit obraz nežádoucího šumu (malých částí obrazu, které znehodnocují hodnotu informace, kterou se snažíme získat) a poté číslice separovat. Využijeme techniky podobné jako při detekci číselníku. V obraze číselníku jsou vyhledány kontury a za pomoci metody `boolean verifyNumberSizes(Mat r)` jsou odfiltrovány nežádoucí kontury. U analogových číselníků se však ukázalo, že je třeba provést modifikaci tohoto způsobu separace číslic, protože vyhledané kontury neodpovídají tvarům číslic na číselníku. Způsobeno je to tím, že číslice na snímaném obraze nejsou ideální a často dochází ke slitkám s okolím a tak získání kontur ne jen číslic, ale i jejich okolní plochy. Čísla nemají čistě bílou barvu, ale často jsou zašedlá, zažloutlá a porušená, tzn. netvoří jeden souvislý obrazec.

Pro možnost provedení separace číslic je tak na ploše číselníku aplikován detektor hran a to v modifikované podobě, konkrétně s využitím Otsunovy prahovací metody. Pomocí Otsunova algoritmu (kapitola 2.3.2.5 Cannyho detektor hran pomocí Otsunovi prahovací hodnoty) se tak vypočítá ideální hodnota prahu, a za její pomoci jsou detekovány hrany v obraze číselníku. Poté až je aplikováno hledání kontur a ověření detekce číslic, po kterém následuje vyseknutí jednotlivých číslic z číselníku, přičemž obrázek každé číslice je uložen v objektu třídy `NumberSegment` a spolu s obrázkem uchováváme i x a y souřadnice levého horního rohu obdélníku, který je obrysem kontury čísla. Současně je tento objekt přiřazen do kolekce číslic v objektu uchovávajícím obrázek číselníku, ze kterého byly číslice separovány.

Pro získání kolekce separovaných čísel byla implementována metoda `segmentStatusAreaAndExtractNumbers(StatusArea statusArea)`.

Ukázka separace číslic na analogovém elektroměru je zobrazena na obrázku 12.

Obrázek 12- separované číslice analogového elektroměru

4.5.2 Digitální elektroměry

4.5.2.1 Detekce čísel na číselníku

Jak již bylo zmíněno, u digitálních elektroměrů nedetekujeme celý číselník, ale pouze jeho část, tvořenou z číslic udávajících stav elektroměru. Za tímto účelem byla ve třídě `DigitElMatProcessor` implementována metoda `segmentLinkedNumbers(Mat img_gray)`, jejíž výstupem je černý obrázek, na němž jsou spojené bílé obdélníky vytvořené na pozici jednotlivých čísel. Souvislá bílá plocha pak slouží pro dokončení detekce výseku číselníku digitálních elektroměrů.

Implementace této techniky byla inspirována technikou top-hat využívanou pro detekci číslic SPZ pomocí hledání shluků vertikálních čar (kapitola 2.3.3.1 Detekce vertikálních hran). Stejně jako v této technice, definujeme způsob segmentace obrazu za pomoci morfologických operací a poté vytvoříme masku rozdílem originálního obrazu a výstupu jeho segmentace. Konkrétně detekujeme číslice ve snímaném obrazu, opět pomocí hledání kontur a ověření jejich charakteristik a poté vykreslíme vyplněné bílé obdélníky v prostoru definovaném obdélníkovým obrysem kontury. Obdélníky nakonec

potřebujeme spojit tak, aby vytvořily celistvou plochu pokrývající námi požadovaná čísla na číselníku. Podobně jako u analogových elektroměrů provedeme na obrázku získaném metodou `segmentLinkedNumbers(Mat img_gray)` detekci kontur, které poměrem velikostí odpovídají shluku číslic na číselníku a vyřízneme z originálního snímku plochu, odpovídající obdélníkovému obrysu kontury. Zde se ale často vyskytuje ještě jeden problém. Čísla a pozadí (displej) na některých snímcích elektroměru nejsou dostatečně barevně odlišena a velký problém dělá vržený stín nebo snímek, kde rozdíl barvy pozadí a číslic na jedné části displeje je jiný, než na jeho jiné části. Potřebujeme tak upravit obraz takovým způsobem, abychom docílili zobrazení čísel jednou barvou a pozadí druhou barvou. K tomu je ideální využití thresholdingu. Obyčejný thresholding však nestačí právě z důvodu nestejného rozdílu mezi barvou pozadí a číslic v celém obraze číselníku. Bylo potřeba nalézt metodu, která určuje práh thresholdingu ne z celého obrázku, ale pro každou část obrázku zvlášť podle sousední plochy. Přesně pro tyto účely byl využit adaptivní thresholding implementovaný v knihovně OpenCV. Pro docílení požadovaného výsledku je třeba správně definovat hodnotu velikosti okolí, ze kterého se prahová hodnota vypočítává.

Stejně jako u analogových číselníků je výsledná implementace nezávislá na barvě pozadí nebo číslic. Musí být dodrženo pouze to, aby barva pozadí i číslic byla jednobarevná.

Získáme tak připravený obrázek čísel z číselníku digitálního elektroměru vhodný pro další zpracování a nahradili jsme techniku, kterou je prováděna detekce číselníku u analogových elektroměrů. Výsledný obrázek uchováme v objektu třídy `StatusAreaDigit`.

4.5.2.2 Detekce a separace číslic

Oproti analogovým elektroměrům zde probíhá separace číslic podstatně jednodušeji. To je umožněno kvalitou obrazu číslic u digitálních elektroměrů. Čísla jsou vždy stejná, nejsou nijak porušená a hlavně jsou dostatečně velká a tvoří celistvou plochu. Jelikož se ovšem jedná o čísla na sedmisegmenovém display, jsou tak tvořena černými segmenty, které nejsou spojeny. Abychom pomocí kontur byli schopni detekovat číslo, musíme

tyto segmenty spojit a vytvořit tak celistvé spojené číslo. Toho je dosaženo pomocí operace dilatace. Na obrázku 13 je zobrazeno použití této metody.

Obrázek 13 – upravený obrázek číslice pro možnost separace

Další postup je již podobný, jako u analogových elektroměrů. Aplikujeme tak algoritmus pro hledání kontur, verifikujeme kontury číslic a jednotlivé číslice vyřízneme, uložíme a přiřadíme číselníku (objekt typu `StatusAreaDigit`), kterému náleží.

4.5.3 Nalezení optimálních hodnot parametrů algoritmů

Pro aplikaci většiny segmentačních algoritmů je třeba zadat parametry, ovlivňující výsledek použití algoritmu. Jejich správná volba, nebo způsob jejich výpočtu je pro ovlivnění kvality systému naprosto klíčový. To se při vývoji odrazilo na době vývoje a času stráveném implementací. Při změně parametrů v počáteční části segmentace je ovlivněn výsledek všech následujících segmentačních technik a tak je často nutné upravit jejich parametry, přičemž pokaždé musí být ověřeno, jestli při každém kroku segmentace dosahujeme požadovaných výsledků. Pro rychlejší možnost změny parametrů a zpětné kontroly výsledků segmentace byly při vývoji do aplikace přidány komponenty, které umožnily změny parametrů za běhu aplikace. Konkrétně se jednalo o posuvníky, pomocí nichž bylo možné nastavovat číselnou hodnotu parametrů (většina parametrů, které bylo třeba měnit, bylo číselných). Tento fakt je hlavním argumentem, proč byla zvolena nejprve implementace pro desktop a následně byl kód převeden pro platformu android. V okně desktopové aplikace se snáze pracuje s nastavením parametrů a navíc bylo vytvořeno více komponent pro zobrazení obrazu, takže bylo možné sledovat výsledky procesu segmentace současně.

Důležité je zmínit způsob určení parametrů pro verifikaci velikosti obdélníků okolo kontur číselníků a číslic v nich. Pro číselníky se za pomoci pravítka změřila reálná velikost šířky a výšky číselníku (u digitálních elektroměrů šlo o šířku a výšku prostoru s požadovanými čísly v číselníku) a z ní se vypočítal poměr těchto velikostí. Stejně tak tomu bylo u číslic analogových elektroměrů, avšak u digitálních elektroměrů tento přístup nestačí. Jelikož na číslech digitálních elektroměrů provádíme morfologickou operaci, změní se tak šířka i výška výsledných modifikovaných celistvých číslic. Ideální hodnota pro šířku a výšku byla tedy nalezena za pomoci posuvníků a následné vizuální kontroly, zda byla čísla verifikována.

Tento přístup s sebou přináší výhodu nezávislosti na velikosti číselníku ve snímaném obraze, protože se vychází z poměrů, které jsou nezávisle na měřítku stejné.

4.6 SVM

Vektorový stroj je implementovaný za pomoci knihovny OpenCV konkrétně třídy `CvSVM`. Pro definování funkčnosti SVM je třeba definovat parametry, k čemuž slouží třída `CvSVMParams`. Prvním nastavovaným parametrem je typ vektorového stroje. Ten se v OpenCV nastaví pomocí konstanty `CvSVM.C_SVC` což znamená, že SVM je nastaveno na klasifikaci typu 1 [11.1]. Dalším parametrem je nastavení jádrové transformace, která je pomocí konstanty `CvSVM.LINEAR` nastavena na lineární, což je nejrychlejší nastavení. Parametr C, který určuje „cenu“ za špatnou klasifikaci, je nastaven na hodnotu 1. Pomocí třídy `TermCriteria` jsou nastavena kritéria iterativní procedury učení SVM. Zde je nastavena hodnota počtu iterací na 1000 a tolerance – minimální přesnost na 0.01. Dokumentace vektorového stroje implementovaného v knihovně OpenCV v její dokumentaci [11].

Vektorový stroje je tak připraven pro natrénování a využití k následné klasifikaci.

4.6.1 Trénování SVM

Pro natrénování vektorového stroje je třeba nejprve vytvořit množinu dat. Pro účely klasifikace číselníků je tak třeba vytvořit dvě množiny dat. Jednu, která reprezentuje

zástupce číselníků – obrázky výseků číselníků, na nich je znázorněn opravdu námi požadovaný číselník a druhou, která reprezentuje negativní zástupce číselníků – obrázky, získané segmentací pro detekci číselníků, na nichž se však ve skutečnosti číselník nevyskytuje. Pro získání těchto obrázků byly ukládány výstupy segmentace a poté ručně vytříděny podle vizuálního vyhodnocení v závislosti na tom, zda je na obrázku znázorněn číselník, nebo není.

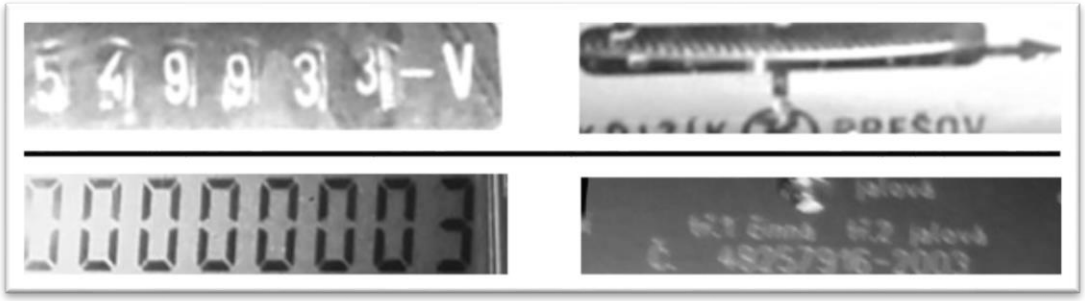
Takto připravená sada pro naučení SVM musí splňovat stejné vlastnosti. To znamená, že obrázky číselníků musejí mít stejnou velikost a mít stejné barevné vlastnosti. V našem případě se jedná o obrázky v šedotónu. Data rozdělíme do dvou složek, jedné pro reprezentanty číselníků, druhé pro negativní reprezentanty.

Pro vytvoření datové sady, se kterou umí pracovat SVM implementované za pomoci OpenCV byla implementována metoda `createDataSet(String platesDir, String noPlatesDir)` ve třídě `TrainSVM`, jejíž první parametr je cesta ke složce s obrázky číselníků a druhý parametr je cesta ke složce s obrázky, na kterých nejsou číselníky zastoupeny.

Posledním krokem je vytvoření samotného objektu reprezentujícího SVM, který provedeme pomocí volání `new CvSVM(trainSVM.trainingData, trainSVM.classes, new Mat(), new Mat(), SVM_params)` kde prvním parametrem je vytvořená trénovací množina, druhým jsou klasifikační třídy, které jsou v našem případě dvě – jedna pro zástupce číselníků, druhá pro negativní zástupce. Parametr `SVM_params` představuje množinu výše zmiňovaných nastavených parametrů.

Takto vytvořený vektorový stroj je připraven pro použití ke klasifikaci číselníků.

Ukázka snímků použitých pro trénování SVM, jak pro analogové, tak pro digitální elektroměry je zobrazena na obrázku 14



Obrázek 14 – ukázka vzorků trénovací množiny SVM

Data použitá při trénovaní SVM jsou přiložena v příloze 4.

4.6.2 Klasifikace kandidátů na číselník pomocí SVM

Pro klasifikaci výsledků segmentace za účelem detekce číselníků byla implementována metoda `getDetectedStatusAreas()`, která z kandidátů na obrázek číselníku vybere ty obrázky, na nichž bylo vyhodnoceno, že je na nich zobrazen číselník. Pro každý obrázek je za pomoci metody `predict(Mat p)` volané na objektu třídy `CvSVM`, vytvořeného při procesu trénování zjištěno, zda byl obrázek vyhodnocen jako reprezentant číselníku či nikoli. Po přetypování výstupu metody na datový typ `int` můžeme dostávat výsledky buď 0, nebo 1, přičemž výsledky s hodnotou 1 znamenají, že se jedná o obrázek číselníku. Právě tyto obrázky jsou pro nás relevantní výstupy segmentace za účelem detekce číselníku a další části systému pracují pouze s nimi.

4.7 Implementace neuronové sítě

Neuronová síť je implementována přesně pro účely vytvářeného systému, nicméně to je dáno až při jejím sestavování. Znamená to, že implementace jednotlivých prvků neuronové sítě je připravena tak, aby bylo možné jednoduše ovlivňovat zásadní parametry sítě jako je počet vrstev, neuronů v nich, atd. a to bez modifikace implementace jednotlivých prvků sítě.

Implementace prvků neuronové sítě se nachází v balíku `my.nn` kde třída `PerceptronNetwork` představuje model kompletní perceptronové neuronové sítě, třída `InputLayer` vstupní vrstvu sítě, třída `PerceptronLayer` ostatní vrstvy sítě, `InputNeuron` představuje neurony vstupní vrstvy sítě a `Perceptron` představuje ostatní neurony ostatních vrstev sítě. Pomocí těchto tříd je možné sestavit neuronovou síť, ale ještě je třeba umožnit její učení, to znamená možnost práce s množinami dat, konkrétně s testovací a trénovací množinou.

Pro vytvoření objektu představujícího jeden vzorek z datové množiny je implementována třída Sample, která umožňuje držet vstupy a výstupy daného vzorku pro účely zpracování neuronovou sítí. Množinu těchto vzorků je možné vytvořit díky třídě `DataSet` a tím sestavit trénovací nebo testovací množinu. Množiny dat jsou

Díky popsané struktuře je možné vytvořit síť o libovolném počtu vrstev a neuronů v jednotlivých jejích vrstvách. Sestavená neuronová síť je připravená pro trénování, abychom ji mohli využít k následnému rozpoznávání.

4.7.1 Učení neuronové sítě

Pro možnost využití trénovací množiny pro naučení neuronové sítě byla implementována metoda `learn()` ve třídě `PerceptronNetwork`. Do metody vstupují parametry ovlivňující proces učení neuronové sítě a jedním z parametrů je sama trénovací množina, která je metodě `learn()` předána ve formě kolekce objektu typu `DataSet`, který obsahuje kolekci objektů typu `Sample`, konkrétně `List<Sample> samples`.

Na začátku procesu učení jsou inicializovány váhy neuronů jednotlivých vrstev sítě. Poté se podle zadaného parametru ovlivňujícího počet epoch učení a minimální chyby sítě vykoná metoda `learnEpoch()`, díky které se vykoná jedna epocha učení na trénovací množině. Při provedení jedné epochy učení je využito metody `weightUpdate(double eta, double alpha)`, implementované za účelem aktualizace vah neuronové sítě v závislosti na zpětně propagované chybě neuronové sítě. Po dosažení stanoveného počtu epoch, nebo dosažení chyby sítě menší než stanovená

hodnota, je proces učení neuronové sítě ukončen a neuronová síť je tak připravena pro použití k rozpoznávání.

4.7.2 Sestavení neuronové sítě

Sestavení samotného modelu neuronové sítě je už velice jednoduchý proces, kdy vytvoříme instanci neuronové sítě třídy `PerceptronNetwork`, poté vytvoříme pro každou vrstvu neuronové sítě instanci třídy `PerceptronLayer` (vyjma vstupní vrstvy, která se vytvoří instancí třídy `InputLayer`) a všechny tyto vrstvy na sebe napojíme pomocí jejich metody `connect(PerceptronLayer layer)`, kde parametr `layer` je vrstva, kterou chceme připojit k vrstvě, na které metodu voláme. Instance neuronové sítě se vytváří s jedním parametrem celočíselné hodnoty, která určuje, kolik neuronů v dané vrstvě bude. Posledním krokem, kterým dokončíme sestavení neuronové sítě je přidání všech vrstev do kolekce vrstev, která je v objektu neuronové sítě. To uděláme voláním metody `addLayer(PerceptronLayer layer)` volané na objektu neuronové sítě.

Pro možnost naučení takto sestavené sítě vytvořme trénovací množinu metodou `createDataSetFromJsonMats(String filePath)`, implementovanou ve třídě `ElektrometerImageMatProcessor`, kde parametr `filePath` představuje cestu ke složce se serializovanými obrázky číslic využívanými pro vytvoření trénovací množiny. Popis tvorby testovací množiny je uveden v kapitole 5.1 Digitální elektroměry.

4.8 Rozpoznání číslic neuronovou sítí

4.8.1 Princip rozpoznání

Aby obrázek čísla mohl být rozpoznán, je třeba z něho získat atributy, které budou sloužit jako vstupy do neuronové sítě, na jejichž základě dokáže neuronová síť spočítat adekvátní výstup, který nás bude informovat o tom, jaké číslo neuronová síť z obrázku rozpoznala. Princip, jakým jsou získávány atributy z obrázku čísla, vede k vytvoření sady 64 čísel. To znamená, že vstupů neuronové sítě je 64, z čehož vyplývá, že pro rozpoznání našich dat je třeba sestavit neuronovou síť tak, aby ve vstupní vrstvě měla

64 neuronů. Z každého obrázku požadujeme rozpoznání jednoho čísla. Jelikož čísel je 10 (0 až 9), je počet neuronů ve výstupní vrstvě nastaven také na 10. Tím máme pro každou kategorii čísel na výstupní vrstvě jeden neuron, a jelikož prvky trénovací množiny jsou definovány sadou vstupů a jedním požadovaným výstupem, jsme tak schopni podle výstupu neuronů výstupní vrstvy určit, jaké číslo síť rozpoznala.

4.8.2 Získání vstupů pro neuronovou síť

Pro získání atributů z obrázků číslic byla ve třídě `ElektrometerImageMatProcessor` implementována metoda `getNumberImageAttributes(Mat numberImg)`. Metoda analyzuje matici reprezentující obrázek čísla, který je černobílý, a jejím výstupem je 64 čísel, celočíselného datového typu v rozsahu od 0 do 40. Obrázek čísla má výšku 64 a šířku 40 bodů a jeho matice je proložena pomyslnými obdélníky velikosti 5x8 pixelů, čímž vytvoříme mřížku, přes celý obrázek. V každém obdélníku je spočten počet černých bodů. Velikost obdélníku byla vypočítána z ověřené metody v práci Systém pro rozpoznávání tištěného písma [2], kde velikost prokládaného obdélníku byla 4x4 a velikost obrázku znaků 32x32. Poměr výšky obrázku znaku je 32/4=8, stejně tak jako výšky. Vypočítáme velikost obdélníku, jímž je náš obrázek prokládán, tedy 64/8=8, což je výška našeho prokládaného obdélníku a 40/8=5, což je jeho šířka. V každém obdélníku tak může být maximálně 8*5=40 černých bodů. Z toho nám vyplývá konkrétní charakter trénovací množiny. Obdélníků v mřížce matice obrázku je 8*8=64, což odpovídá počtu vstupů, a požadovaný výstup neuronové sítě pro tyto vstupy je složen z deseti čísel hodnoty 1 nebo 0, přičemž hodnota 1 je na pozici, která odpovídá pozici neuronu zastupujícím kategorii čísla konkrétní hodnoty. To znamená, že např. pro číslo 0 bude požadovaný výstup sítě definován jako [1,0,0,0,0,0,0,0,0,0] a pro číslo 3 to bude [0,0,0,1,0,0,0,0,0,0]. Díky vytvoření takto zadefinované trénovací množině můžeme naučit neuronovou síť tak, aby pro implementovaný systém rozpoznávala separovaná čísla číselníků.

4.9 Umožnění vyhodnocení výsledku rozpoznání

Pro konečné vyhodnocení čísla, které bylo rozpoznáno neuronovou sítí po předložení vstupů, slouží metoda `rozpoznej(double[] input)`, implementovaná ve třídě `ElektrometerImageMatProcessor`. Vstupy předloženy síti, na níž je následně vypočítán její výstup. Pro každý neuron výstupní vrstvy tak získáme výstupní hodnotu v rozsahu 0 až 1, přičemž čím více se hodnota blíží hodnotě 1, tím větší příslušnost dané kategorii zastupující výstupní neuron síť určila. Můžeme tak říct, že neuron s nejvyšší výstupní hodnotou nám značí, že číslo spadá do kategorie, kterou neuron zastupuje. Konkrétně jestliže hodnoty na výstupních neuronech jsou např. [0.001, 0.014, 0.0003, 0.0087, 0.009, 0.924, 0.03, 0.071, 0.065, 0.0754], zjistíme podle pozice nejvyšší hodnoty, což je v tomto případě 0.924, že sítí rozpoznaná číslice na obrázku je 5.

Po takto provedeném rozpoznání všech číslic separovaných z číselníku zavoláme metodu `getRozpoznanaCisla()`, implementovanou ve třídě `StatusArea` a získáme tak seřazený finální výsledek rozpoznání stavu číselníku elektroměru.

Implementace hlavní části navrhované aplikace byla popsána a zbývá doplnit popis ostatních náležitostí, jejichž potřeba byla zjištěna během vývoje, a taktéž byly implementovány.

4.10 Ukládání přenositelných dat

Pro vytvoření trénovací množiny neuronové sítě je třeba mít nejprve nasbírán dostatek dat. Konkrétně potřebujeme obrázky číslic. Stejně tak pro učení vektorového stroje potřebujeme nejprve nasbírat obrázky číselníků elektroměr. Abychom byli schopni tato data nasbírat, byla implementována metoda `saveEyeStatusAreaAndNumbers()` ve třídě `ElektrometerImageMatProcessor`, ve které proběhne uložení všech potřebných obrázků pro další zpracování. Pro volání metody bylo do GUI přidáno tlačítko, po jehož stisku se nastaví příznak pro aktivaci ukládání, čímž zajistíme volání metody při segmentaci obrazu. Konkrétně jsou ukládány obrázky snímání fotoaparátem telefonu, detekované číselníky a jednotlivá čísla na číselnících. Každý z těchto obrázků je uložen pod názvem s pro něj specifickým prefixem, podle toho, jestli jde o číselník, nezpracovaný snímaný snímek nebo číslo. Každý z těchto obrázků obsahuje číslo, které

spojuje všechny obrázky týkající se zpracování jednoho snímku – čísla číselníku, číselník i nezpracovaný snímek tak obsahují stejné číslo. Při ukládání dat z dalšího snímku se číslo inkrementuje.

Takto jsou ukládány všechny obrázky z každého snímku (framu) videa, na kterém, dojde k detekci alespoň jednoho čísla na nalezeném číselníku ve snímku, čímž můžeme nasbírat data vhodná k použití pro učení a testování neuronové sítě, případně si uložit ostatní výstupy procesu segmentace.

Další data, které je možné ukládat, jsou modely vektorového stroje a neuronové sítě. Tato možnost je takřka nutností, jelikož proces učení těchto nástrojů je výpočetně náročný a hlavně při nalezení ideálně naučeného modelu je zbytečné provádět znova a znova učení na stejných datech. Vektorový stroj i neuronovou síť je tedy možné serializovat a tím uložit do souboru, který můžeme později použít pro načtení již naučeného modelu. Vektorový stroj je konkrétně serializován do xml souboru, který je platformně přenositelný. Neuronová síť je serializována ve formátu čitelném při použití jazyku Java.

4.11 Desktopová aplikace pro podporu práce s neuronovou sítí

Po získání veškerých dat pro učení neuronové sítě a vektorového stroje, je prováděno samotné učení. U vektorového stroje je situace do značné míry jednoduchá – vytřídíme obrázky, na kterých je číselník a na kterých ne a rozdělíme do dvou složek. U neuronové sítě je však situace složitější. Čísla získaná z číselníků musíme vytřídit, následně určit, jaká číslice na obrázku je, a těchto obrázků čísel potřebujeme značné množství. Bez umožnění využití prvků automatizace je celý proces opravdu velmi zdlouhavý. Navíc neuronovou síť je třeba otestovat, což s sebou nese stejné problémy. Z těchto důvodů byla vytvořena desktopová aplikace, která umožňuje automatizaci některých kroků a urychluje třídění a pojmenovávání vytříděných dat.

Abychom mohli obrázky přenášet mezi mobilním telefonem, kde byly pořízeny a počítačem, kde je chceme analyzovat, byl pro jejich uložení zvolen formát json, který zajišťuje i platformní přenositelnost. Navíc, abychom obrázky nepřenášeli modifikované kompresními algoritmy, jsou serializovány ještě ve formě matice,

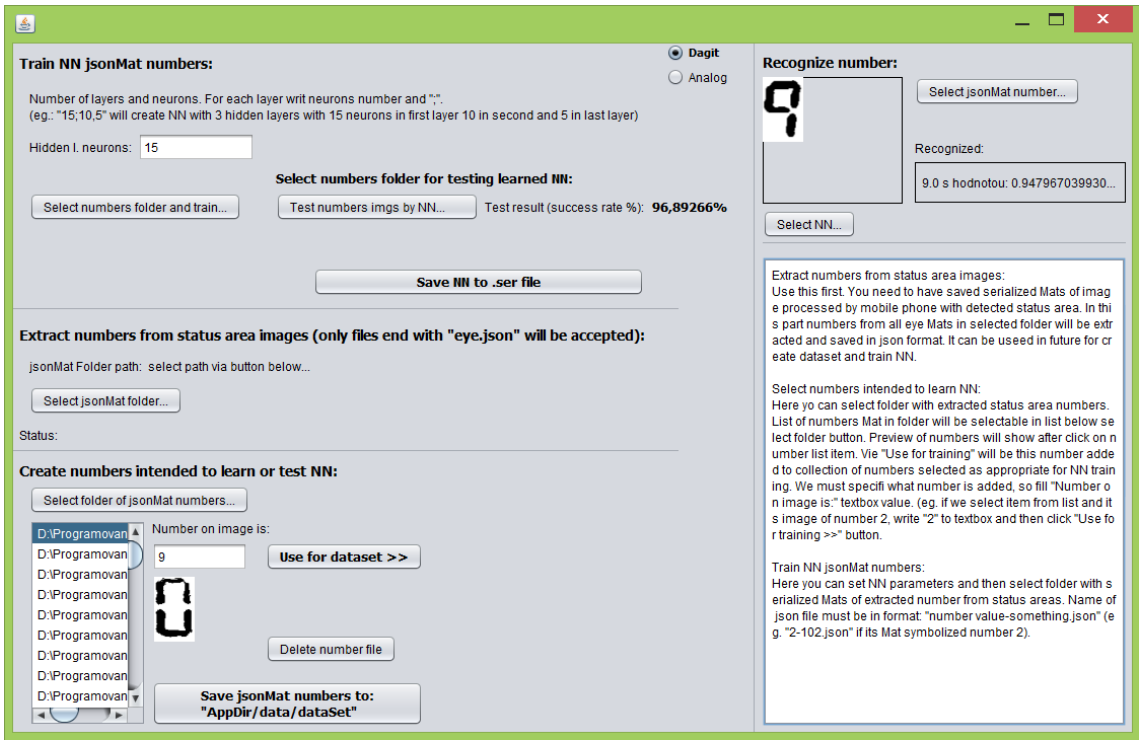
se kterou umí pracovat knihovna OpenCV. Byla tedy implementována třída `MatSerializerJson`, která umožňuje serializaci objektů typu `Mat` do textového souboru ve formátu json a jeho deserializaci, pro načtení aplikací.

Desktopová aplikace umožňuje s daty provádět následující věci:

- Načtení souboru s obrázky číslic do seznamu, zobrazení grafické podoby vybraného obrázku ze seznamu, zadání čísla, které je na obrázku zobrazeno a jeho následné uložení do složky pro pojmenované vybrané obrázky
- Načtení vybraných pojmenovaných obrázků a vytvoření tak trénovací množiny pro neuronovou síť
- Vytvoření testovací množiny ze sady obrázků ve složce a uložení pojmenovaných obrázků do složky s obrázky určenými pro vytvoření testovací množiny
- Sestavení neuronové sítě a její naučení trénovací množině
- Otestování naučené neuronové sítě na testovací množině vytvořené výběrem složky s obrázky, z nichž má být vytvořena testovací množina
- Možnost výběru obrázku, který je neuronovou sítí rozpoznán a výpis výstupu neuronu, ze kterého jsme určili, jaké číslo bylo rozpoznáno
- Serializaci neuronové sítě – její uložení do souboru pro možnost využití pro načtení při spuštění aplikace pro rozpoznávání elektroměrů

Popis ovládání a jmenných konvencí souborů je v anglickém jazyce napsán přímo v okně desktopové aplikace tak, aby ho měl uživatel vždy při ruce. Celým procesem tvorby trénovací a testovací množiny dat včetně učení a testování neuronové sítě je popsán v 5.1 Digitální elektroměry testování

Ukázka implementované desktopové aplikace je zobrazena na obrázku 15.



Obrázek 15 – ukázka implementace desktopové aplikace pro podporu práce s neuronovou sítí

4.12 Potřebná data a souborová struktura

Aby celá aplikace fungovala, je nutné dodržet určitá pravidla, co se datové struktury a jejího obsahu týče. Nejzásadnější věcí, je načtení klasifikačních modelů aplikací ihned po jejím spuštění, kdy je načten soubor, v němž je serializovaná naučená neuronová síť. Navíc pro každý typ elektroměru existuje jeden takovýto soubor. Konkrétně tedy musíme zajistit existenci souboru *nnDigit.ser* a *nnAnalog.ser*, což jsou právě soubory se serializovanými modely neuronových sítí.

Další dva soubory, které aplikace načítá po svém spuštění, jsou *svmDigit.ser* a *svmAnalog.ser*, které, jak už z jejich názvu vyplývá, slouží pro načtení modelu vektorového stroje. Přepnutí jak modelu neuronové sítě, tak modelu vektorového stroje je realizováno příslušným ovládacím prvkem v grafickém rozhraní aplikace, stejně tak jako přepnutí segmentačního modelu. Ve chvíli přepnutí dojde ke znovunačtení

serializovaných souborů klasifikačních modelů, a v tuto chvíli jsou tyto soubory opět využívány.

Z těchto faktu také vyplývá, jakým způsobem lze ovlivnit výsledky klasifikace a jak nahradit modely neuronových sítí a vektorových strojů. Stačí vytvořit a naučit neuronovou síť a vektorový stroj novým datům, tyto naučené modely uložit a nahrát do příslušného adresáře pod správným jménem.

V případě, že chceme provádět učení neuronové sítě přímo v mobilním zařízení, je třeba mít v adresářové struktuře složku *numbersForTraining*, která obsahuje data použitá pro vytvoření trénovací množiny. Pro učení neuronové sítě datům z této složky byla implementována metoda `initAI()` nacházející se v příslušných třídách připadajících jednotlivým typům elektroměrů (`DigitElProcessor` a `AnalogElProcessor`).

Kořenovou složkou pro všechna data aplikace je složka *ElektrometerOCR/data/*, která musí existovat v kořenovém adresáři úložného prostoru mobilního zařízení. Serializované klasifikační modely pak musí existovat přímo v této složce. Cesta k datové složce aplikace lze samozřejmě změnit. Je definována ve třídě `ElektrometerImageMatProcessor` v proměnné s názvem `appDir`.

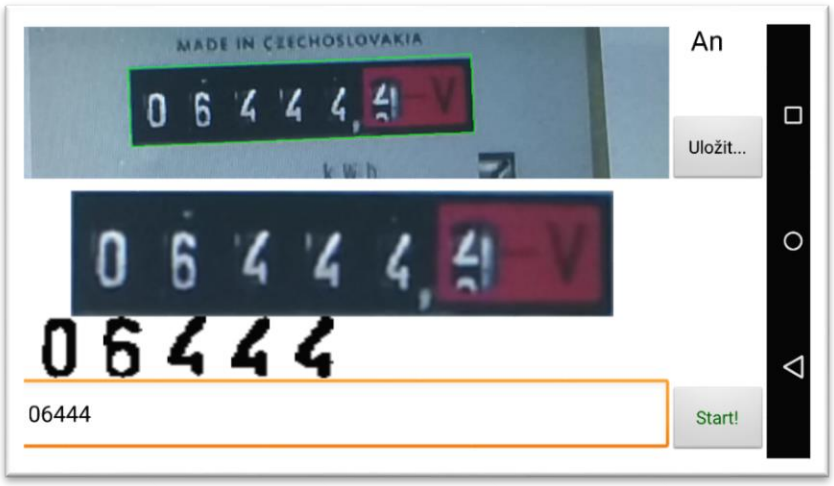
Souborová struktura využívaná desktopovou aplikací, má pak následující vlastnosti. Obsahuje složku *createdNumbers*, využívanou pro ukládání číslic získaných z číselníků elektroměrů, které byly vybrány pro separaci číslic. Dále složku *dataSet*, do které se ukládají pojmenované vytříděné obrázky číslic, jimž byla přiřazena informace o tom, jaká číslice se na obrázku vyskytuje. Aby takto vybrané číslice mohly být zpracovány neuronovou sítí jako testovací či trénovací data, je informace o tom, jaké číslo se na obrázku vyskytuje obsažena přímo v názvu ukládaného souboru čísla. Pro obrázek, představující číslici 1 je tedy název souboru 1-*xx*.json, kde *xx* může být libovolný řetězec. V našem případě je to číslo, vygenerované podle pořadí ukládání obrázku pojmenovaného čísla. Dále existuje složka *serNN*, do které jsou ukládány serializované modely neuronových sítí. Jejich název obsahuje časovou značku podle toho, kdy byl model sítě uložen, čímž je zabezpečeno, že se modely neuronových sítí nepřepisují při uložení dalšího modelu sítě. Soubory samotných modelů klasifikátorů (neuronové sítě

a SVM), jsou uloženy v kořenovém adresáři datové složky využívané aplikací. Jako kořenová složka je nastavena složka *data/*. Cestu ke kořenové složce lze opět změnit přepsáním hodnoty proměnné `appDir` ve třídě `ElektrometerImageMatProcessor`.

Zdrojové kódy včetně adresářových struktur jsou přiloženy v příloze 3.

4.13 Ukázky implementované mobilní aplikace

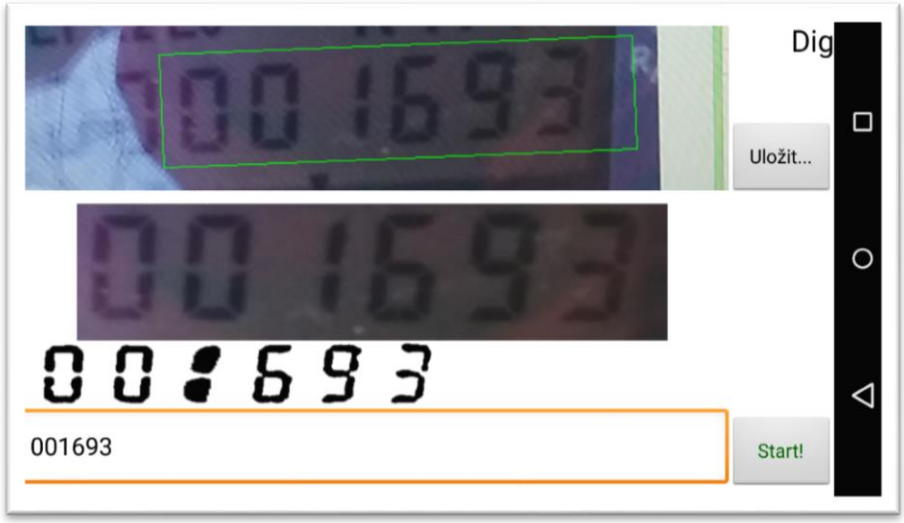
Na obrázku 16 můžeme vidět výsledek implementované aplikace při snímání analogového elektroměru.



Obrázek 16 – mobilní aplikace při snímání analogového elektroměru

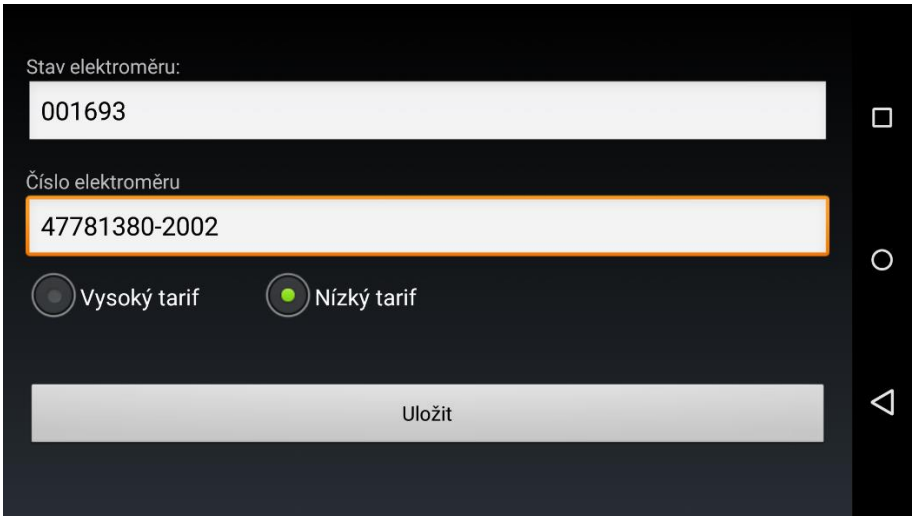
Obrázek demonstruje snímání elektroměru, přičemž bylo dosaženo detekce jeho číselníku a ten byl zeleně orámován. Detekovaný číselník byl separován a je uživateli zobrazen. Dále byla z číselníku separována čísla a opět zobrazena uživateli. Ve spodní části (oranžové rámování) je vidět výsledek rozpoznaného stavu elektroměru.

Podobná situace je znázorněna na obrázku 17, zde se ovšem jedná o digitální elektroměr.



Obrázek 17 – mobilní aplikace při snímání analogového elektroměru

Textový údaj o rozpoznaném stavu elektroměru lze editovat a ve chvíli, kdy chce uživatel přejít k dokončení procesu odečtu a uložit údaje, může kliknout na tlačítko „Uložit…“. Obrazovka aplikace pro uložení údajů odečtu je zobrazena na obrázku 18.



Obrázek 18 – obrazovka pro uložení údajů odečtu

5 Testování

Za pomoci desktopové aplikace vytvořeným i za účelem testování byla implementovaná aplikace otestována. Testování probíhalo tím způsobem, že proběhlo snímání dostupných fotografií elektroměrů a v průběhu procesu rozpoznávání byly ukládání grafické výstupy, konkrétně originální snímky snímané plochy, detekované číselníky na originálních snímcích a jednotlivá čísla, která byla separována z číselníků. Tato data byla přenesena z mobilního zařízení do počítače a následně za využití aplikace pro podporu práce s neuronovou sítí analyzována, zpracována pro možnost vytvoření trénovací a testovací množiny a následně otestována.

Proces testování je velice zdlouhavý, jelikož je třeba veškeré číslice separované z číselníků projít a přiřadit jim popisek, to znamená zadat, jaké číslo se na každém z obrázků vyskytuje.

Konkrétně byly z mobilního telefonu přeneseny složky *snimaniSaveAnalog* a *snimaniSaveDigit*, které obsahují uložené obrázky získané při ukládání výstupů aplikace na mobilním telefonu. Popíšeme si proces práce s daty pro účely trénování neuronové sítě na datech ze složky *snimaniSaveDigit*.

5.1 Digitální elektroměry

Složka *snimaniSaveDigit* obsahuje podsložky pojmenované číslem. Při ukládání na mobilním telefonu bylo stiskem tlačítka umožněno vytvořit novou složku, s číselným názvem o jedno vyšším, než před stiskem tlačítka, do které se výstupy ukládaly, což odpovídá právě názvům podsložek. V každé z těchto složek jsou uložena zmiňovaná data. Jejich názvy odpovídají obsahu dat, konkrétně pro první snímek ukládaného číselníku je jméno souboru 1_eye.jpg, což je obrázek originálního, neupraveného snímku. Ekvivalentem k tomuto obrázku je 1_eye.json, což je soubor se serializovanou maticovou reprezentací obrázku, který je uložen pro budoucí zpracování desktopovým programem. Další ukládaný obrázek je 1_area.jpg, což je obrázek číselníku, který byl v originálu vyříznut. Posledními ukládanými daty jsou obrázky a jejich .json ekvivalent. Jednotlivé obrázky číslic jsou uloženy pod jménem 1_num1.jpg až 1_num8.jpg

v případě, že z číselníku bylo separováno 8 číslic. Prefix 1_ značí, že uložená data náleží prvnímu uloženému snímku.

Pro vytvoření trénovací množiny vybereme v aplikace po kliku na tlačítko „*Select jsonMat folder*“ v sekci „*Extract numbers from status area images*“, kterou najdeme v prostřední části okna desktopové aplikace, složku *snimaniSaveDigit*, a v ní vybereme jednu z podsložek. Ve složky *createdNumbers* se nám tím vytvoří všechna čísla, které bylo možné z vybrané složky vytvořit a to tím způsobem, že jsou načteny všechny soubory končící názvem *eye.json*. Tyto soubory jsou načteny a zpracovány naprosto stejným způsobem, jako zpracovává mobilní aplikace jednotlivé snímky při snímání elektroměru, to znamená, že na konci procesu segmentace, po separaci číslic, dojde k uložení všech číslic ze všech číselníků do souboru *createdNumbers*. Dále pokračujeme kliknutím na tlačítko *„Select folder of jsonMat numbers…“*, kde po výběru složky, z níž chceme data (obrázky číslic) načíst (v našem případě naplněná složka *createdNumbers*) je vyplněn seznam v levé dolní části aplikace. Při kliku na načtenou položku v seznamu máme možnost zadat, jaké číslo se na obrázku vyskytuje, a tlačítkem „*Use for dataset >>“* přidat pojmenovaný obrázek čísla do seznamu určeného pro vytvoření trénovací, nebo testovací množiny. Projdeme tak všechna čísla a nakonec klikem na tlačítko „*Save jsonMat numbers*“, uložíme všechny pojmenované obrázky číslic do složky *dataSet*.

Nyní zbývá vytvořit neuronovou síť, a naučit ji trénovací množině. To provedeme výběrem složky *dataSet* (v případě, že aktuálně obsahuje data pro trénovací množinu) po kliku na tlačítko „*Select number folder and train“*. Vytvoříme tak model neuronové sítě, naučené námi zadaným datům.

S naučeným modelem neuronové sítě se nám otevírá možnost otestování výstupu sítě zvolením jedné konkrétní číslice k rozpoznání, což lze provést kliknutím na tlačítko „*Select jsonMat number“* a výběrem souboru obrázku čísla v .json formátu. Po volbě souboru se zobrazí grafická podoba obrázku a hodnota výstupu neuronu, který číslo klasifikoval.

Další možnost testování, která se nám nabízí, je otestování neuronové sítě na testovací množině, což je nejzásadnější možnost testování, kterou vlastně zjistíme výsledek

úspěšnosti celého systému. K vytvoření testovací množiny použijeme naprosto stejný postup, jakým byla vytvořena trénovací množina, ale vytvoříme sadu mnohem většího počtu testovacích dat. Kliknutím na tlačítko „*Test numbers imgs by NN...*“ zvolíme složku s testovací množinou, čímž je podle počtu chybně a správně klasifikovaných obrázků číslic vypočítána úspěšnost rozpoznání neuronovou sítí.

Pro vytvoření testovací množiny bylo použito zhruba 20 vzorků obrázků od každé číslice. Pro vytvoření testovací množiny byl použit následující počet vzorků pro každé číslo:

0: 149x | 1: 35x | 2: 22x | 3: 21x | 4: 17x | 5: 34x | 6: 30x | 7: 0x | 8: 17x | 9: 29x

Na dostupných fotografiích elektroměrů bohužel není obsažena číslice 7, což je důvodem, proč tato číslice nemohla být v trénovací množině obsažena, respektive ani jeden z digitálních elektroměrů, na kterém bylo možné provést detekci číselníku, číslici 7 neobsahuje. Počet číselníků, z jejichž snímků byla čísla pro učení sítě separována, byl 15 a snímání každého elektroměru trvalo cca 8 sekund. Počet snímků, z kterých byla data vybírána byl okolo 270.

Výsledek testu úspěšnosti neuronové sítě, určené pro rozpoznávání digitálních elektroměrů, naučené zmíněným trénovacím datům, byl na testovací množině **96,893 %**.

5.2 Analogové elektroměry

Vytvoření trénovací i testovací množiny probíhalo obdobně jako u digitálních elektroměrů. Postup a technika je naprosto stejná, pouze vycházíme z dat získaných při snímání analogových elektroměrů. Data trénovací množiny byly získány ze snímání cca 360 ti snímků elektroměrů, kterých bylo snímáno přibližně 30. Trénovací množina vytvořená ze snímaných dat obsahuje vzorky čísel, jejichž počet je následující:

0: 25x | 1: 11x | 2: 12x | 3: 16x | 4: 25x | 5: 11x | 6: 17x | 7: 16x | 8: 14x | 9: 18x

Testovací množina pak obsahuje 298 vzorků číslic analogových elektroměrů. Úspěšnost neuronová sítě, naučené zmíněným trénovacím datům, dosáhla na testovací množině hodnoty **68,463 %.**

5.3 Parametry neuronové sítě

Při obou testech měla neuronová síť stejnou struktura a byla naučena se stejnými zadanými parametry. Počet neuronů vstupní vrstvy je vždy 64, výstupní vrstvy pak 10. Testovaná neuronová síť pak měla jednu skrytou vrstvu, ve které bylo 15 neuronů. Náhodná inicializace vah byla nastavena v rozsahu -0.3 až 0.3. Hodnota parametru eta byla 0.01.

5.4 Test na reálném elektroměru

Aplikace byla otestována i na jednom analogovém elektroměru. Vzhlede tohoto elektroměru byl naprosto odlišný, než pro jaký byla aplikace připravena a jaký implementuje segmentační modu. Nicméně i přes tento fakt se podařilo po chvíli snímání elektroměr sejmout. Při nalezení vhodného úhlu snímání a pozice, na kterou potřebujeme číselník elektroměru ve snímaném obraze dostat, byly jeho číslice úspěšné separovány.

Na obrázku 19 je zobrazena fotografie snímané plochy elektroměru, na kterém bylo reálné snímání provedeno. Na první pohled můžeme jasně vidět, že jeho vzhled se zásadně liší od vzhledu analogového elektroměru, pro který byl systém navržen. Při bližším pohledu na obrázek můžeme vidět zelený obdélník okolo číselníku elektroměru, což demonstruje úspěšnost detekce číselníku na reálném analogovém elektroměru.

Obrázek 19 – Ukázka reálného elektroměru

Při procesu rozpoznávání bylo aktivováno ukládání výstupů a tak nasbírána reálná data pro analýzu. Z nasbíraných dat byla vytvořena testovací množina a to z každého snímku s detekovaným číselníkem. Konkrétně bylo použito 21 snímků číselníku, přičemž nám z těchto snímků vzniklo 126 obrázků číslic. Na této testovací množině byla otestována neuronová síť naučená při předchozím testování. Její úspěšnost rozpoznání byla pouhých **25,397 %.**

Obrázek 20 zobrazuje číslice, které byly z číselníku reálného elektroměru separovány.

Obrázek 20- číslice separované z reálného elektroměru

Takto nízká naměřená hodnota vedla k myšlence testu změny topologie neuronové sítě. Počet neuronů ve skryté vrstvě byl tedy upraven z 15 na 30. Neuronová síť byla naučena na stejné trénovací množině jako při prvním testu na analogových elektroměrech, a opět otestována na stejné testovací množině reálného elektroměru, jako síť s předchozí topologií. Výsledek úspěšnosti sítě se až překvapivě navýšil. Neuronová síť se třiceti neurony ve skryté vrstvě dosáhla s nezměněnými ostatními parametry úspěšnosti **86,508 %.** Tato skutečnost vedla k provedení opětovných testů s naprosto stejnými daty jako v předchozích testech avšak s nově nastaveným počtem neuronů ve skryté vrstvě.

5.5 Nejlepší dosažené výsledky

Při opětovném testování digitálních, analogových i jednoho reálného analogového elektroměru s využitím stejných trénovacích i testovacích množin jako při prvních testech, ovšem s upravenou topologií sítě, bylo naměřeno následujících hodnot.

U digitální elektroměrů dosáhla hodnota úspěšnosti rozpoznání sítí **98,588 %.** U analogových elektroměrů pak hodnota stoupla na **95,973 %**. Výsledek posledního testu na reálném analogovém elektroměru byl již zmíněných **86,508 %.**

Počet neuronů ve skryté vrstvě byl nadále modifikován a byly prováděny opětovné testy. Se zvyšujícím se počtem neuronů (víc než 30). Docházelo zprvu k drobným zvyšujícím se úspěšnostem rozpoznání v řádek desetin až jednotek procent. Od zhruba čtyřiceti neuronů výše se však úspěšnost téměř nezvětšovala, spíše začala klesat. Stejně tak při použití menšího počtu neuronů se výsledky začaly zhoršovat.

Finální model neuronové sítě, výchozí model neuronové sítě, nahraný v datové složce aplikace pro mobilní telefon má ve skryté vrstvě tedy právě 30 neuronů.

Na mobilním telefonu muselo být aktivováno učení sítě, jelikož serializovaný soubor s modelem neuronové sítě, jako jediný, není přenositelný mezi platformami.

Data použitá pro vytvoření trénovací a testovací množiny jsou přiložena v příloze 4.

6 Závěr

Úspěšnost provedení odečtu elektroměru s využitím implementovaného systému je závislá na kvalitě obrazu snímaných elektroměrů a na použitých klasifikačních modelech. V případě špatného vizuálního obrazu získaného při snímání elektroměru fotoaparátem mobilního telefonu, nemusíme být schopni rozpoznání provést. Největším problémem jsou odlesky, které se kvůli vlastnostem materiálů, ze kterých jsou tvořeny skla a plasty, objevující se před číselníky elektroměrů, mohou objevovat. Tento problém se však vyskytoval na dostupných fotografiích elektroměrů převážně z důvodu nevhodně pořízené fotografie, odlesku blesku fotoaparátu a podobně. Při reálném použití jsme schopni odlesky do značné míry eliminovat, stejně jako problémy se snímáním elektroměru v nevhodném úhlu, nebo z nevhodné vzdálenosti.

Z provedených testů je viditelné, jaká je funkčnost a úspěšnost aplikace s použitím výchozích klasifikačních modelů. Lze z nich ale také vyvodit, že co se týče klasifikace číslic, je tato úspěšnost závislá na použitých trénovacích množinách, které je snadné vytvářet a pomocí nich vytvářet i nové modely neuronových sítí. Pro nasazení aplikace pro reálné účely by bylo tedy vhodné nejprve nasbírat dostatek dat, které po vhodném vytřídění a povedou k co nejlepším výsledkům implementovaného systému v případě

reálného použití. Můžeme tak lehce aplikaci přizpůsobit konkrétním potřebám. Pro podporu možnosti odečtu dalších potřebných typů elektroměrů stačí přidat další segmentační moduly a pro ně vhodné klasifikační modely. I díky vytvořené desktopové aplikaci jsou tato rozšíření a přizpůsobení pro konkrétní reálné potřeby relativně jednoduchou úlohou.

Výsledná implementace celého systému tedy přináší nejen možnost využití jednoho navrženého řešení problematiky, ale tvoří spíše sadu nástrojů, které jsou využitelné pro řešení problematiky podobného typu, jakou se tato práce zabývá.

6.1 Návrhy na zlepšení

Při vývoji aplikace bylo zjištěno množství způsobů, které by mohlo vést ke zlepšení celého systému. Další návrhy na zlepšení vycházejí z prostudovaných materiálů a výsledků již provedených výzkumů.

Při implementaci segmentačních algoritmů se došlo ke zjištění, že některé z jejich parametrů by bylo vhodné nastavovat v závislosti na vzdálenosti snímače od snímaného elektroměru. Konkrétně například při detekci číslic, kdybychom znali velikost číslice v obraze (což jde při znalosti vzdálenosti snímače vypočítat), mohli bychom segmentaci provádět výrazně efektivněji. Pro toto by bylo vhodné využívat zařízení, které umožňuje zjištění vzdálenosti od určitého snímaného objektu (například díky použití dvou snímačů).

Dalšího možného zlepšení by mohlo být dosaženo aplikováním klasifikace pomocí SVM na čísla číselníku elektroměru. Můžeme SVM použít stejně jako na číselníky – třídit, zda bylo detekování číslo, či nikoli. Můžeme ale i sestavit více SVM, které by mohly být využity pro rozpoznání číslice. Pro každé číslo můžeme vytvořit jeden vektorový stroj a obrázek číslice předkládat každému z nich, což může také vést k úspěšnému rozpoznání.

Při využití jiné techniky získávání atributů, které slouží jako vstupy do neuronové sítě, z obrázků číslic, budeme dosahovat jiných výsledků úspěšnosti rozpoznání i při použití aktuálně implementované sítě. Proto změna této techniky může vést k větší efektivitě a lepším výsledkům celého systému.

Stejně tak můžeme vyměnit celý rozpoznávací modul a místo jedné perceptronové neuronové sítě použít například konvoluční neuronové, které jsou v současné době považovány za průlomové. Důkazem toho může být nedávné (rok 2015) vydání DeepBelief frameworku [22], který implementuje právě neuronové sítě a metody deep learningu a úspěšnost rozpoznávání za pomoci technik tohoto frameworku dosahuje vysokých výsledků.

Inspirace na zlepšení rozpoznávacího modulu a volbu typu neuronové sítě můžeme nalézt na webových stránkách zabývajících se rozpoznáváním ručně psaného písma, konkrétně na databázi MNIST [23]. Na webu je v tabulce uvedeno, jaký rozpoznávací modul byl pro rozpoznání číslic použit, jakého bylo dosaženo úspěchu rozpoznávání a odkaz na materiály zabývající se konkrétním řešením.

Co se segmentace týče, existuje algoritmus SWT (stroke width transform), jehož využití bylo testováno pro detekci textu v obraze [14]. Využití tohoto algoritmu by mohlo vést ke zlepšení procesu segmentace.

Další úpravy, které by zlepšily použitelnost systému, se týkají spíše usnadnění práce uživateli. Rozšíření systému o část, která by zajišťovala získávání seznamu elektroměrů určených k odečtu, a uživatelsky přívětivá možnost výběru elektroměru, na kterém se odečet provádí, by vedla ke značnému zlepšení, co se interakce s uživatelem a zefektivnění provádění odečtu týče.

Zásadní věc, kterou by bylo vhodné implementovat, je serializace neuronové sítě ve formátu, který je platformně přenositelný (např. json, xml). Možnost vytvoření modelu neuronové sítě na desktopovém zařízení je velice důležitá hlavně z důvodu náročnosti výpočetního výkonu. Některý model sítě se může učit předložená data velice dlouhou dobu, někdy i v řádu hodin. Možnost přenosu takového modelu sítě a jeho využití v mobilní aplikaci by byla velmi vhodná.

Možností a způsobů jakými lze systém vylepšit je spousta. V současnosti je tato problematika aktuální téma a neexistuje jednoznačné univerzální řešení, kterým lze problematiku řešit.

6.2 Umožnění dalšího vývoje

Pro umožnění dalšího vývoje většímu množství lidí byla spuštěna webová stránka www.opencvcharrecognition.com, na které jsou dostupné zdrojové kódy implementované aplikace. Tyto kódy jsou k dispozici v git repositáři, jehož adresa je na webu k dispozici. Web bude v budoucnu upravován a přizpůsobován dle potřeby. Zamýšlený plán je informovat komunitu o existenci implementovaného systému a dát jí možnost podílet se na jejím rozvíjení. Díky využití knihovny OpenCV, která má komunitu o velkém množství uživatelů, je šance, že se najdou lidé, kteří budou mít zájem o testování a navrhování různých technik, které jsou v systému využitelné.

Navíc je v plánu umožnění nahrávání nasbíraných dat, pro účely dalšího testování a zlepšování klasifikačních modelů. Uživatelé webových stránek by tak mohli tato data postupně rozšiřovat.

Na webových stránkách je plánována i možnost vytvoření modelu neuronové sítě online s možností otestování a případné uložení serializovaného modelu, což je zajímavá možnost demonstrování využití neuronové sítě.

Popis celého systému, případně jednotlivých algoritmů použitých při implementaci, je vhodný i pro studijní účely ostatních lidí, kteří budou vystaveni řešení podobných problémů, jako řeší implementovaná aplikace.

Zdroje

[1] Leitkep, Z.,2012: Rozpoznávání naměřených hodnot. [Recognition measured values. Bc. Thesis, in Czech.] – 30 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

[2] Hanzlík Ondrej, 2011: Systém pro rozpoznávání tišteného písma [Character recognition system. Bc. Thesis, in Czech.] – 30 p., Faculty of Science, University of South Bohemia, České Budejovice, Czech Republic.

[3] Adamec, 2011: Zpracování a rozpoznání obrazu, Přírodovědecká fakulta, Univerzita palackého, Olomouc, Czech Republic.

[4] Comparison of the Open CV Feature detection algorithms. [online]. Dostupné z: http://computer-vision-talks.com/articles/2011-01-04-comparison-of-the-opencv-feature-detection-algorithms/

[5] Předzpracování obrazu pro neuronovou síť. ČVUT Praha, 2000. Diplomová práce. ČVUT Praha.

[6] Basic Structures. OpenCV doc [online]. [cit. 2015-04-23]. Dostupné z: http://docs.opencv.org/modules/core/doc/basic_structures.html

[7] Managing the Activity Lifecycle. Android doc [online]. [cit. 2015-04-23]. Dostupné z: http://developer.android.com/training/basics/activity-lifecycle/index.html

[8] Camera. Android doc [online]. [cit. 2015-04-23]. Dostupné z: http://developer.android.com/guide/topics/media/camera.html

[9] Opencv documentation!. OpenCV doc [online]. [cit. 2015-04-23]. Dostupné z: http://docs.opencv.org/

[10] Real-Time Video Processing Algorithm for Instant License Plate Recognition in iOS Apps (using OpenCV & GPGPU). Azoft [online]. [cit. 2015-04-23]. Dostupné z: http://rnd.azoft.com/instant-license-plate-recognition-in-ios-apps/

[11] Support Vector Machines. OpenCV doc [online]. [cit. 2015-04-23]. Dostupné z: http://docs.opencv.org/modules/ml/doc/support_vector_machines.html

[12] Support vector machines. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-23]. Dostupné z: http://cs.wikipedia.org/wiki/Support_vector_machines

[13] Cannyho hranový detektor. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-23]. Dostupné z: http://cs.wikipedia.org/wiki/Cannyho_hranov%C3%BD_detektor

[14] Scene Text Localization and Recognition with Oriented Stroke Detection [online]. [cit. 2015-04-23]. Dostupné z: http://cmp.felk.cvut.cz/~neumalu1/neumann-iccv2013.pdf

[15] SKRBEK, Miroslav. Neuronove site a AI.

[16] Multilayer perceptron. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-23]. Dostupné z: http://en.wikipedia.org/wiki/Multilayer_perceptron

[17] Perceptron. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-04-23]. Dostupné z: http://cs.wikipedia.org/wiki/Perceptron

[19] Obrazové segmentační techniky, FAKULTA INFORMAČNÍCH TECHNOLOGIÍ: Přehled existujících metod [online]. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, 12.10.2005 [cit. 2015-04-23]. Dostupné z: http://www.fit.vutbr.cz/~spanel/segmentace/

[20] ADAMEC, Václav. Zpracování a rozpoznávání obrazu. PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO KATEDRA INFORMATIKY, 2011.

[21] VALA, Tomáš. Rozpoznávání SPZ z jednoho. Univerzita Karlova v Praze Matematicko-fyzikální fakulta, 2006.

[22] The SDK for Jetpac's iOS Deep Belief image recognition framework. GitHub [online]. [cit. 2015-04-23]. Dostupné z: https://github.com/jetpacapp/DeepBeliefSDK

[23] THE MNIST DATABASE of handwritten digits. [online]. [cit. 2015-04-23]. Dostupné z: http://yann.lecun.com/exdb/mnist/

Přílohy

Příloha 1. Fotografie elektroměrů

Příloha 2. Textový soubor s návodem na nastavení vývojového prostředí

Příloha 3. Zdrojové kódy aplikace a potřebná data

Příloha 4: Data použitá při testování