Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Segmentácia sémanticky významných oblastí v obraze

Diplomová práca

2019
Bc. Lukáš Mayer

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Segmentácia sémanticky významných oblastí v obraze

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	prof. Ing. Igor Farkaš, Dr.

Bratislava, 2019
Bc. Lukáš Mayer

47797630

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Lukáš Mayer
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Segmentácia sémanticky významných oblastí v obraze
Segmentation of semantically relevant image areas

Anotácia: Kontext obrazu v reálnych snímkach býva väčšinou obsahovo veľmi pestrý, preto jedným z mnohých problémov pri rozpoznávaní znakov v obraze, ktoré sa dnes rieši pomocou neurónových sietí, je definovanie znaku alebo reťazca znakov na obraze, ktoré nás zaujímajú. Vysegmentovať oblasť na obrázku, ktorá obsahuje práve želané znaky sa dá viacerými prístupmi. Sľubným prístupom je použitie neurónovej siete.

Cieľ: 1. Preskúmať aktuálny stav výskumu v oblasti segmentácie významných oblastí.
2. Pokúsiť sa vylepšiť existujúce segmentačné metódy, a to v konkrétnej úlohe obrázkov vodomerov, na ktorých treba detekovať číslice popisujúce stav vodomeru.

Literatúra: Shapiro L. G., Stockman G. C. (2001). Computer Vision, pp. 279-325, New Jersey, Prentice-Hall.
Liu X. et al. (2014), Detection and segmentation text from natural scene images based on graph model. WSEAS Transactions On Signal Processing.
Lowe D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints.International Journal of Computer Vision.

Vedúci: prof. Ing. Igor Farkaš, Dr.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 26.11.2018

Dátum schválenia: 26.02.2019 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie:

Chcel by som vyjadriť vďačnosť môjmu školiteľovi prof. Ing. Igorovi Farkašovi, Dr. za ochotu a všetku pomoc pri vedení tejto diplomovej práce. Taktiež by som sa chcel poďakovať spoločnosti Cognexa za poskytnutie datasetu a viaceré konzultácie a v neposlednom rade ďakujem všetkým svojim blízkym za obrovskú podporu a pochopenie počas celej doby môjho štúdia.

Abstrakt

Vývoj v oblasti počítačového videnia za posledné roky dosahuje pozoruhodné výsledky, ktoré umožňujú väčšie možnosti využívania počítačov v každodennom živote. V tejto práci sme sa zamerali na problém sémantickej segmentácie obrazu, ktorého cieľom je roztriediť pixely obrazu do oblastí na základe kontextu zobrazovanej scény.
Našou úlohou bolo preskúmať rôzne prístupy segmentácie, za účelom nájsť optimálny algoritmus pre segmentovanie ciferníka vodomeru zachytenom na prirodzenom obrázku. Prirodzené obrázky disponujú množstvom nedostatkov ako na príklad rozdielny jas, kontrast, rotácia, škálovanie, atď. V diplomovej práci sme odprezentovali výsledky algoritmov využívajúcich hlboké neurónové siete, ktoré sa ukázali byť schopnejšie vysporiadať sa s variabilnými parametrami prirodzených obrázkov na rozdiel od testovaných prístupov, ktoré ciferník vodomeru segmentovali bez využitia hlbokého učenia.

Kľúčové slová: sémantická segmentácia, detekcia objektov na obraze, template matching, neurónové siete, konvolučný autoenkóder, detekcia textu na obraze

Abstract

The developments in computer vision have made remarkable results in recent years, allowing more opportunities to utilize computers in everyday life. In our thesis we have focused on the problem of semantic segmentation, which aims to sort the pixels of an image into regions based on the context of the displayed scene. Our aim was to explore different segmentation approaches to find the optimal algorithm for segmenting the face of a water meter captured in a natural image. These have a number of shortcomings such as different brightness, contrast, rotation, scaling, etc. In our thesis we have presented the results of algorithms using deep neural networks, which proved to be more efficient in dealing with the variable parameters of natural images as opposed to the tested approaches that segmented the water meter without using deep learning.

Keywords: semantic segmentation, image object detection, template matching, neural networks, convolutional autoencoder, image text detection

Obsah

1 Úvod **1**
1.1 Odčítanie stavu na vodomeri 2

2 Aktuálny prehľad prác **3**
2.1 Lokalizácia objektu v obraze bez využitia hlbokých neurónových sietí . 4
2.1.1 Binárna segmentácia obrazu 4
2.1.2 Segmentácia v rámci ANPR systémov 5
2.2 Lokalizácia objektu v obraze s využitím hlbokých neurónových sietí . . 11
2.2.1 YOLO 13
2.2.2 Sémantická segmentácia obrázka na úrovni pixelov 13

3 Použité technológie a metódy **16**
3.1 Predspracovanie obrazu 16
3.2 Binárne spracovanie obrazu 17
3.2.1 Prahovanie 17
3.2.2 Konvolúcia 18
3.2.3 Cannyho hranový detektor 19
3.2.4 Binárne morfologické transformácie 22
3.2.5 Šedotónové morfologické transformácie 23
3.3 Metódy hľadania zhody 25
3.4 Algoritmy párovania príznakov 26
3.5 Viacvrstvové neurónové siete 28
3.6 Konvolučné neurónové siete 33
3.7 Autoenkódery 35

4 Testované algoritmy a dataset **37**
4.1 Dataset 37
4.1.1 Augmentácie 39
4.2 Morfologická segmentácia 39
4.3 Hľadanie zhody v obraze 41
4.4 Predikcia rohových súradníc 43

4.5 Predikcia masky 45
4.6 Detekcia textu v obraze 47
4.6.1 EAST model 48

5 Vyhodnotenie **50**
5.1 Metrika 50
5.2 Úspešnosť metód bez využitia hlbokého učenia 51
5.3 Úspešnosť metód s využitím hlbokého učenia 55

6 Záver **59**

Dodatok **67**

Zoznam obrázkov

2.1 Obrázok s histogramom po aplikovaní vertikálnej projekcie [9] 7
2.2 Graf histogramu vertikálnej projekcie [9]. 8
2.3 Graf vývoja úspešnosti segmentačných algoritmov na datasete ImageNet [61] od roku 2010 [12]. 12
2.4 Architektúra konvolučnej neurónovej siete zloženej zo SE ("Squeeze and Excitation") blokov [16]. 12
2.5 YOLO rozdelí obrázok na mriežku s rozmermi N x N, pričom pre každý dielik odpredikuje B rámikov ohraničenia objektu s pravdepodobnosťami ich správnosti a s pravdepodobnosťami príslušnosti daného objektu do jednotlivých tried. [20]. 14
2.6 Architektúra konvolučného enkódera a dekódera [22]. 15
2.7 Pamätanie si indexov maximálnych hodnôt vo fáze Poolingu (vľavo) a použitie indexov v rámci Unpooligu v dekóderi (vpravo) [23]. 15

3.1 Konvolučná maska (matica 3×3 v strede) deteguje zvislú hranu na vstupnom obrázku (matica vľavo) [28]. 19
3.2 Sobelove filtre pre detekciu horizontálnych a vertikálnych hrán 20
3.3 Vyhodnocovanie bodu X_1 na súradniciach $V(i, j)$ v rámci hľadania lokálnych maxím [29]. 21
3.4 Ukážka princípu dvojitého prahovania obrázku [30]. 22
3.5 Povrch funkcie reprezentujúci maximálne hodnoty bodov množiny A [34]. 24
3.6 Matice referenčného a vzorového obrázka. 26
3.7 Matice s výslednými hodnotami korelácie. 26
3.8 Generovanie SIFT deskriptoru z lokálnych gradientov do mriežky s rozmermi 2×2 [37] 28
3.9 Jednoduchý perceptrón má n binárnych vstupov a jeden výstup. [44] . 29
3.10 Ukážka úlohy prahovej hodnoty počas učenia perceptrónu [45]. 30
3.11 Ukážka typov aktivačných funkcií [62]. 31
3.12 Vývoj chyby pomocou optimalizačného algoritmu s klesaním podľa gradientu. [46] 32

3.13 Architektúra konvolučnej neurónovej siete. 33
3.14 Efekt aplikovania dropout vrstvy v sieti. [64] 35
3.15 Architektúra Autoenkódera. [52] 36

4.1 Ukážka segmentovaných objektov na prirodzenom obrázku [60]. 38
4.2 Ukážka obrázkov z datasetu vodomerov s dobre identifikovateľným ciferníkom. 38
4.3 Ukážka snímkov z datasetu vodomerov so slabšie čitateľným stavom. . 39
4.4 Porovnanie výsledkov algoritmu morfologickej detekcie na obrázkoch s vozidlami a obrázkoch s vodomermi. 41
4.5 Variant vzorového obrázka so samostatnými znakmi. 42
4.6 Variant vzorového obrázka s kompletným ciferníkom vodomeru. 43
4.7 Predikovanie binárnej masky ciferníka vodomeru. 46
4.8 Výsledky detekcie textu pri prahoch 50% , 80%, 90%. 48
4.9 Architektúra plne konvolučnej neurónovej siete na predikciu textu. [57] 49

5.1 Ukážka výsledkov segmentácie s využitím morfologických operácii. Zelené orámovanie na obrázkoch predstavuje skutočnú anotáciu obrázkov a žlté orámovanie je výsledok morfologickej segmentácie. 52
5.2 Porovnanie niekoľkých výsledkov testu RSTM(Rotate-Scale Template Matching) algoritmu a jeho kontrolného ekvivalentu. Zelené orámovanie na obrázkoch predstavuje skutočnú anotáciu obrázkov a žlté orámovanie je výsledok daných algoritmov. 54
5.3 Ukážka výsledkov predikcie súradníc rohov ciferníka pomocou viacvrstvovej doprednej neurónovej siete. Zelené orámovanie na obrázkoch predstavuje skutočnú anotáciu obrázkov a žlté orámovanie je výsledok predikcie siete. 55
5.4 Ukážka výsledkov segmentácie pomocou konvolučného autoenkódera. . 56
5.5 Ukážka výsledkov metódy detekcie textu na obraze. Zelené orámovania predstavujú detegovaný text pomocou modelu EAST. Červené orámovanie je najväčšie detegované textové pole (druhá heuristika) a žlté orámovanie je najmenšie pokrytie detegovaných prámovaní (prvá heuristika). 57

Kapitola 1

Úvod

V tejto diplomovej práci sa budeme zaoberať spracovaním prirodzených obrázkov. Konkrétne sa budeme venovať hľadaniu a detegovaniu oblastí na obrázkoch, ktoré sú pre nás sémanticky významné. Pod prirodzeným obrázkom rozumieme ľubovolnú fotografickú snímku, zhotovenú v bežných podmienkách okolitého sveta. Takto zhotovené fotografie podliehajú mnohým faktorom ako napríklad slabý alebo príliš silný jas, kontrast fotografie, ostrosť, perspektíva, z ktorej je fotografia zhotovená a mnohé ďalšie. Toto všetko má dopad na ďalšie spracovanie obrazu.

V dnešnej dobe sa s technológiami, ktoré spracúvajú obraz pre rôzne komerčné alebo vedecké účely, stretneme na každom kroku. Napríklad parkovacie domy zvyknú byť vybavené systémom na rozpoznávanie evidenčného čísla vozidiel (ďalej len EČV), v luxusnejších vozidlách sa už bežne vyskytuje funkcia automatického čítania dopravných značiek, ktorá napomáha šoférom v cestnej premávke prostredníctvom detekcie dopravných značiek z videozáznamu prednej kamery. Aby bola táto funkcia použiteľná v automobilovom priemysle, musí fungovať veľmi presne a hlavne rýchlo, aby vyprodukovaná informácia bola pre šoféra použiteľná. Čím ďalej, tým viac sa do praxe zavádza detegovanie a rozpoznávanie ľudských tvárí. Niektoré telefóny dokonca začínajú využívať tzv. face ID na autentifikáciu osôb, čo prirodzene kladie vysoké nároky na výkon aj chybovosť danej technológie. Podobných príkladov z praxe je však omnoho viac. Sémanticky významnou oblasťou pre spomenuté prípady je zakaždým niečo iné. Pre telefón sú to črty ľudskej tváre (nos, ústa, oči, atď...), pre garážový systém zase tabuľka s EČV vozidla. Pre detekciu tabuľky s EČV existuje niekoľko známych algoritmov s rôznym prístupom, ktoré sú schopné zo snímku detegovať EČV, ak sa na ňom nachádza. Čo však robiť v prípade, že na obrázku potrebujem detegovať niečo iné a zároveň menej jednoznačné? Čo robiť, ak potrebujeme aby bol počítač schopný takýto objekt detegovať spoľahlivo a zároveň rýchlo?

1.1 Odčítanie stavu na vodomeri

V našej diplomovej práci sa zameriame na konkrétny problém, týkajúci sa automatizácie odčítavania stavu vodomerov z fotografie. Mnohé vodomery sa nachádzajú na veľmi ťažko prístupných miestach a odčítanie ich stavu býva nielen náročné, ale aj nebezpečné. Nebezpečie vzniká v šachtách, v ktorých sa okrem vodomerov nachádzajú merače iných inžinierskych sietí. Obzvlášť pri poruche plynovej prípojky alebo potrubia vzniká po vstupe do šachty pre vodomeračov život ohrozujúci stav. Z tohto dôvodu, ale aj z dôvodu zrýchlenia celého procesu, vzišla iniciatíva čiastočne automatizovať proces odčítavania stavu vodomerov. Automatizácia procesu, akým je odpočet stavu vodomerov, spôsobom odfotenia vodomeru a následného spracovania tohto snímku predstavuje zaujímavý, ale pritom netriviálny informatický problém. Odpoveď a riešenie našich požiadaviek by nám mohla poskytnúť oblasť počítačového videnia, ktorej riešenia sa stále viac stávajú praktickou súčasťou každodenného života človeka. Veľa problémov, ktoré táto oblasť skúma, rieši nasledujúce úlohy: klasifikácia objektov, detekcia objektov, segmentácia objektov. Proces rozpoznávania textu a znakov na prirodzenom obrázku sa obvykle skladá z dvoch krokov: Detekcia textu s jeho následnou segmentáciou je prvý krok. Potom pomocou OCR (Optical Character Recognition) je rozpoznaný text na segmente. Problém rozpoznávania, resp. čítania číslic z obrázka, nie je ničím novým a už v 90-tych rokoch bol dobre známy. Dôkazom toho je známy dataset MNIST, ktorý obsahuje 70000 obrázkov rukou písaných číslic s rozmerom 20x20 px. normalizovaných do okienok veľkosti 28x28 px. Vtedy bola motiváciou automatizácia čítania PSČ čísel na amerických poštách [1]. V dnešnej dobe už existujú riešenia, ktoré nad MNISTom dosahujú výborné výsledky s chybovosťou pod 0,2% [2]. Ak by sme sa zamerali na riešenie našej úlohy iba spôsobom detegovania znakov na obrázku, komplikácie by spôsobil fakt, že obrázky s vodomermi môžu okrem číselníka vodomeru obsahovať aj rôzne ďalšie znaky s iným kontextom. Aby sme teda mohli z obrázka čítať číslice na vodomeri rýchlo a presne, potrebujeme najprv vysegmentovať z obrázka displej s číslicami vodomeru. Tieto číslice už konvolučná neurónová sieť prečíta bez väčších komplikácií. Ako rýchlo dokážeme na obrázku detegovať displej vodomeru? A ako presné výsledky dosiahneme? Ktorá technológia či prístup budú najvhodnejšie? Presne toto sú otázky, na ktoré sa v tejto práci pokúsime nájsť odpoveď.

Kapitola 2

Aktuálny prehľad prác

V tejto kapitole si priblížime viaceré témy, ktoré sa nejakým spôsobom dotýkajú nášho zadania. Ukážeme si aktuálne články, prezentujúce moderné prístupy a technológie spracovania obrazu, ale aj problémy, ktoré sa riešili už pri začiatkoch podobnej problematiky. Ako sme už uviedli v predošlej kapitole, medzi niekoľko známych problémov, ktorými sa oblasť počítačového videnia zaoberá, patrí:

- Klasifikácia objektov - algoritmus na obrázku deteguje objekty a určí, do ktorej z vopred definovaných sémantických kategórií patria.
- Detekcia objektov - metóda veľmi podobná rozpoznávaniu objektov s tým rozdielom, že algoritmus má za úlohu objekty na obrázku nielen rozpoznať, ale ich aj na obrázku vyznačiť (najčastejšie bounding boxom).
- Segmentácia obrazu - rozdelí obraz na niekolko súvislých častí, pričom nerozlišuje kontext obrázku ani význam jednotlivých častí.
- Špeciálnym typom segmentácie obrazu je sémantická segmentácia. Tá rozdeľuje obraz do viacerých, taktiež súvislých oblastí, ktoré reprezentujú niektorú z vopred definovaných tried objektov. Takáto segmentácia sa dá implementovať aj na úrovni klasifikácie jednotlivých pixelov obrázka, miesto klasifikácie celej oblasti obrázka.

  Viac sa o problematike spracovania obrazu môžeme dočítať v knihe "Computer Vision" [3].

2.1 Lokalizácia objektu v obraze bez využitia hlbokých neurónových sietí

Potreba spracovávať digitálny obraz a informáciu, ktorú nesie, tu bola ešte pred príchodom neurónových sietí do tejto oblasti. Napriek tomu, že v dnešnej dobe sú v oblasti spracovania obrazu hlboké neurónové siete najpopulárnejšou voľbou, existuje viacero spôsobov ako segmentovať objekty nachádzajuce sa na obraze aj bez ich použitia (Binárna segmentácia, Template matching, Prahovanie a ďalšie). My si teraz priblížime viaceré prístupy, ktoré v tejto oblasti boli publikované.

2.1.1 Binárna segmentácia obrazu

Najmä do roku 2000 sa digitálne spracovanie obrazu venovalo predovšetkým metódam segmentácie ako: prahovanie, hranová segmentácia, či segmentácia založená na oblastiach obrazu [4]. Jeden z prvých článkov týkajúci sa segmentácie digitálneho obrazu pomocou počítača, sa objavuje už v 60-tych rokoch minulého storočia, keď Lawrence Roberts predstavil hranový detektor, ktorý je podľa neho aj pomenovaný [5].

Detekcia hrán na obrázku

Jedným zo základných spôsobov predspracovania obrazu je využívanie konvolučných filtrov. Každú transformáciu obrázka alebo ľubovolný filter nad obrázkom vieme definovať nejakou konvolučnou maticou. V princípe platí, že na detekciu hrán na obrázku sa používa opakované aplikovanie nejakej z konvolučných matíc. Typ hranovej detekcie závisí od použitého operátora:

- Robertsov operátor [5]
- Horizontálna a vertikálna detekcia hrán [6]
- Sobelov hranový detektor
- Laplaceov filter
- Cannyho hranový detektor [7]

Viac si k princípom konvolúcie povieme v kapitole 3, ktorá je zameraná na vysvetlenie metód využitých v tejto práci.

Segmentácia obrazu s využitím morfologických operácií

Už sme uviedli, že segmentácia je vlastne rozdelenie obrázka do niekoľkých oblastí s rovnakou vlastnosťou. V článku [8] autori zo SVMIT Engineering College prezentujú vylepšený segmentačný algoritmus, pozostávajúci z dvoch fáz. V prvej fáze aplikujú na vstupný obrázok hranový detektor, pričom pre tieto účely využili hybridnú Fuzzy-Cannyho metódu. Ide o kombináciu klasickej Cannyho hranovej detekcie(kapitola 3.2.3) s fuzzy interferenčným systémom, pričom výsledkom by malo byť zlepšenie výsledkov klasickéj Cannyho hranovej detekcie. Ako prvé sa teda „fuzzyfikuje“ obrázok. Výstup z fuzzyfikácie sa následne použije ako vstup pre klasickú Cannyho detekciu hrán.

V druhej fáze popisovanej segmentácie použili snímok s detegovanými hranami v prvej fáze. Vstupom je teda binárny obrázok, zobrazujúci detegované hrany. Na takýto obrázok následne aplikovali morfologické operácie. Tie sa využívajú za rôznym účelom ako napríklad analýza textúr obrázka, eliminácie šumu, či ohraničovania objektov. V článku sú ďalej popísané dve najzákladnejšie morfologické operácie: dilatácia a erózia. Dilatáca je operácia, ktorá pridáva pixely na hranách objektov a erózia ich zas odoberá. Obe operácie potrebujú na vstupe binárny respektíve šedotónový obrázok a kernel. Kernel je matica, ktorá definuje, ako veľmi majú byť objekty na obrázku dilatované, respektíve erodované. V článku je vysvetlená technika, ktorá pri správnom nastavení jednotlivých parametrov umožňuje spájať objekty do väčších sémanticky významnejších celkov. Táto metóda sa radí k tým jednoduchším spomedzi spôsobov obrazovej segmentácie. Ide o relatívne efektívnu metódu, ktorá v určitých prípadoch menej komplexných úloh dáva uspokojivé výsledky.

2.1.2 Segmentácia v rámci ANPR systémov

Ondrej Martinský vo svojej práci [9] detailne popisuje metódy využívané v probléme ANPR (Automatic Number Plate Recognition). ANPR je problém, ktorý rieši detekciu tabuľky s evidenčným číslom vozidla. Tento problém je známy a my sme ho už spomenuli v úvodnej kapitole vďaka jeho podobnosti s problémom segmentácie oblasti s displejom vodomeru na obrázku. Algoritmy riešiace ANPR problém sa obvykle skladajú z nasledujúcich štyroch krokov:

1. Predspracovanie obrazu - v rámci tohto kroku sa uskutočnia všetky potrebné úpravy obrázka pred začatím analýzy jeho obsahu.

2. Lokalizácia tabuľky - na základe horizontálneho a vertikálneho hranového histogramu alebo inej analýzy sa deteguje oblasť obrázka, na ktorej sa nachádza tabuľka s EČV.

3. Segmentácia znakov - z vyseknutej časti obrázka obsahujúcej iba tabuľku s EČV sa v tomto kroku vysegmentujú všetky znaky separátne.

4. Rozpoznanie znakov - vysegmentované znaky sa následne identifikujú prostredníctvom OCR (Optical Character Recognition).

Pre riešenie problému s detekovaním displeja na vodomeri sa inšpirujeme prvými dvoma krokmi vyššie spomenutého postupu. Vhodné predspracovanie obrazu a lokalizácia objektu na ňom sú dva problémy, ktoré budeme potrebovať vyriešiť aj pri práci s vodomermi. Metódy detekcie EČV síce zahŕňajú aj segmentáciu znakov a ich rozpoznávanie, my sa však zameriame iba na lokalizáciu a segmentovanie samotnej tabuľky z obrázku. To, že ľudský mozog vie veľmi ľahko a jednoznačne identifikovať na automobile jeho EČV, pre počítač až tak jednoduché a zrejmé nie je. Počítač musí EČV detegovať „svojím pohľadom“ súradníc obrázka a matematických funkcií.
Úlohou je naučiť počítač zamerať sa na to, čo človek vidí ako obdĺžnikový kus plechu popísaný znakmi zväčša umiestnený na prednej a zadnej strane vozidla. Môžeme povedať, že celá problematika počítačového videnia sa točí okolo otázky, ako preložiť a definovať ľuďmi vnímanú realitu do jazyka zrozumitelného počítačom. Prvým krokom v tejto úlohe je definovať EČV tak, aby ho algoritmus mohol podla danej definície hľadať.

Povedzme, že EČV je *obdĺžniková oblasť so zvýšeným počtom horizontálnych a vertikálnych hrán*. Takáto definícia je preložiteľná do matematickej reči. Otázne je, či je dostatočne jednoznačná. Akú heuristiku stanoviť v prípade, že definícii vyhovujúcich oblastí na obrázku bude viacero? Algoritmus preto vyberie N najvyhovujúcejších adeptov a na tých následne aplikuje ďalšie výberové algoritmy.

Aby bol algoritmus schopný analyzovať obrázok a vyhodnocovať výskyt hrán na ňom, je potrebné ho vhodným spôsobom predspracovať. Ak je treba, použijú sa algoritmy, ktoré sa vysporiadajú so svetelnými defektami a následne sa vhodne aplikujú hranové detektory, prípadne iné morfologické metódy na spracovanie obrazu. Detegované čiary autor potom rozanalyzoval v troch krokoch:

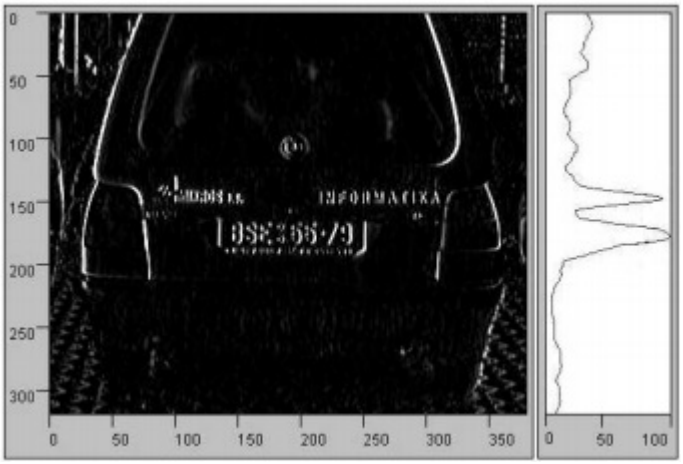
1. Horizontálna a vertikálna projekcia

2. Orezávanie obrázka

3. Výber kandidátov

Horizontálna a vertikálna projekcia

Ide o štatistickú analýzu obrázka pre vyhodnotenie hustoty hrán, ktoré na ňom boli detegované. Aplikovaním horizontálnej projekcie na obrázku transformovanom horizontálnym hranovým filtrom detegujeme oblasti na obraze s výraznejším počtom horizon-



Obr. 2.1: Obrázok s histogramom po aplikovaní vertikálnej projekcie [9]

tálnych hrán. Analogicky sa urobí analýza pre vertikálne hrany, tak ako je to ukázané na obrázku 2.1. Pomocou oboch projekcií sa na histogramoch definujú oblasti, ktoré obsahujú väčšie množstvo čiar oboch tipov.

Orezávanie obrázka

Pomocou maximálnych hodnôt na horizontálnom a vertikálnom histograme sa určia výseky, ktoré definujú potenciálnych kandidátov hľadanej tabuľky (viď ukážku na obrázku 2.2). Kedže obrázok chápeme ako svetelnú funkciu f definovanú ako $f = (x, y)$, vyseknutý pás b bude definovaný ako:

$$b = (x_{\mathrm{b0}}, y_{\mathrm{b0}}, x_{\mathrm{b1}}, y_{\mathrm{b1}}) \tag{2.1}$$

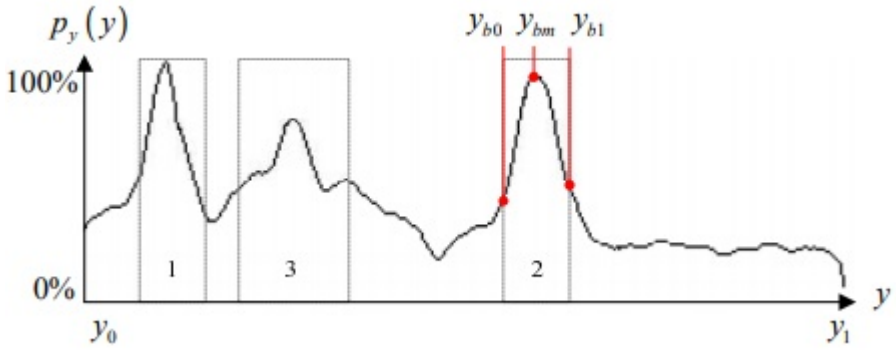
pričom platí:

$$(x_{\mathrm{b0}} = x_{\min}) \wedge (x_{\mathrm{b1}} = x_{\max}) \wedge (y_{\min} \leq_{\mathrm{b0}} <_{\mathrm{b1}} \wedge y_{\max})$$

V tomto prípade sa použije ako podklad snímok transformovaný horizontálnou projekciou. Z horizontálneho pásu teraz vysekneme vždy dvoma vertikálnymi čiarami miesta s maximálnou hustotou čiar. Pre výber vertikálnych výsekov použijeme pôvodný snímok transformovaný vertikálnou projekciou. Z každého pásu tak dostaneme práve jednu oblasť s potenciálnym kandidátom prítomnosti EČV. Obdĺžnik p, ktorý takto dostaneme, môžeme formálne definovať ako:

$$p = (x_{\mathrm{p0}}, y_{\mathrm{p0}}, x_{\mathrm{p1}}, y_{\mathrm{p1}}) \tag{2.2}$$

pričom platí:



Obr. 2.2: Graf histogramu vertikálnej projekcie [9].

$$(x_{\mathrm{b0}} \leq x_{\mathrm{p0}} \leq x_{\mathrm{p1}} \leq x_{\mathrm{b1}}) \wedge (y_{\mathrm{p0}} = y_{\mathrm{b0}}) \wedge (y_{\mathrm{p0}} = y_{\mathrm{b0}})$$

Celý tento proces (s menšími technickými rozdielmi) aplikujeme na každom vybranom kandidátovi ešte raz, aby sme zúžili vybranú oblasť. Histogram danej projekcie zobrazuje v hodnôt, pričom v je výška obrazu (počet riadkov pixelov). Každá hodnota v histograme reprezentuje množstvo svetelných hrán v prislúchajúcom riadku pixelov. Nás zaujímajú lokálne maximá tohto grafu. Tie definujú množinu riadkov obrázka so zvýšeným množstvom hrán. Každé takéto lokálne maximum predstavuje jedného kandidáta.

Niekedy však štatistická analýza nevráti úplne vyhovujúci výsledok. Dáta môžu mať veľký rozptyl a z histogramu nebude úplne možné jednoznačne určiť jednoznačných kandidátov. Tento nepriaznivý stav sa však dá trocha zlepšiť použitím konvolučných operácií, ako napr. rozostrenie obrázka (blur), rank vektor (zmenší štatistický rozptyl dát).

Výber kandidátov

V predošlom kroku bolo vysegmentovaných n oblastí z pôvodného obrázka. Počet kandidátov n obmedzíme na nejakú rozumnú hodnotu. Z nich treba určiť víťaza. Ten by mal v ideálnom prípade zobrazovať hľadanú tabuľku EČV. Autor prezentuje pre výber víťaza dvojkrokový algoritmus, ktorý v prvom kroku ohodnotí kandidátov pomocou jedenej alebo kombináciou viacerých funkcií. Následne kandidátov zoradí podľa ich priradenej hodnoty. Použité boli 4 funkcie v rámci prvého kroku. Hodnotu kandidátov vieme vyjadriť ako súčet týchto váhovaných funkcií.

$$\alpha = 0.15 \cdot \alpha_1 + 0.25 \cdot \alpha_2 + 0.4 \cdot \alpha_3 + 0.4 \cdot \alpha_4$$

- $\alpha_1 = |y_{\mathrm{b1}} - y_{\mathrm{b1}}|$
  Funkcia vyhodnocuje kandidátov podľa výšky vyseknutého pásu, pričom preferovaní sú kandidáti s menšou výškou.

- $\alpha_2 = 1/p_{\mathrm{y}}(y_{\mathrm{bm}})$
  Výraz v menovateli predstavuje výšku lokálneho minima grafu vertikálnej projekcie, podľa ktorej sa z obrázka vysekávali pásy s kandidátmi. Čím vyššie hodnoty lokálne maximum dosiahne, tým je kandidát preferovanejší.

- $\alpha_3 = 1/\sum_{y=y_{\mathrm{b0}}}^{y_{\mathrm{b1}}} p_{\mathrm{y}}(y)$
  V tejto funkcii, podobne ako v predošlej, vyhodnocujeme lokálne maximá grafu vertikálnej projekcie, ale tentokrát výraz v menovateli reprezentuje celú plochu pod jednotlivými oblasťami lokálneho maxima. Opäť platí, čím väčšia plocha, tým preferovanejší kandidát.

- $\alpha_4 = ||x_{\mathrm{p1}} - x_{\mathrm{p1}}|/|y_{\mathrm{b0}} - y_{\mathrm{b1}}||$
  Posledná z uvedených funkcií je špecifická - prípad od prípadu. Autor v nej vyhodnocuje pomer výšky a sírky vysegmentovaného kandidáta. Aj medzi jednotlivými tabuľkami s EČV je to veľmi špecifické, kedže v rôznych krajinách existujú rôzne typy tabuliek. Autor sa zameriava na tuzemský typ jednoriadkových tabuliek, kde pomer medzi šírkou a výškou tabuľky je približne 5. Kandidáti s pomerom najbližšie k tomuto číslu sú preto najpreferovanejší. V našej práci bude pri využití týchto analýz zásadné dobré popísanie hľadaných vodomerov.

V druhom kroku potom analyzuje jednotlivých kandidátov do hĺbky od toho najúspešnejšieho, podľa vyššie popísanej analýzy, k menej významným, až kým nejakého neprehlási za hľadaný objekt. HĺbkovÁ analýza preverí analyzovaného kandidáta, či spĺňa podmienky odpovedajúce hľadanému objektu.

Metóda Template matchingu a jej aplikácie

V článkoch, ktoré sme popisovali doposiaľ, sa na segmentáciu obrazu pozeralo spôsobom, ktorý sa v princípe skladal z dvoch fáz. V prvej fáze mohol byť vstupný obrázok spracovaný a pripravený na segmentáciu. Takáto predspracovanie zahŕňalo transformáciu na šedotónový obrázok, na ktorom sa vykonala procedúra, ktorá buď detekovala hrany, rohy alebo inak vyznačila kontúry vstupného obrázka. Segmentácia takéhoto obrazu potom spočívala v analýze detegovaných kontúr, ktoré mohli byť ešte dodatočne modifikované morfologickými operáciami. V prípade metódy template matchingu to funguje trocha inak. Algoritmus sa snaží na vstupnom(referenčnom) obrázku lokalizovať miesta, ktoré korelujú so vzorovým obrázkom. Vzorový obrázok zobrazuje nejaký objekt alebo scénu, ktoré nás zaujímajú. Pre naše potreby sa takýto prístup javí ako vyhovujúci, kedže lokalizácia displeja vodomeru je presne to, čo potrebujeme docieliť. V článku [10] je prezentované využitie algoritmu Template matchingu za účelom registrácie lekárskych snímkov. Jedná sa o metódu, ktorá transformuje viaceré snímky

do jedného súradného systému. Táto metóda má využitie vo viacerých diagnostických postupoch. V tomto článku sú použité šedotónové snímky ľudského mozgu s rozmermi 512x512 pixelov. Na snímkoch mozgu sú následne pomocou rôznych metód template matchingu lokalizované vzorové obrázky rôznej veľkosti, zobrazujúce vybrané časti snímok ľudského mozgu.

Základný princíp jednoduchého Template matchingu spočíva v tom, že pre každú pozíciu predlohy, resp. vzorového obrázku, oproti referenčnému obrázku sa spočíta miera ich podobnosti.Všetky pozície, v ktorých je medzi oboma snímkami korelácia väčšia ako stanovený prah sa označia na referenčnom obrázku ako potenciálne miesto výskytu hľadaného vzoru. Jedna z vecí, v ktorej sa líšia rôzne implementácie, môže byť práve spôsob porovnania podobnosti oboch obrázkov. Medzi najpoužívanejšie funkcie pre výpočet korelácie medzi dvoma obrázkami sa používajú:

- Sum Of Absolute Differences (SAD)
- Sum Of Squared Differences (SSD)
- Normalized Cross Correlation (NCC)

Bližšie si jednotlivé funkcie priblížime v kapitole 3. V práci [10] sa pre výpočet podobnosti používa metóda normalizovanej krížovej korelácie (NCC). Metóda NCC porovnáva normalizované obrázky, ktoré dostane odpočítaním priemernej hodnoty a následným vydelením štandardnou odchyľkou. Je to robustné riešenie, ktoré je výpočtovo pomerne náročné, ale jeho výhodou je odolnosť voči defektom v kontraste, či svetelnosti porovnávaných obrázkov. Nevýhodou zasa je, že táto metóda nie je schopná sa vysporiadať s rôznou rotáciou, skosením alebo rozdielnym merítkom oboch snímok. Ďalej je v článku prezentovaný algoritmus registrácie snímkov a jeho výsledky. Ten konkrétne pre naše účely potrebný nebude, avšak metóda počítajúca koreláciu medzi rôznymi prekrytiami referenčného a vzorového obrázka, môže byť spôsob, ktorý by pri správnom zvolení vzorového obrázka mohol na snímkach s vodomermi detegovať potrebné časti vodomeru. Nevýhody štandardného Template matchingu, tak ako je prezentovaný v predošlom článku, sa pokúsili vylepšiť autori v článku [11] metódou, ktorá jednotlivé variácie vzájomnej pozície vzorového a referenčného obrázka simuluje. Výsledkom je algoritmus, ktorý by mal byť invariantný aj voči rozdielnej rotácii, skoseniu, či priblíženiu referenčného obrázka voči vzorovému obrázku a zároveň by mal byť dostatočne rýchly. V článku sú odprezentované dva algoritmy. Jeden z nich je „brute force“ algoritmus, teda algoritmus, ktorý hrubou silou vyskúša nájsť vzor na referenčnom obrázku spôsobom obyčajného template matchingu pre dostatočne veľa možných natočení, priblížení a skosení. Takýto spôsob si vie síce poradiť s rozdielnou pozíciou porovnávaných obrázkov, ale je to za cenu vysokej výpočtovej zložitosti. Druhý popísaný algoritmus je nazývaný „Ciratefi“, ktorý je efektívnejšou variantou k vyššie

popísanému brute force algoritmu. Ciratefi funguje na princípe redukovania pixelov referenčného obrázka, ktoré určite nereprezentujú vzorový obrázok na referenčnom. Takéto redukovanie prebieha v 3 krokoch filtrovania.

1. Cifi (Circular sampling Filter) - v rámci tohto filtra sa z pixelov určia kandidáti prvého stupňa a tiež ich pravdepodobnostný koeficient priblíženia.

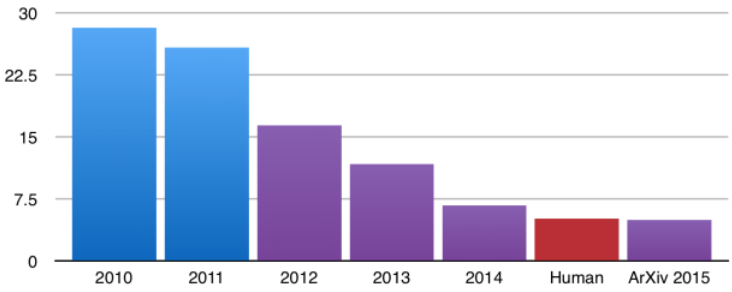
2. Rafi (Radial sampling Filter) - tento filter využije projekciu oboch porovnávaných obrázkov na radiálne čiary a opäť určí zo zostávajúcich pixelov kandidátov druhého stupňa. Ostatné pixely sú vyradené.

3. Tefi (Template matching Filter) - je štandardný template matching, ktorý je sám o sebe invariantný voči svetelným defektom obrázkov.

Autori v článku tieto algoritmy vzájomne porovnávali. Ak sa pri „brute force“ algoritme zvolí 6 rôznych hodnôt škálovania, ktoré by sa vyskúšali pre 36 rôznych stupňov natočenia referenčného obrázka, musela by byť metóda Template matchingu aplikovaná 216-krát pre každý vstup. Takýto algoritmus bežal na 3-GHz procesore 9173s, kým pri použití „Ciratefi“ algoritmu na rovnakom vstupe dostali výsledok za 22s.

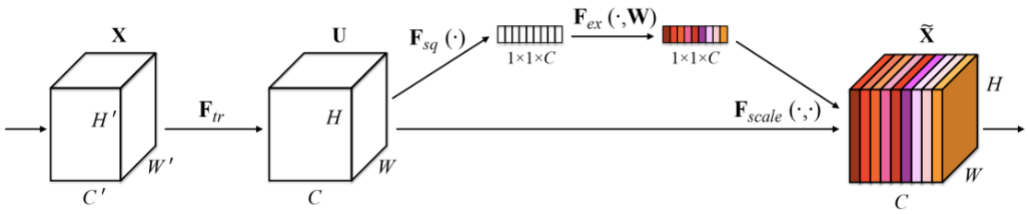
2.2 Lokalizácia objektu v obraze s využitím hlbokých neurónových sietí

Hlboké učenie je súčasťou metód strojového učenia. Ide o metódu, ktorá umožňuje odpovedať na rôzne otázky týkajúce sa dát, vďaka tomu, že sa z veľkého množstva podobných dát na tieto otázky naučila odpovedať. Hodnoty, ktoré dostaneme z výstupných neurónov, predstavujú odpoveď neurónovej siete pre údaje, ktoré sme jej dali na vstupe. K tomu, aby bola neurónová sieť schopná vyhodnocovať vstupy s malou chybovosťou, je potrebné jej „ukázať“ veľké množstvo anotovaných (trénovacích) dát. Množstvo potrebných dát na trénovanie siete je závislé od komplexnosti problému, ktorý očakávame, že neurónová sieť bude riešiť. Neurónové siete sa v posledných rokoch stali populárnym nástrojom využívaným v mnohých odvetviach, medzi iným aj v oblasti spracovania obrazu. Riešenia stojace na princípe hlbokého učenia v tejto oblasti dosahujú už niekoľko rokov najlepšie výsledky v obore. Ak sa zameriame len na oblasť segmentácie obrazu, ktorá je v našej práci kľúčová, na obrázku 2.3 môžeme vidieť, že aj v tejto oblasti sú state-of-the-art výsledky dosahované tiež pomocou hlbokých neurónových sietí. Pojem *state-of-the-art* budeme v ďalšom texte rozumieť ako najlepšie výsledky z danej oblasti podľa najnovších poznatkov.



Obr. 2.3: Graf vývoja úspešnosti segmentačných algoritmov na datasete ImageNet [61] od roku 2010 [12].

Prvé významnejšie výsledky sa podarilo dosiahnúť v roku 2012 konvolučnou neurónovou sieťou AlexNet [13], ktorá sa skladala z 5 konvolučných vrstiev a 3 plne prepojených vrstiev. Sieť bola natrénovaná na približne 15 miliónoch obrázkov datasetu ImageNet, rozdelených do viac ako 20 000 kategórií. Najnižšia chyba na tejto neurónovej sieti sa hýbala na úrovni 15,3%, čím prekonala vtedajšie najlepšie výsledky o viac ako 10%. Ako je uvedené aj v zdroji [14], toto prventstvo si od 2012 odovzdalo viacero prezentovaných riešení. Aktuálny „state-of-the-art“ pochádza z roku 2017, kedy boli predstavené takzvané SE-Nets (Squeeze-and-Excitation Networks) [15]. Ide o riešenie, ktoré v architektúre konvolučnej neurónovej siete využíva nový prvok nazývaný SE-blok.



Obr. 2.4: Architektúra konvolučnej neurónovej siete zloženej zo SE ("Squeeze and Excitation") blokov [16].

SE-bloky zabezpečujú zvýšenie vzájomnej závislosti kanálov siete za pomerne nízku výpočtovú cenu. Myšlienka spočíva v pridaní parametra pre každý kanál siete, aby mohla byť pri trénovaní jeho váha ľahko modifikovaná. Toto umožní rozlišovať jednotlivé filtre siete podľa ich relevantnosti pre výsledok. Benefitom je, že SE-bloky môžu nahradiť konvolučné bloky v architektúre už existujúcich neurónových sietí(obrázok. 2.4).

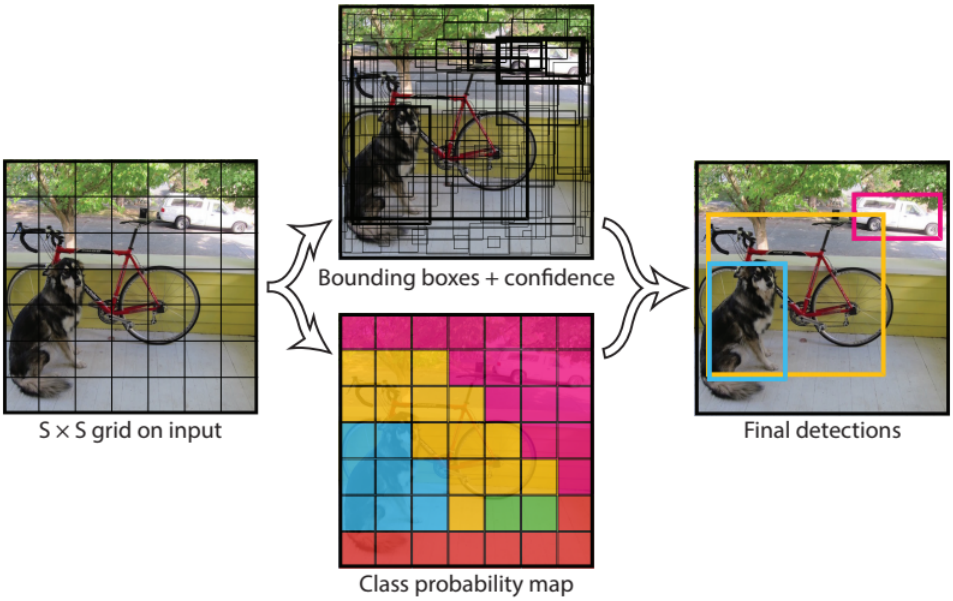
2.2.1 YOLO

O detekcii objektov v oblasti počítačového videnia už vieme, že sa vzťahuje na lokalizáciu a rozpoznanie (klasifikáciu) hľadaného objektu. Skoršie algoritmy detegovali objekty dvojkrokovo. V prvom kroku detegujú objekty na obrázku a v ďalšom kroku sú nájdení kandidáti, pomocou konvolučnej neurónovej siete, klasifikovaní do vopred definovaných kategórií. Napriek tomu, že algoritmy tohto typu boli viackrát vylepšované (Fast R-CNN [17], Faster R-CNN [18], Mask R-CNN [19]) a v princípe dosahujú perfektné výsledky, ich hlavným nedostatkom je slabá výpočtová rýchlosť. Práve metóda YOLO (You Only Look Once) je citelne rýchlejšou alternatívou detekcie objektov na obrázku k vyššie uvedeným variantom. Túto metódu prvýkrát prezentoval Joseph Redmon v roku 2015 [20] a využíva takzvaný One-Stage detektor. V metóde YOLO sa zlučuje lokalizácia aj detekcia objektov do jedného kroku, v ktorom používa jednu konvolučnú neurónovú sieť. Táto sieť už detegovanie objektov nerieši ako klasifikačný, ale ako regresný problém. Vďaka tomu, že YOLO sa pozerá na celý obrázok a nie iba na jeho výsek, ako tomu je pri dvojkrokových metódach spomenutých v texte vyššie, dosahuje YOLO menšiu chybovosť v situáciach, keď pracuje s novým typom dát. Takýto prístup je výmenou, v ktorej za relatívne malý úbytok v presnosti dostaneme výrazné zrýchlenie behu algoritmu. Ten vie spracovať 150 snímkov za sekundu, čo znamená, že YOLO algoritmus je schopný detegovať objekty na videu počas jeho behu s odozvou menej ako 25 milisekúnd. [20] Od prvej verzie z roku 2015, ktorú poznáme pod označením „v1“ boli medzičasom prezentované ďalšie dve vylepšené varianty (YOLOv2, YOLOv3). Všetky tri však majú princíp rovnaký.

Algoritmus rozdelí obrázok na mriežku s rozmerom N x N dielikov, pričom každý dielik predikuje najviac jeden objekt. V rámci jedného dielika je predikovaný obmedzený počet ohraničení, ktoré majú ukázať, kde sa presne detegovaný objekt nachádza. Okrem vyznačenia predikovaného rámiku a pravdepodobnosti, že tento rámik skutočne vyznačuje nájdený objekt, vráti algoritmus pre detegovaný objekt aj pravdepodobnostné hodnoty príslušnosti daného objektu do preddefinovaných tried. Aby sme z množstva rámikov pre rôzne objekty dostali relevantný výsledok, vyberieme rámiky s najväčším pravdepodobnostným skóre. Tento proces je znázornený na obrázku 2.5.

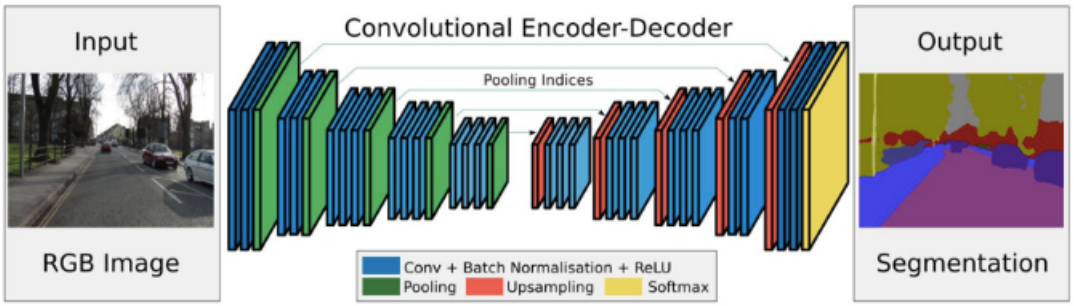
2.2.2 Sémantická segmentácia obrázka na úrovni pixelov

Sémantická segmentácia obrázka je problém, ktorý sa v oblasti počítačového videnia intenzívne skúma a existuje viacero rôznych prístupov k tejto problematike. V roku 2015 boli publikované články [21, 22], ktoré sa zaoberajú problémom rozdelenia obrázka do súvislých sémanticky jednotných oblastí na úrovni pixelov s využitím takzvaných SegNet sietí. Na obrázku 2.6 je možno vidieť, že SegNet využíva symetrickú architektúru ,



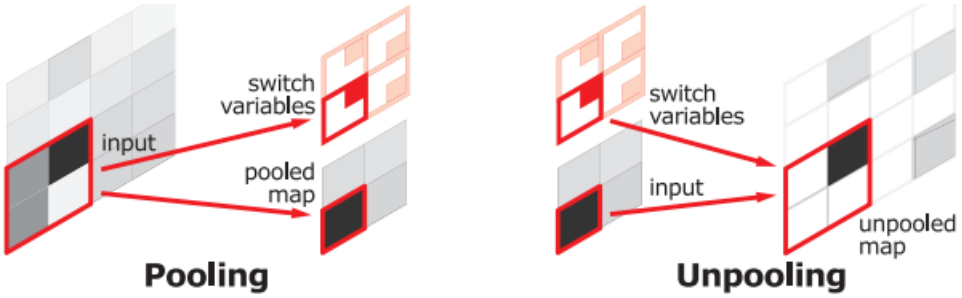
Obr. 2.5: YOLO rozdelí obrázok na mriežku s rozmermi N x N, pričom pre každý dielik odpredikuje B rámikov ohraničenia objektu s pravdepodobnosťami ich správnosti a s pravdepodobnosťami príslušnosti daného objektu do jednotlivých tried. [20].

ktorá pozostáva z konvolučných vrstiev kódovacej a dekódovacej časti siete. Enkóder a Dekóder sú síce zložené z rovnakých typov vrstiev rovnakého počtu, ale ich úloha v procese segmentácie pixelov na obrázku je odlišná. V prvej fáze Encoder prostredníctvom svojich vrstiev z obrázka extrahuje z množiny farebných pixelov „high level“ informáciu o objektoch, ktoré nimi boli zobrazované. Enkóder pozostáva z konvolučných vrstiev, batch-normalizačnej a nelineárnej vrstvy, za ktorými nasleduje max-poolingova vrstva. Každý typ vrstvy tejto konvolučnej neurónovej siete má inú funkciu. V ďalšej fáze, keď už enkóder detegoval čo vidí, je úlohou dekódera definovať, kde sa nájdené veci na vstupe nachádzajú. Inak povedané, dekóder musí klasifikovať jednotlivé pixely obrázka do tried, ktoré enkóder na vstupe detegoval. Povedali sme si, že SegNet sieť má symetrickú architektúru, ktorá je rozdelená na dve časti (jedna patriaca Enkóderu a druhá patriaca Dekóderu). Jednotlivé vrstivy rovnakého typu v enkóderi aj dekóderi fungujú principiálne rovnako, ale ich úloha sa v oboch kóderoch navzájom líši. Pomocou konvolučných vrstiev enkódera sú z obrázka extrahované jeho príznaky (features). Kým na opačnej strane v dekóderi konvolučná mapa „vypĺňa“ prázdne pixely po opätovnom zväčšení vstupu v predošlej vsrtve. Max-pooling je podvzorkovacia metóda, ktorá v enkóderi redukujé dimenziu obrázka aplikovaním filtra. Ten transformuje vstup do zredukovaného výstupu tak, že oblasť vstupného obrázka, na ktorú sa práve pozerá, reprezentuje na výstupe maximálnou hodnotou tejto oblasti. Takýto proces slúži na zamedzenie pretrénovania neurónovej siete. V dekóderi sa deje presne opačný proces, keď



Obr. 2.6: Architektúra konvolučného enkódera a dekódera [22].

jedna hodnota masky rozmerov 1x1 sa rozšíri na masku s rozmermi 2×2. Tu nastáva otázka, na ktorú pozíciu po upsamplingu má byť uložená hodnota z menšej masky tak, aby došlo k najmenšej chybovosti v ďalších vrstvách?



Obr. 2.7: Pamätanie si indexov maximálnych hodnôt vo fáze Poolingu (vľavo) a použitie indexov v rámci Unpooligu v dekóderi (vpravo) [23].

SegNet túto situáciu rieši ukladaním si pozície maximálnej hodnoty ešte v enkóderi, aby mohol dekóder využiť tento index práve pri fáze upsamplingu, tak ako je to znázornené na obrázku 2.7. SegNet ešte obsahuje batch normalizačné vrstvy, ktoré normalizujú distribúciu dát, aby sa urýchlilo učenie sa siete. Za dekodérom sa nachádza ešte jedna vrstva. Jedná sa o trénovateľný klasifikátor, ktorý klasifikuje každý pixel nezávisle.

Kapitola 3

Použité technológie a metódy

Vývoj v oblasti počítačového videnia naznačuje, že pre spracovávanie informácii z digitálnej snímky je funkčná a presná detekcia objektov, alebo sémanticky významných oblastí kľúčová. So zvyšujúcim sa množstvom informácii, ktoré spoločnosť v rôznych oblastiach generuje, narastajú aj nároky na výpočtovú efektivitu jednotlivých metód. Ako sme už v práci spomenuli, trend za posledné roky naznačuje, že strojové učenie, obzvlášť metódy stojace na báze hlbokého učenia, sú schopné sa s tradičnými úlohami z oblasti počítačového videnia vysporiadať najlepšie. Aj tieto prístupy však majú svoje úskalia. Jedným z nich je napríklad trénovanie neurónovej siete. Natrénovať neurónovú sieť je netriviálna optimalizačná úloha, ktorej cieľom je nájsť čo najlepšiu množinu váh pre danú sieť, ktoré by riešili špecifický problém. Dôležité je tiež disponovať dostatočným množstvom trénovacích dát. Aké množstvo dát je však dostatočné? V závislosti od úlohy a komplexnosti algoritmu, ktorý na jej vyriešenie chceme použiť, sa tieto požiadavky líšia. Ostala pred nami teda úloha, aplikovať jednotlivé segmentačné prístupy na náš dataset a vyhodnotiť relevantnosť ich použitia v našom probléme detekcie ciferníka na vodomeroch. V tejto kapitole síce neobsiahneme kompletne celú tému segmentácie objektov v rámci počítačového videnia, ale aspoň zbežne si priblížime detaily a technikality viacerých metód, ktoré sme v rámci experimentálnej časti práce vyskúšali. Tieto metódy si v nasledujúcich dvoch kapitolách detailnejšie popíšeme.

3.1 Predspracovanie obrazu

Rovnako ako pri detegovaní tabuľky EČV na obrázku, tak aj pri detegovaní počítadla na vodomeroch bude potrebné snímok upraviť tak, aby na ňom algoritmus mohol čo najpresnejšie rozpoznávať kontúry hľadaných objektov. Naznačuje to aj skutočnosť, že fotografie datasetu, ktorý v rámci nášho testovania využijeme, trpia na zlé svetelné podmienky, zlú ostrosť fotografie a podobné vady, ktorými by pravdepodobne trpeli aj fotografie zhotovené vodomeračmi v teréne. Neexistuje však univerzálny prístup, ako

vhodne spracovať obraz tak, aby segmentačné algoritmy, pre ktoré sa využije, dosahovali optimálne výsledky. Pre každý algoritmus je sada nástrojov, ktorá predspracuje jeho vstupy vyberaná individuálne. Tento výber je závislý od mnohých faktorov(kvalita dát, požiadavky použitej technológie, atď.). V tejto časti si predstavíme niekoľko štandardných nástrojov na predspracovanie obrazu, ktoré neskôr aplikujeme v rámci našich experimentov.

3.2 Binárne spracovanie obrazu

Väčšina segmentačných algoritmov, ktoré sme v rámci testovania vyskúšali, pracuje v nejakej časti s binárnymi obrázkami. Binárne obrázky majú pre spracovanie počítačom viaceré výhody. Jedna z výhod je jednoduchá reprezentácia a ukladanie informácie. Binárny obrázok je špeciálny druh šedotónového obrázka, v ktorom každý pixel môže nadobúdať hodnoty 0 alebo 255. Na uloženie tejto informácie preto postačuje 1 bit. Ďalšou výhodou je jednoduchšie spracovanie algoritmami. Kedže však náš dataset obsahuje farebné obrázky, je potrebné ich najprv do binárnej podoby transformovať. Prvým krokom takéhoto procesu je zväčša jeho transformácia na šedotónový. Z obrázka sa tak eliminuje informácia o sýtosti a odtieňoch a zachová sa iba informácia o jase. Takýto obrázok je matica bodov, kde je každý bod reprezentovaný hodnotou v rozsahu 0 - 255. Šedotónové obrázky sa následne spracúvajú klasickými alebo morfologickými operátormi, ktoré modifikujú obrázok pre potreby neskoršej segmentácie. Ako sa aj môžeme dočítať v [24], hranová detekcia patrí k základným nástrojom pri spracovávaní obrázka. Šedotónový obrázok vieme reprezentovať prostredníctvom funkcie f, ktorá definuje intenzitu jasu pixelu na súradniciach x a y. Výraznejšia zmena tejto funkcie znamená detegovanie pixelov na obrázku so zvýšeným kontrastom. Takúto zmenu funkcie interpretujeme ako detegovanie hrany na danom mieste obrázka, ktorý funkcia $f(x, y)$ popisuje. Viac detailov sa dá nájsť v práci [25].

3.2.1 Prahovanie

Pri segmentácii objektov je výsledkom najčastejšie matica, na ktorej je hľadaný objekt reprezentovaný pixelmi s hodnotou 1. Všetky ostatné body, ktoré nepredstavujú hľadaný objekt, majú hodnotu 0. Takýto obrázok nazývame binárna maska. Pomocou binárnej masky sme následne schopní lokalizovať hľadaný objekt na obrázku. Rozhodnúť však o nejakom bode či reprezentuje alebo nereprezentuje hľadaný objekt je neľahká úloha, ktorú riešia segmentačné algoritmy. Jedným zo základných a najjednoduchších segmentačných algoritmov je Prahovanie, ktoré je ako jedna zo základných metód spracovania binárnych obrázkov popísané v knihe Machine Vision [25]. Tento algoritmus rozhoduje o jednotlivých pixeloch vstupného obrázka na základe ich inten-

zity. Ak určíme pre segmentáciu prahovú hodnotu P, tak binárnu masku B získame zo vstupného šedotónového obrázka S nasledovne:

$$B\left[i,j\right] = \begin{cases} 1 \text{ ak } S[i,j] \leq T \\ 0 \text{ inak} \end{cases}$$

V princípe však pri prahovaní môžu byť podmienky klasifikovania pixelov definované aj komplikovanejšie. Vo všeobecnosti ale platí:

$$B\left[i,j\right] = \begin{cases} 1 \text{ ak } S[i,j] \in \mathbb{Z} \\ 0 \text{ inak} \end{cases},$$

kde $\mathbb{Z}$ je množina prípustných hodnôt intenzity.

Vybrať vhodnú hodnotu prahu alebo definovať správne množinu pre prípustné hodnoty intenzity bodov, je pomerne netriviálna úloha. Pre jej rišenie sa dnes väčšinou používajú algoritmy, ktoré hľadajú optimálne hodnoty prahu automaticky.

3.2.2 Konvolúcia

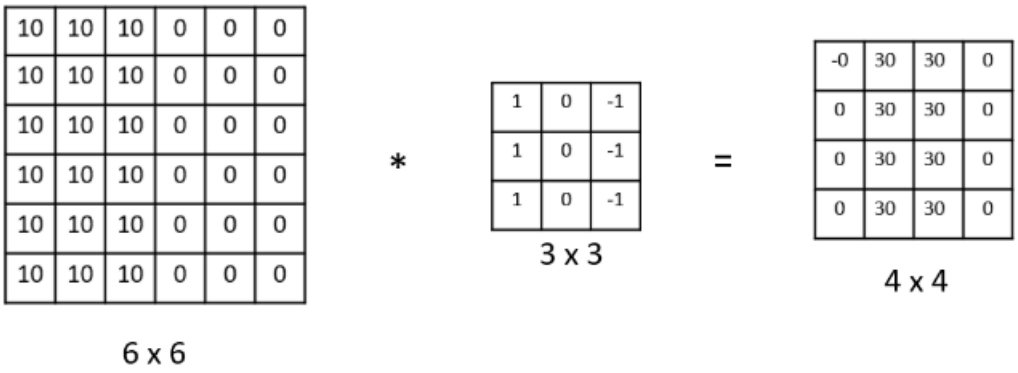
Pre spracovanie dvojrozmerných obrázkov sa využíva konvolúcia. V knihe „DIP - Digital Image Processing“ [26] je konvolúcia popísaná ako metóda, ktorá systematicky prechádza obrázok a pre každý jeho pixel vypočíta novú hodnotu pomocou malého okolia reprezentatívneho bodu. Diskrétna konvolúcia sa formálne definuje nasledovne:

$$g(x,y) = \sum \sum_{(m,n)\in O} h(x-m, y-n) f(m,n) \tag{3.1}$$

kde:

- f - je funkcia pôvodného obrazu
- g - funkcia nového (upraveného obrazu)
- h - konvolučná maska udávajúca koeficienty okolitých bodov práve upravovaného bodu

Princíp konvolúcie spočíva v tom, že pre každý pixel vstupného obrázka je vypočítaná nová hodnota pomocou konvolučnej masky a okolia daného bodu. V podstate ide o úpravu jednej matice druhou. Každá konvolučná maska jednoznačne definuje nejakú transformáciu alebo filter, ktorým sa upravuje zdrojový obrázok. Ukážeme si to na príklade jednoduchého operátora, ktorý slúži ako hranový detektor. Na obrázku 3.1 je vidieť, že vstupný obrázok s rozmermi 6x6 pixelov je transformovaný Prewittovým hranovým filtrom. Maska tohto filtra má rozmery 3×3. V tomto prípade má filter na



Obr. 3.1: Konvolučná maska (matica 3×3 v strede) deteguje zvislú hranu na vstupnom obrázku (matica vľavo) [28].

vstupnom obrázku detegovať vertikálne hrany. Rozmiestnenie váh v matici filtra (kerneli) určuje, na aké hrany má byť filtračná maska citlivá a hrany akého typu bude detegovať. V kapitole 2 sme uviedli, že existuje viacero hranových detektorov. Typ filtra hranového detektoru je hlavným činiteľom, ktorý definuje, na aký typ hrán bude daný filter citlivý a od toho aj závisí ich využitie.

3.2.3 Cannyho hranový detektor

Zameriame sa teraz na Cannyho hranový detektor, ktorý sme aj my používali v našej práci. Tento algoritmus vyvinul John F. Canny [7] a je považovaný za jeden z najuniverzálnejších a najpoužívanejších hranových detektorov. Cannyho algoritmus výpočtu pozostáva z piatich krokov, ktoré si teraz vysvetlíme tak, ako ich opisujú články: [29], [24].

1. Redukcia šumu

V prvom kroku je potrebné redukovať potenciálny šum vstupného obrázka. Samotná detekcia hrán je na neho senzitívna a mohol by skresliť výsledok. V Cannyho algoritme sa preto najskôr aplikuje Gaussov kernel (vzorec 3.3) na transformovanie vstupného obrázka (vzorec 3.2).

$$g(m,n) = G_\sigma(m,n) \star f(m,n) \tag{3.2}$$

$$G_\sigma(m,n) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{m^2+n^2}{2\pi\sigma^2}) \tag{3.3}$$

Čím má kernel väčšie rozmery, tým viac sa prejaví intenzita vyhladzovania šumu.

2. Výpočet gradientu

V tomto kroku algoritmus určí gradienty a k tomu využije štandardný Sobelov

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_x = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Obr. 3.2: Sobelove filtre pre detekciu horizontálnych a vertikálnych hrán

operátor hranovej detekcie. Ten sa skladá z dvoch filtrov(Obr. 3.2), pričom každý z filtrov slúži na detekciu hrán iného typu.

Aplikovaním konvolúcie na vstupný obrázok pomocou oboch filtrov sa na výstupe zvýraznia vertikálne aj horizontálne čiary vstupného obrázka. Po detegovaní hrán sa ešte vypočíta smer a veľkosť gradientu, pričom výsledný smer je zaokruhlený na jednu zo štyroch možných hodnôt stupňov natočenia (0, 45, 90, 135). Smerov je síce dohromady osem, ale znamienko vypočítanej hodnoty v tomto prípade nehrá úlohu. Následne totiž budú v každom kroku vyhodnocované body ležiace oproti sebe. Pre každý pixel vstupného obrázka sa vypočíta veľkosť gradientu a jeho smer podľa vzorcov 3.4 a 3.5:

$$|G| = \sqrt{V_x^2 + V_y^2} \tag{3.4}$$

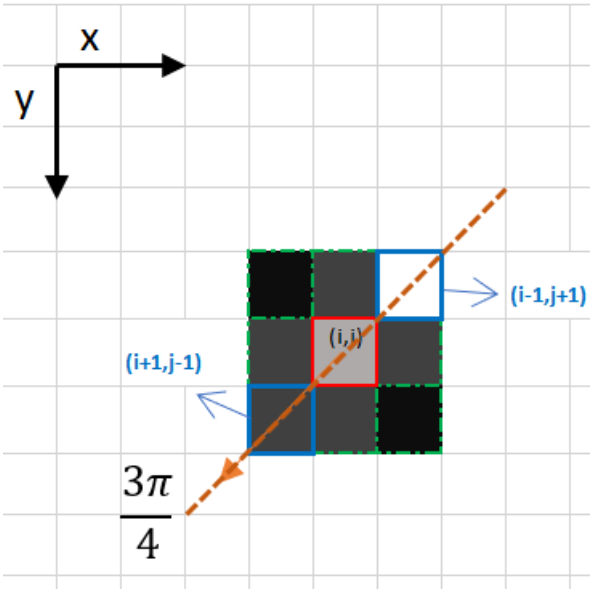
$$\theta(x, y) = \arctan(\frac{V_y}{V_x}) \tag{3.5}$$

kde V_x a V_y predstavujú transformovaný vstup pomocou horizontálneho a vertikálneho hranového filtra.

V tomto bode máme detegované hrany. Kedže želaný výsledok by mal mať všetky detegované hrany rovnomerne výrazné, je potrebné dosiahnuť, že pre každý výsledný pixel bude jeho hodnota buď 0 alebo 255.

3. Nájdenie lokálnych maxím

Aby sme tento želaný stav dosiahli, je treba prejsť obrázok a na hranách hľadať lokálne maximá. V praxi sa to prejaví stenšením detegovaných hrán. Funguje to tak, že algoritmus si ako prvé vytvorí masku M zhodnej veľkosti, ako je vstupný obrázok, inicializovanú na samé 0. Jedná sa teda o maticu rovnakej veľkosti, ako je matica spočítaných gradientov v predchádzajúcom kroku Cannyho algoritmu. Táto maska slúži na zaznamenanie si výsledkov hľadania lokálnych maxím.
Následne algoritmus prechádza každý pixel vstupného obrázku s detegovanými hranami pomocou filtra veľkosti 3×3. Jednotlivé pixely sú vyhodnotené nasledovne. Algoritmus sa pozrie na smer gradientu v danom bode, ktorý nazveme



[28]

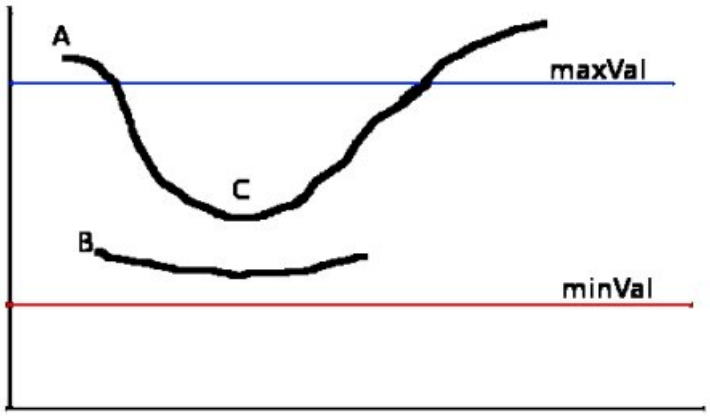
Obr. 3.3: Vyhodnocovanie bodu X_1 na súradniciach $V(i, j)$ v rámci hľadania lokálnych maxím [29].

X_1 ležiaci na súradniciach $V(i, j)$ vstupnej matice V, viď Obr. 3.3. Potom sa pozrie na dva susedné body, ktoré ležia v smere gradientu bodu X_1. Tieto body označme X_2 a X_3. Teraz sa pozrie algoritmus na hodnoty bodov X_1, X_2, X_3 a pokiaľ $val(X_1) = MAX(val(X_1), val(X_2), val(X_3))$, tak na pozíciu $M(i, j)$ uloží hodnotu $val(X_1)$. V opačnom prípade tam uloží hodnotu 0. Po prejdení celej matice je výsledná detekcia po eliminácii hrán uložená v matici M.

4. Dvojité prahovanie

Po aplikovaní predošlého kroku sa detegované hrany stenšili, avšak stále môžu existovať v obrázku pixely, ktoré sa doň dostali vďaka zostatkovému šumu, alebo kvôli zhoršeným svetelným podmienkam pôvodného obrázku. Snahou je zbaviť sa takýchto pixelov v obrázku. To sa dá docieliť použitím dvojitého prahovania. Dvojité prahovanie spočíva v definovaní si dvoch prahových hodnôt (viď obrázok 3.4). Vďaka tomu rozdelíme pixely obrázka do troch kategórii podľa ich intenzity jasu. Informáciu o príslušnosti každého pixelu do jednotlivých kategórii si budeme uchovávať v matici T. Význam takéhoto rozdelenia spočiva v tom, že o silných pixeloch(pixely s hodnotou jasu nad hranicou horného prahu) vieme s určitosťou povedať, že nesú relevantnú informáciu o nejakej hrane. Slabé pixely (hodnotu jasu majú pod spodným prahom) zas s veľkou pravdepodobnosťou nenesú pre nás podstatnú informáciu. Jedná sa väčšinou o šum v obraze a preto ich vyradíme. Tretiu skupinu takzvaných neutrálnych pixelov (majú hodnotu intenzity jasu medzi horným a spodným prahom) je potrebné rozhodnúť, či nesú alebo nenesú

relevantnú informáciu. To sa urobí v poslednom kroku algoritmu.



Obr. 3.4: Ukážka princípu dvojitého prahovania obrázku [30].

5. Mapovanie hrán hysteréziou
V tomto kroku už iba prejdeme maticu T a pre každý pixel zo skupiny strednej intenzity rozhodneme, či na výstupe bude alebo nebude zobrazený. Maticu T prejdeme kernelom k s rozmermi 3×3. Kernel k funguje tak, že pokiaľ stredne silný pixel, nad ktorým je momenálne stred kernelu k, má vo svojom okolí aspoň jeden silný pixel, potom aj daný pixel sa stane silným a bude zobrazený na výstupe. V opačnom prípade nebude zobrazený.

3.2.4 Binárne morfologické transformácie

Jedná sa o sadu nelineárnych operácií, ktoré sú využívané v rámci predspracovávania obrazu (eliminácia šumu, zjednodušenie tvarov v obraze a ďalšie). Oproti predošlým technikám, ktoré sme už v tejto kapitole popísali, sa morfologické operácie líšia v tom, že namiesto analyzovania intenzity jasu jednotlivých bodov sa sústredia skôr na ich relatívne usporiadanie. Objasníme si teraz základné pojmy, týkajúce sa morfologických operácií, ako sú popísané v práci [31]. Morfologickým spracovaním obrazu sme schopní detegovať štruktúru a geometrické tvary zobrazených objektov. Morfologické operácie sa rozlišujú podľa toho, či pracujú s binárnymi alebo so šedotónovými obrázkami. Pri binárnych obrázkoch je bod v obraze (pixel) reprezentovaný usporiadanou dvojicou čísel. V prípade šedotónových obrázkov je to usporiadaná trojica čísel. Celý obrázok potom chápeme ako bodovú množinu. Morfologické transformácie sú potom reláciou medzi bodovou množinou obrazu a takzvaným „štruktúrnym elementom“. Ten si definujeme tiež ako množinu bodov usporiadaných do rôznych tvarov, z ktorých jeden predstavuje reprezentatívny bod. V rámci morfologických transformácií sa štruktúrny element systematicky posúva po pozíciách na vstupnom obrázku a v závislosti od zvolenej morfologickej operácie je s ním porovnávaný. V jednoduchosti si teraz vysvetlíme

princíp fungovania dvoch základných morfologických operácií, ktoré sme aj my v práci viackrát aplikovali.

Dilatácia

Jedná sa o morfologickú transformáciu, ktorá na výstupe vracia binárny obrázok. V práci [32] sa uvádza, že tento obrázok vzniká ako vektorový súčet bodov dvoch množín M a B (Pozri vzorec 3.6). Vo výstupnom obrázku sa nachádzajú hodnoty 1 na všetkých pozíciách, ktoré prekrýval štruktúrny element B v momente, keď sa jeho reprezentatívny bod zhoduje s bodom obrázka M pod ním. Formálne sa táto operácia definuje nasledovne:

$$M \oplus B = \left\{ p \in \varepsilon^2 : p = m + b, m \in M, b \in B \right\} \tag{3.6}$$

Dilatácia má efekt rozširovania kontúr a vypĺňania medzier medzi regiónmi na obraze.

Erózia

Erózia binárneho obrázka M štruktúrnym elementom B je duálna transformácia k Dilatácii. Táto operácia vykonáva vektorový rozdiel medzi množinou bodov vstupného obrázka a menšou množinou bodov štruktúrneho elementu (viď vzorec 3.7) [32]. Eróziu dvoch množín bodov označujeme $M \Theta B$ a vypočíta sa nasledovne:

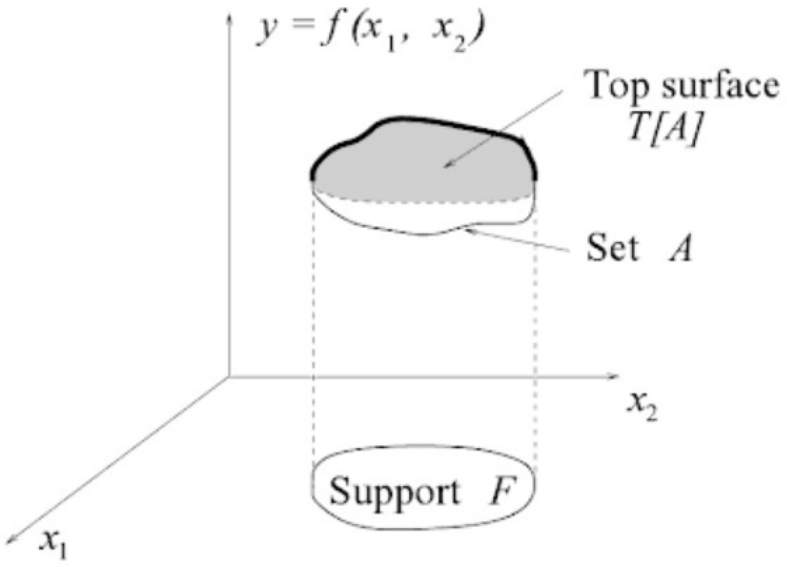
$$M \Theta B = \left\{ p \in E^2 : p + b \in X \quad pre \quad \forall b \in B \right\} \tag{3.7}$$

Výsledný efekt erózie je, že sa výsledné štruktúry zjednodšia, hrany objektov sa stenšia, prípadne detaily obrázka malých rozmerov celkom zaniknú. Podobne ako to bolo pri lineárnych filtroch, tak aj tu plati, že čím väčší je použitý Štruktúrny element, tým má použitá operácia väčší efekt.

3.2.5 Šedotónové morfologické transformácie

V úvode tejto kapitoly o morfologických transformáciách sme spomenuli, že bod v šedotónovom obrázku je reprezentovaný usporiadanou trojicou čísel, kde prvé dve hodnoty predstavujú súradnice bodu a tretia hodnota hovorí o intenzite jasu daného bodu. Kvôli tomuto je potrebné pri spracovávaní takýchto obrázkov rozmýšľať o trojrozmernom euklidovskom priestore. Napriek tomu, že vyššie popísané operácie a viacmenej všetky morfologické operácie boli primárne definované pre prácu s binárnymi vstupmi, využitie týchto operácií bolo rozšírené aj na prácu so šedotónovými obrázkami. Tomu výrazne dopomohol Stanley Sternberg, ktorý v roku 1978 zaviedol pojem Umbra [33]. Pojem Umbra sa v kontexte morfologických operácií neodmysliteľne spája s pojmom povrch funkcie. Povrch funkcie $T[A]$ predstavuje maximálnu hodnotu funkcie f v danom bode. **Umbra** pre funkciu f, ktorá reprezentuje šedotónový obrázok nad množinou bodov A

je množina všetkých bodov, ktorých intenzita jasu je menšia alebo rovná intenzite jasu v danom bode obrázka. Zjednodušene povedané, je to celý priestor nachádzajúci sa pod vrcholom funkcie f, tak ako to znázorňuje obrázok 3.5. Funkcia f priraďuje každému



Obr. 3.5: Povrch funkcie reprezentujúci maximálne hodnoty bodov množiny A [34].

bodu množiny A, ktorá reprezentuje vstupný obrázok, hodnotu jeho intenzity jasu. Potom povrch funkcie $T[A]$ formálne definujeme nasledovne:

$$T[A](x) = \max\{y \mid (x, y) \in A\} \tag{3.8}$$

Pre rozšírenie základných binárnych morfologických operácií, ako je dilatácia a erózia, sa použili funkcie **min** a **max**. Tieto funkcie určujú minimálnu, resp. maximálnu hodnotu intenzity jasu daného pixelu.

Pri erózii sa využije operácia *min* tak, že sa v každej pozícii vyberie minimálna hodnota z obrazových hodnôt, ktoré sú v danom momente vo výseku, ktorý pokrýva štruktúrny element. Šedotónová erózia sa formálne môže zapísať nasledovne:

$$f \Theta s = V\{U[f] \Theta U[s]\} \tag{3.9}$$

Šedotónová dilatácia na rozdiel od erózie vyberá namiesto minimálnej hodnoty maximálnu z obrazových hodnôt daného výseku. Aj pri šedotónových obrázkoch platí, že medzi týmito dvoma operáciami platí dualita. Formálne sa táto operácia zapíše nasledovne:

$$f \oplus s = V\{U[f] \oplus U[s]\} \tag{3.10}$$

3.3 Metódy hľadania zhody

Predstavíme si teraz ďalšiu populárnu metódu, ktorá sa často používa v oblasti počítačového videnia v rámci úloh súvisiacich s detekciou objektov na obrázku. Základný princíp metódy obyčajného Template matchingu, v článku [35] nazývaného tiež „Naivny Template matching“, spočíva v detegovaní výseku na obrázku, ktorý pre tieto účely budeme volať referenčný. Detegovaný výsek má čo najviac zodpovedať menšiemu obrázku, ktorý budeme volať vzorový. Algoritmus tak na vstupe dostane tieto dva obrázky (vzorový a referenčný) a postupne porovnáva podobnosť vzorového obrázka so všetkými možnými podoblasťami referenčného obrázka, ktoré sú zhodných rozmerov ako vzor. Po prejdení celého obrázka algoritmus vráti podoblasť referenčného obrázka, ktorá vykazuje najväčšiu zhodu so vzorom. Existuje však aj viacero sofistikovanejších prístupov k tejto metóde. Vychádzajúc opäť z článku [35], sa jednotlivé prístupy template matchingu dajú rozdeliť do dvoch kategórií podľa prístupu, akým hľadajú zhodu medzi porovnávanými obrázkami:

- Prístup hľadajúci zhodné oblasti - takéto metódy sa zvyknú nazývať tiež korelačné.
- Prístup hľadajúci zhodné príznaky - tieto metódy detegujú na obrázkoch kľúčové príznakové body, ktoré sa usilujú medzi obrázkami spárovať.

Vrátime sa teraz k Naivnej metóde Template matchingu. Podstatným krokom tejto metódy je vyhodnotenie zhody (korelácie) medzi dvoma porovnávanými obrázkami. Koreláciou budeme číselne vyjadrovať vzájomnú podobnosť dvoch porovnávaných obrázkov. Na jej spočítanie sa používa viacero spôsobov.

Krížová korelácia (Cross-Correlation)
Je to základný spôsob pre spočítanie korelácie dvoch obrázkov. V podstate ide iba o sčítanie súčinov prekrývajúcich sa pixelov.

$$CC(O_1, O_2) = \sum_{x,y} O_1(x, y) \times O_1(x, y) \tag{3.11}$$

Suma absolútnych rozdielov (Sum of Absolute Differences)
Táto metóda je pomerne intuitívna a v podstate jej názov nám vo veľkej miere napovedá, o čo v nej pôjde. Pre každú možnú pozíciu vzoru na referenčnom obrázku sa spočíta súčet absolútnych hodnôt rozdielov pre každý pixel. Princíp fungovania si ukážeme na príklade dvoch menších matíc. :

Na obrázku 3.6 môžeme vidieť, že pre vzorový obrázok existujú práve dve možné pozície v rámci referenčného obrázka. Pre tieto dve pozície vzniknú matice s vý-

3	4	6
2	5	1
6	4	0

(a) Vzorový obrázok

3	4	5	1
6	3	1	2
7	4	0	1

(b) Referenčný obrázok

Obr. 3.6: Matice referenčného a vzorového obrázka.

slednými hodnotami tak, ako je to na obrázku 3.7. Výsledky v maticiach sú absolútne hodnoty rozdielov hodnôt pixelov. Čím je súčet hodnôt výslednej matice nižší, tým je korelácia medzi vzorovým obrázkom a danou pozíciou referenčného obrázka väčšia.

0	0	1
4	2	0
1	0	0

(a) Matica s výsledným porovnaním na ľavej pozícii (8).

1	1	5
1	4	1
2	4	1

(b) Matica s výsledným porovnaním na pravej pozícii (20).

Obr. 3.7: Matice s výslednými hodnotami korelácie.

Suma štvorcov rozdielov (Sum of Squared Differences)
Aj v tejto metóde sa porovnávajú prekrývajúce sa pixely referenčného a vzorového obrázka. Princíp je obdobný ako v metóde z predchádzajúcej ukážky, s tým rozdielom, že teraz sa rozdiely pixelom pred sčítaním ešte umocnia na druhú a až takéto hodnoty sú sčitané pre každú pozíciu vzoru na referenčnom obrázku.

Normalizovaná krížová korelácia (Normalized Cross-Correlation)
Táto metrika je vylepšením jednoduchej krížovej korelácie (vzorec 3.11). Jedna z výhod oproti pôvodnej metrike spočíva v konzistentnosti výsledkov pri globálnych výkyvoch jasu na obrázkoch. Toto je docielené odčítaním priemernej hodnoty jasu od každej hodnoty pixelu. Ďalšou zmenou je normalizovanie konečnej hodnoty korelácie do intervalu [-1,1].

3.4 Algoritmy párovania príznakov

Feature matching je ďalšia metóda, ktorá porovnáva dva obrázky a vyhodnocuje ich podobnosť, respektíve ich spoločné črty. Výhoda týchto metód spočíva v tom, že nemajú na referenčný obrázok až tak prísne pravidlá ako tomu je pri Template matchingových metódach hľadajúcich zhodnú oblasť. Tie majú problém vysporiadať sa s rôznym

priblížením alebo rotáciou porovnávaných obrázkov. Algoritmy identifikujúce príznaky na obrázkoch by mali byť invariantné voči škálovaniu aj rotácii referenčného obrázka. Existuje viacero algoritmov pre feature matching. Najznámejšie z nich sú: SIFT, SURF, BRIEF, ORB [36]. My sa pozrieme trocha detailnejšie na algoritmus SIFT (Scale Invariant Feature Transform), s ktorým prišiel v roku 2004 David G. Lowe z University of British Columbia [37] Na príklade tohto algoritmu si priblížime základné fungovanie metód feature matchingu. Pri popise jednotlivých krokov SIFT algoritmu som čerpal z [38] a [37]. Zjednodušene povedané, algoritmus SIFT funguje nasledovne. Najskôr na vzorovom obrázku deteguje takzvané kľúčové body, ktoré si zapamätá a informáciu o nich si uloží do deskriptorov. Následne algoritmus prechádza referenčným obrázkom a zisťuje, či sa na danom bode nenachádza niekorý zo zapamätaných kľúčových bodov. SIFT je pomerne netriviálny algoritmus, ktorý sa skladá z nasledujúcich štyroch hlavných krokov:

1. Detekcia kľúčových bodov

Pre detegovanie kľúčových bodov v obraze je potrebné nájsť lokálne maximá a minimá. V tomto bode algoritmus prehľadáva rôzne miesta v rôznom priblížení na obraze. Snaží sa tak určiť miesta na obraze, ktoré by boli detegovatelné z rôznych pohľadov a pri rôznej mierke. Aby však takéto miesta našiel, je potrebné najprv vytvoriť množinu zväčšení (Scale space). Koenderink (1984) [39] a Lindeberg (1994) [40] ukázali, že najlepšie pre tieto účeli je použiť Gaussovský kernel. Množinu zväčšení tak definujeme nasledovne:

$$L(x, y, \sigma) = G(x, y, \sigma) \star I(x, y) \tag{3.12}$$

kde $\star$ je konvolučný operátor, $I(x, y)$ je vstupný obrázok a $G(x, y, \sigma)$ je škálovatelný Gaussovský operátor 3.3. Za účelom detekcie extrémov v množine zväčšení algoritmus využíva funkciu DoG (Difference of Gaussian).

2. Lokalizácia kľúčových bodov

V predošlom kroku si algoritmus vytvoril zoznam potenciálnych kandidátov na vhodné kľúčové body. V tomto kroku ich pretriedi a vyradí zo zoznamu tie body, ktoré majú príliš nízky kontrast alebo sú ťažšie lokalizovateľné. Ide predovšetkým o body ležiace v okolí hrán, pretože by pri práci s hranami mohli pôsobiť ako nepriaznivý šum.

3. Priradenie orientácie kľúčovým bodom

Teraz treba kľúčovým bodom, ktoré ostali v zozname adeptov, priradiť orientácie. Deskriptor týchto bodov, ktorý algoritmus skonštruuje v ďalšom bode, tak dosiahne invariantnosť na rotáciu obrázka. Najprv sa vyberie správny obraz v

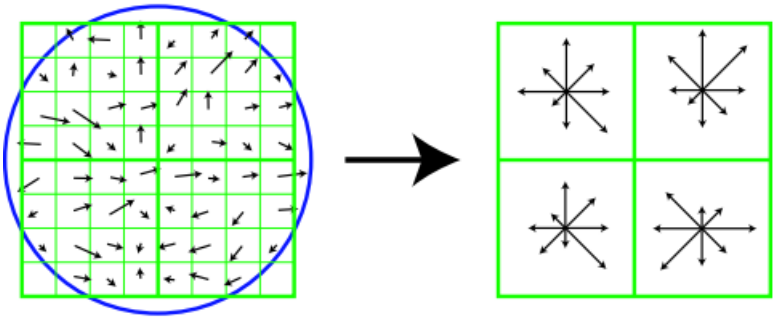
mierke daného kľúčového boda. Pre každý bod sa tak vypočíta veľkosť gradientu (vzorec 3.13) a orientácia daného bodu (vzorec 3.14).

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (3.13)$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \quad (3.14)$$

Z týchto údajov môže vzniknúť histogram orientácií gradientov jednotlivých bodov v okolí kľúčových bodov. Každý prvok tohto histogramu je tiež ováhovaný podľa vypočítanej veľkosti gradientu. Maximálny bod v histograme, teda bod s maximálnou hodnotou a niekoľko bodov s hodnotou aspoň 80% veľkosti maximálneho bodu, sa použijú na definovanie orientácie, resp. orientácií pre kľúčový bod v danom okolí.

4. Vygenerovanie deskriptorov
V poslednom kroku detekcie kľúčových bodov sú vygenerované Deskriptory. Štandardne sa detektor skladá zo 16 histogramov, v ktorých sú akumulované hodnoty veľkostí a orientácie gradientov bodov z okolia kľúčoveho bodu. Tieto histogramy sú uložené do mriežky s rozmermi 4×4, pričom každý histogram má napočítaných 8 orientácií. Ukážku pre generovanie do mriežky 2×2 vidno na Obr. 3.8.



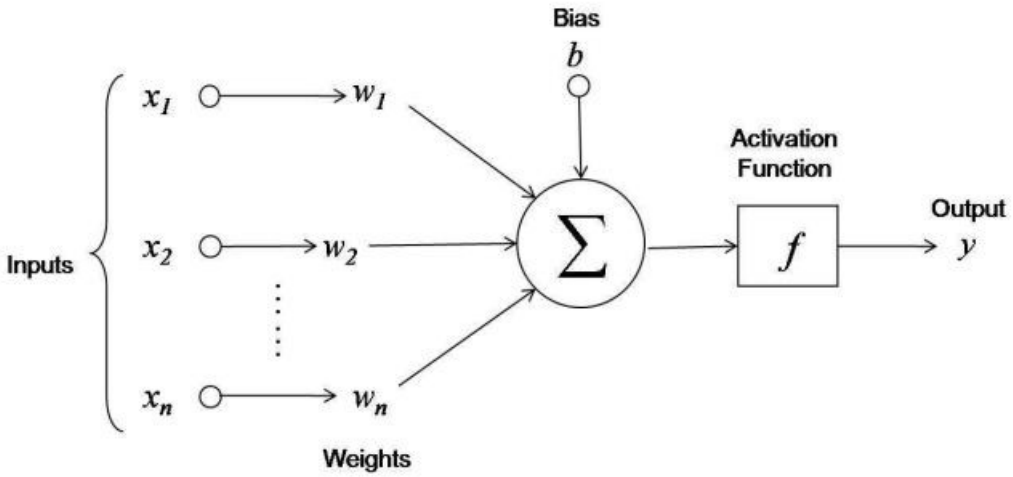
Obr. 3.8: Generovanie SIFT deskriptoru z lokálnych gradientov do mriežky s rozmermi 2×2 [37]

3.5 Viacvrstvové neurónové siete

V kapitole 2 sme sa dozvedeli, že neurónové siete sú silný a momentálne veľmi preferovaný výpočtový model pre segmentačné úlohy v oblasti počítačového videnia. Z tohto dôvodu sme sa aj my, v rámci experimentálnej časti práce, zamerali aj na riešenia

využívajúce hlboké neurónové siete. V tejto časti si priblížime fungovanie dopredných neurónových sietí. Následne sa pozrieme na základné princípy konvolučných neurónových sietí a tiež autoenkóderov. Ako sme naznačili v úvode, cieľom týchto pasáží nie je čitateľa detailne naučiť problematiku hlbokých neurónových sietí, ale skôr si ujednotiť základné pojmy a pripomenúť si princípy metód, ktoré v ďalších kapitolách využijeme v experimentoch testovaných na našom datasete.

V rámci popisovania perceptrónu a viacvrstvových neurónových sietí sme čerpali z [41]. Umelá neurónová sieť je model, ktorý na istej úrovni abstrakcie simuluje činnosť ľudského mozgu. Ten je zložený z obrovského množstva neurónov, ktoré sú vzájomne poprepájané a pomocou elektrických impulzov si medzi sebou odovzdávajú signály. Každý neurón v mozgu tento signál trochu transformuje. Podobne to teda funguje aj v umelých neurónových sieťach [42]. V tých pracujeme s umelým, resp. matematickým modelom neurónu (viď obrázok 3.9).



Obr. 3.9: Jednoduchý perceptrón má n binárnych vstupov a jeden výstup. [44]

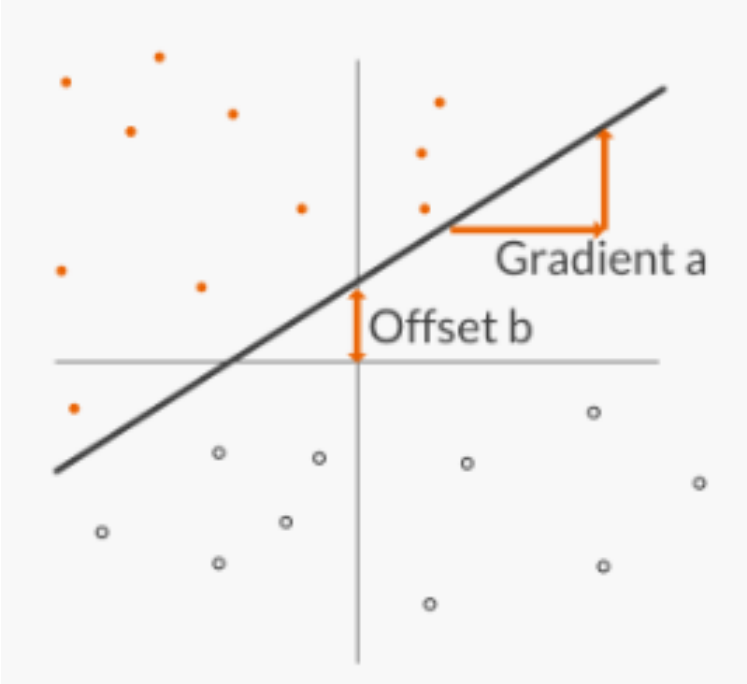
Koncom 50-tych rokov minulého storočia bol Frankom Rosenblattom vyvinutý jednoduchý model neurónovej siete nazývaný **perceptrón** [43]. Perceptrón je špeciálny druh jednovrstvovej neurónovej siete, ktorý bol určený na riešenie binárnej klasifikácie [41]. Na vstupe dostane n hodnôt $x_1, x_2, ..., x_n$. Každý zo vstupov má priradenú váhu w. Váha vstupu je reálne číslo, reprezentujúce dôležitosť danej vstupnej hodnoty pre výsledok. Ako je vidno vo vzorci 3.15, výstupom perceptrónu je hodnota 0 alebo 1 podľa toho, či je súčet súčinov vstupov a ich váh väčší alebo menší ako definovaný prah.

$$vystup = \begin{cases} 0 & \text{ak} \quad \sum_j w_j x_j \leq prah \\ 1 & \text{ak} \quad \sum_{\mathrm{j}} \mathrm{w_j x_j} > \mathrm{prah} \end{cases} \tag{3.15}$$

Perceptrón sa v dnešnej dobe používa v trocha upravenej forme a obsahuje takzvaný potenciál neurónu (známy aj ako **bias** alebo **offset**), čo je doplnkový vstupný neurón s prahovou hodnotou. Po jeho zavedení sa pôvodná funkcia perceptrónu (vzorec 3.15) zapíše ako vzorec 3.16. Potenciál neurónu umožňuje posúvať hranice prahu v priebehu klasifikácie(Obr. 3.10). Bez neho by perceptrón nebol schopný určiť najlepšiu hranicu medzi separovanými dátami.

$$vystup = \begin{cases} 0 & \text{ak} \quad w \cdot x + b \leq 0 \\ 1 & \text{ak} \quad w \cdot x + b > 0 \end{cases} \tag{3.16}$$

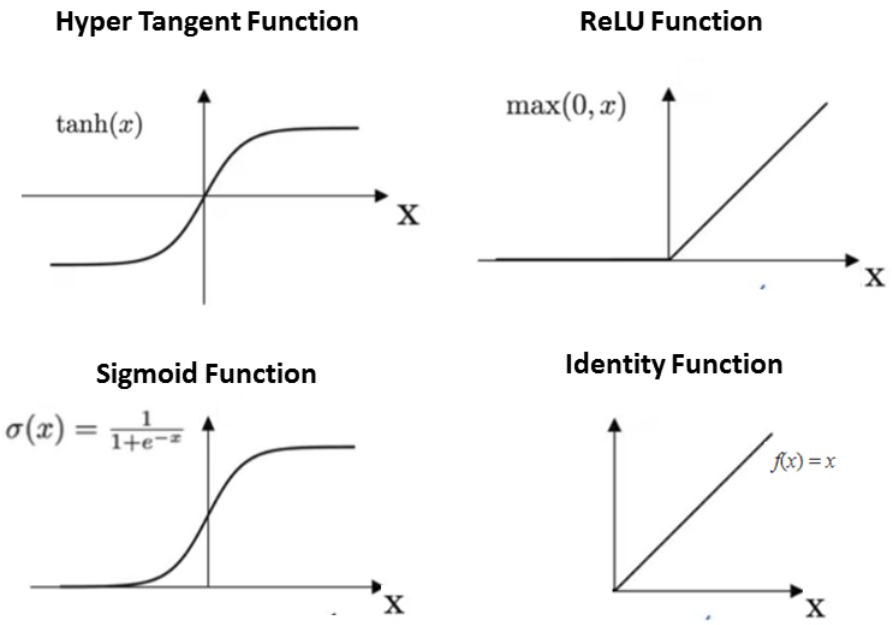
Perceptrón je schopný klasifikovať lineárne separovateľné dáta do dvoch tried na základne jednoduchého rozhodovania. Toto rozhodnutie v neuróne robí takzvaná aktivačná funkcia, ktorej výsledok predstavuje výstup daného neurónu. Využívajú sa však aj iné typy aktivačných funkcií (Sigmoid, Tanh, ReLU,atď.), ktorých ilustráciu môžeme vidieť na obrázku 3.11. Na základe ich použitia sa jednotlivé neuróny navzájom odlišujú.



Obr. 3.10: Ukážka úlohy prahovej hodnoty počas učenia perceptrónu [45].

Spojením viacerých perceprónov sa dá vytvoriť model, ktorý je schopný rozhodovať aj komplikovanejšie úlohy. V podstate sa tak dá simulovať ľubovoľná logická operácia. Vo všeobecnosti môžeme povedať, že umelé neurónové siete sa skladajú zo vstupnej vrstvy, výstupnej vrstvy a minimálne jednej skrytej vrstvy. Model zložený z viacerých

vrstiev neurónov sa nazýva **Viacvrstvová neurónová sieť (MLP - Multi Layer Perceptron)** a ide o najrozšírenejší typ neurónovej siete. V takejto sieti sú neuróny usporiadané do viacerých vrstiev. Vrstiev môže byť v sieti ľubovolne veľa. Každá vrstva má inú úlohu, ktorú potrebuje vyriešiť v rámci celkovej úlohy, ktorú daná sieť rieši. Výstup jednej vrstvy je tak vstupom pre nasledujúcu vrstvu siete. Podľa typu prepojení, ktoré sú použité medzi neurónmi v sieti, rozlišujeme dopredné siete a rekurentné siete.



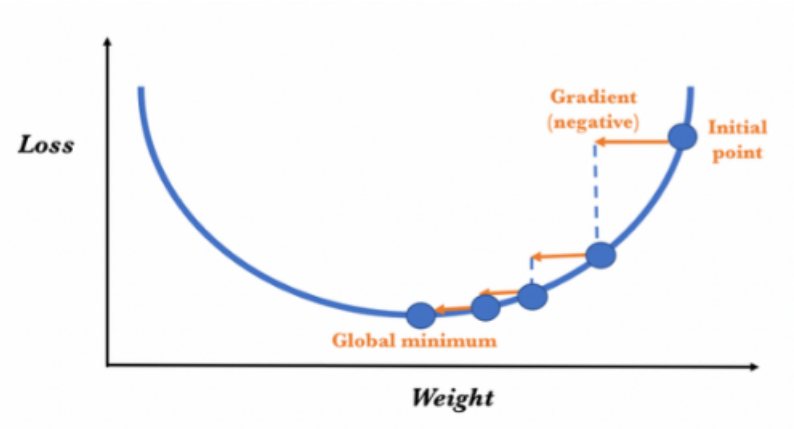
Obr. 3.11: Ukážka typov aktivačných funkcií [62].

- **Dopredné neurónové siete** majú prepojenia iba medzi neurónmi susedných vrstiev, pričom prepojenie je vždy orientované z neurónu v $k-tej$ vrstve do neurónu v $k+1$ vrstve. Architektúra takejto siete sa dá reprezentovať acyklickým orientovaným grafom.

- **Rekurentné neurónové siete** umožňujú prepojenia neurónov v rámci tej istej vrstvy a tiež obojsmerné prepojenia medzi dvoma vrstvami.

Zameriame sa na dopredné neurónové siete, keďže tie sme využili aj v rámci našich experimentov. Naučiť neurónovú sieť riešiť nejaký problém znamená, vhodne určiť parametre jej neurónov (zjednodušene povedané). V prípade jednoduchej siete riešiacej jednoduchú úlohu, akou je napríklad Perceptrón riešiaci binárnu segmentáciu, je správne nastavenie vektora váh pre vektor vstupov otázkou relatívne jednoduchého odladenia. Pri komplexnejších archtektúrach sietí, s väčším množstvom vrstiev neurónov, to je komplikovaná úloha. Pre tento účel existujú učiace algoritmy. Proces učenia je v podstate optimalizačný problém, pri ktorom sa na základe definovaných pravidiel

iteratívne menia parametre neurónov tak, aby sa čo najviac minimalizovala chyba predikcie siete. Na odhadnutie jej chybovosti sa používajú chybové funkcie. Tie určujú na základe zvolenej metriky, ako veľmi vzdialená bola predikcia od reality. Algoritmus spätného šírenia chyby, známy tiež ako **backpropagation** algoritmus, je najčastejšie využívaný algoritmus pre výpočet gradientu vo viacvrstvových dopredných neurónových sieťach. Jedná sa o základnú metódu, ktorá sa používa pri trénovaní neurónovej siete spôsobom učenia s učiteľom. Tento spôsob je pre naše účely relevantný, kedže náš dataset má k trénovacím obrázkom vodomerov aj údaje o súradniciach oblasti, ktorá na týchto obrázkoch má byť detegovaná. Algoritmus najprv inicializuje parametre siete [46]. Následne vezme sadu dát určenú na trénovanie siete a urobí na nej prvú predikciu. Potom spočíta rozdiel medzi predikciou a realitou pomocou zvolenej chybovej funkcie. Následne začne optimalizačný algoritmus spätným šírením modifikovať parametre všetkých neurónov siete na základe spočítanej chyby. Podstatnou časťou tohto učiaceho procesu je **optimalizačný algoritmus**, ktorý má na starosti upravovanie parametrov. Jeden z najbežnejších optimalizačných algoritmov je algoritmus klesania podľa gradientu (**gradient descent**), pozri Obr.3.12.



Obr. 3.12: Vývoj chyby pomocou optimalizačného algoritmu s klesaním podľa gradientu. [46]

Táto metóda spočíta prvú deriváciu chybovej funkcie (gradient), ktorá sa vypočíta v momente, keď sa dokončil jeden beh predikcie siete na testovacích dátach. Následne sa spätným chodom po vrstvách upravujú parametre neurónov pomocou gradientu. Keďže gradient ukazuje v smere rastu funkcie, tak pre zníženie chyby je potrebné urobiť posun opačným smerom. Veľkosť zmeny váhy je určená vzorcom:

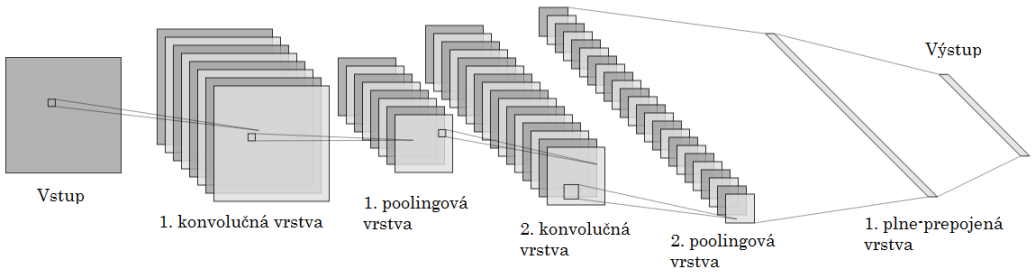
$$w_{ij} = w_{ij} - \alpha \frac{\partial Loss}{\partial w_{ij}} \tag{3.17}$$

kde: $\alpha \in (0, 1)$ je parameter rýchlosti učenia (learning rate) a
$\frac{\partial Loss}{\partial w_{ij}}$ je gradient chyby pre daný neurón.

Algoritmus opakuje zmenu parametrov neurónov (vzorec 3.17) až do bodu, kedy sa nevie posunúť na nižšiu hodnotu chyby (viď obrázok 3.12) [46]. Poznáme viacero ďalších optimalizačných algoritmov (Momentum, RMSProp, Adam, ...), ktoré vo svojom základe využívajú princíp klesania podľa gradientu. Väčšina z nich sú inžinierskými vylepšeniami pôvodného algoritmu za účelom zefektívniť hľadanie minima chybovej funkcie [47].

3.6 Konvolučné neurónové siete

Vo svojej podstate sú konvolučné neurónové siete iba špeciálnym typom viacvrstvovej neurónovej siete. Hlavným rozdielom oproti štandardným „MLP“ sieťam je ich štruktúra. Pomocou konvolučných sietí sa v posledných rokoch darí dosahovať stále lepšie výsledky v oblasti počítačového videnia. Predovšetkým pri segmentácii obrazu či detekcii objektov na obraze sú konvolučné siete vo väčšine prípadov voľbou číslo jeden. Vďaka ich zameraniu na učenie sa a extrahovanie príznakov z obrazu sa využívajú vo viacerých odvetviach, v rámci ktorých spracovávajú obrázky, ale aj video záznamy. Pri farebných obrázkoch, ako to bude aj v našom prípade, dostane sieť na vstupe dvoj-rozmerné pole obsahujúce farebný kanál obrázka. Ako môže byť vidieť aj na obrázku 3.13, konvolučná neurónová sieť má architektúru, ktorá pozostáva z blokov. Podľa [51] sa môžu bloky konvolučnej neurónovej siete skladať z konvolučných vrstiev, nelineárnych vrstiev a z poolingových vrstiev. Na konci, za týmito blokmi, sa zvykne nachádzať ešte jedna alebo viacero plne-prepojených vrstiev. Základným prv-



Obr. 3.13: Architektúra konvolučnej neurónovej siete.

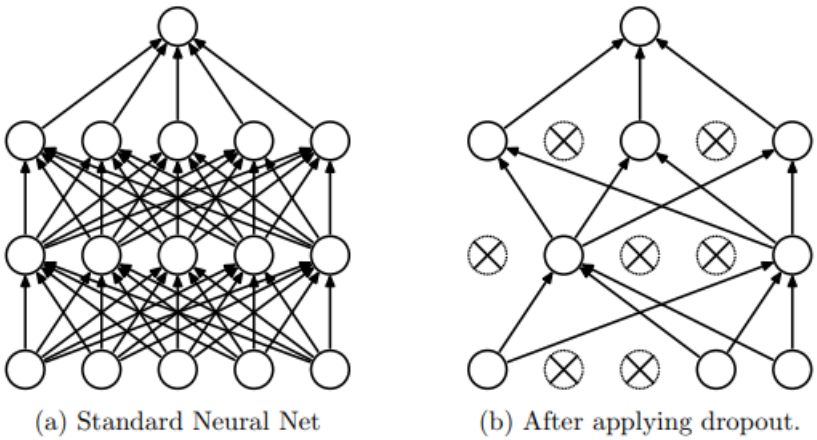
kom tejto architektúry je **konvolučná vrstva**. Každá konvolučná vrstva aplikuje na svoj vstup nejakú funkciu, podobne ako pri bežných sieťach, aby z obrázka extrahovala určitý typ príznakov a poslala ho na vstup pre ďalšiu vrstvu. Každým takýmto kro-

kom sa zložitosť obrázka znižuje. V práci [48] z ktorej sme čerpali, sú popísané základné typy vrstiev konvolučných sietí. Konvolučná vrstva sa skladá zo sady trénovateľných filtrov, pričom každý filter má za úlohu extrahovať zo vstupu nejaký typ informácie (hrany, rohy, oblúky, ...). V rámci filtrov v konvolučných vrstvách zdieľajú neuróny jedného filtra rovnakú množinu váh. Ich výsledkom na výstupe sú extrahované príznaky obrázka, ktoré sa uložia do takzvanej príznakovej mapy. Vďaka zdielaniu váh medzi neurónmi jedného filtra sú konvolučné siete výrazne efektívnejšie ako klasické viacvrstvové neurónové siete. Na obrázku 3.13 s architektúrou siete to zaznačené nie je, avšak je konvencia, že za každou konvolučnou vrstvou sa nachádza **nelineárna vrstva** (aktivačná vrstva). Cieľom je do modelu, ktorý na úrovni konvolučných vrstiev počíta lineárne operácie, zaviesť nelinearitu. Dnes sa najčastejšie za týmto účelom používa ReLU funkcia. Tá má oproti iným funkciám (Sigmoid, Tanh) výhodu, že sieť sa s jej použitím dokáže natrénovať výrazne rýchlejšie, než s inými typmi vrstiev a navyše bez výraznejšej straty presnosti [49]. Svoju efektívnosť dosahuje aj vďaka jednoduchosti jej výpočtu(vzorec 3.18). Dosahuje tiež lepšie výsledky, čo sa týka problému miznúceho gradientu. Tento problém sa prejavuje pomalým trénovaním nižších vrstiev siete, čo sa deje v prípade, keď sa gradient exponenciálne zmenšuje. Ako je to uvedené aj v zdroji [50], ReLu funkcia aktivuje len neuróny, ktoré nemajú negatívnu hodnotu. Vďaka tomu ostane veľa neurónov v sieti neaktívnych. Takýto prístup sa podobá fungovaniu biologickej neurónovej siete v ľudskom tele. Aj tam sa pri rôznych podnetoch aktivujú len niektoré neuróny. Vďaka tejto vlastnosti ReLU funkcie je neurónová sieť schopná sa naučiť rozlišovať oblasti na obraze podľa ich významu.

$$f(x) = max(0, x) \tag{3.18}$$

ReLU funkcia iba premapuje záporné aktivačné hodnoty zo vstupu na 0. Základná myšlienka architektúry konvolučnej siete vo všeobecnosti spočíva v striedaní konvolučných a poolingových vrstiev. Princíp poolingových vrstiev spočíva v redukcii veľkosti vstupu. Vstupná matica sa rozdelí na menšie, neprekrývajúce sa štvorcové podoblasti. A z každej takejto podoblasti sa extrahuje jedna hodnota, ktorá túto podoblasť bude reprezentovať. Najčastejšie sa používa max-poolingová vrstva, ktorá z danej podoblasti vyberie maximálnu hodnotu a iba tú posiela na výstup. Poolingové vrstvy zabezpečujú znižovanie počtu parametrov, čo výrazne zlepšuje výpočtovú zložitosť modelu a tiež redukujú efekt pretrénovania siete. Dôležitú úlohu pri znižovaní efektu pretrénovania konvolučnej siete zohrávajú **dropout** vrstvy. V článku [63] autor popisuje ich základný princíp. Dropout vrstvy počas trénovania siete odpájajú jednotlivé neuróny. To znamená, že pri doprednom aj spätnom prechode cez sieť je každý neurón s pravdepodobnosťou $1 - p$ odpojený od siete (a s ním aj všetky jeho prepojenia na ostatné neuróny) a s pravdepodobnosťou p je tento neurón v sieti ponechaný. Použitím dropout vrstiev síce narastie počet epoch potrebných na skonvergovanie siete, ale trénovací čas

každej epochy sa vďaka zredukovanému počtu neurónov skráti(obrázok 3.14).



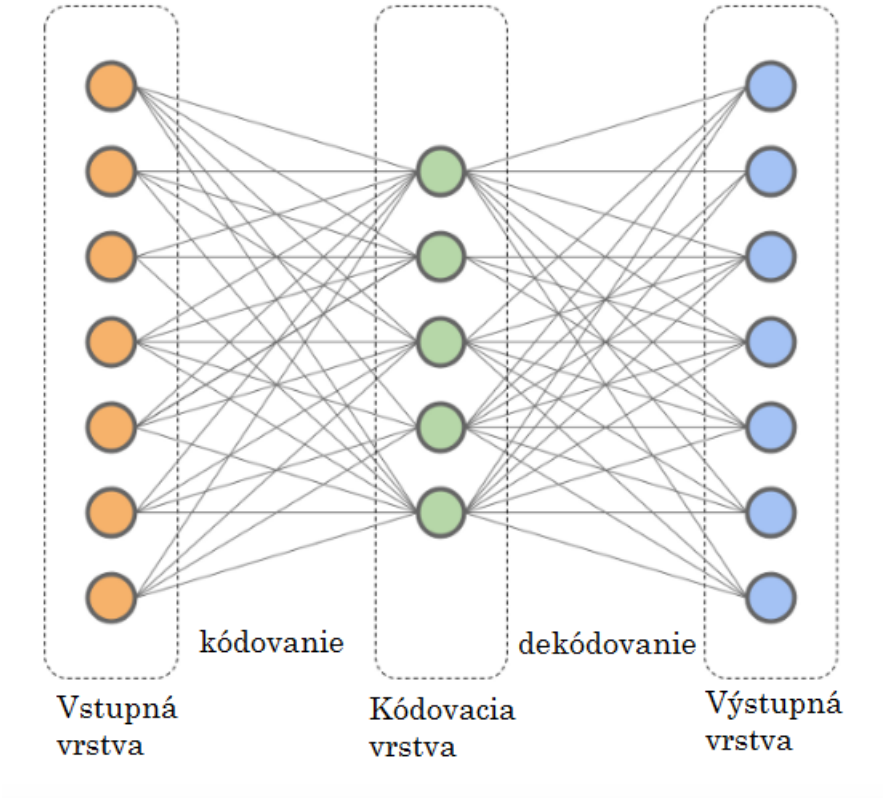
Obr. 3.14: Efekt aplikovania dropout vrstvy v sieti. [64]

Na záver pred výstupom má konvolučná neurónová sieť ešte niekoľko plne-prepojených (**fully-connected**) vrstiev. Tie sú typovo úplne zhodné s vrstvami štandardnej viac-vrstvovej siete, pričom každý neurón $k-1$ vrstvy je prepojený s každým neurónom k-tej vrstvy. Táto vrstva reprezentuje vektor príznakov, ktorý sa ďalej používa buď na klasifikáciu, regresiu alebo ako vstup pre ďalšiu sieť. Čerpané z [51].

3.7 Autoenkódery

Špeciálnym typom umelej neurónovej siete sú autoenkódery.Tie sa skladajú z troch vrstiev (pozri Obr. 3.15). Autoenkóder sa skladá zo vstupnej vrstvy, kódovacej vrstvy a výstupnej vrstvy. Myšlienkou autoenkóderov je zakódovať vstupné dáta do kódovacej vrstvy, ktorá má menšie rozmery ako vstupná a výstupná vrstva. Následne je vstup rekonštruovaný (dekódovaný) na výstupnú vrstvu.

Pretože sa na strednej vrstve redukuje počet neurónov, vzniká takzvaný „bottleneck“ efekt, vďaka ktorému je sieť nútená efektívne skomprimovať informácie o vstupe. Sieť hľadá korelácie medzi príznakmi vstupného obrázka. Takto detegované štruktúry sa sieť dokáže naučiť a na výstupe ich rekonštruovať. Podľa [53] boli autoenkódery pôvodne využívané na redukciu dimenzie, či učenie sa príznakov v obraze. Dnes môžeme vnímať autoenkódery ako špeciálny typ dopredných neurónových sietí, ktoré sú trénované obdobnými technikami. Existuje však aj iná trénovacia technika, nazývaná **recirkulácia**, ktorá sa využíva pri trénovaní autoenkóderov. Táto technika má v porovnaní s trénovacím algoritmom spätného šírenia chyby omnoho bližšie k biologickým princípom učenia sa, avšak zriedkavo je využívaná pre účely strojového učenia. Štandardný autoenkóder má plne-prepojené vrstvy tak, ako tomu bolo aj v MLP modeloch. Pre spracovanie



Obr. 3.15: Architektúra Autoenkódera. [52]

obrazu sa tak musí obrázok uložiť do vektora, aby následne mohol byť použitý na vstupe štandardného autoenkódera. Touto konverziou sa však stráca veľké množstvo informácie potrebnej pre extrahovanie príznakov daného snímku. Tento problém rieši **Konvolučný autoenkóder**, ktorý vo fáze kódovania vstupu využíva konvolučné a poolingové vrstvy na extrahovanie príznakov obrázka. Pre rekonštrukciu zakódovaného vstupu je potom potrebná dekonvolúcia. Tento proces funguje na princípe, ako sme si ho už popísali v sekcii 2.2.2.

Kapitola 4

Testované algoritmy a dataset

V predošlých kapitolách 2 a 3 sme sa oboznámili s problematikou segmentácie obrazu a detekcie objektov na obraze. Ukázali sme si niekoľko populárnych prístupov riešiacich úlohy z tejto oblasti a vysvetlili si základné princípy ich fungovania. V tejto kapitole sa zameriame na experimentálnu časť našej práce. Experimenty, ktoré sme skúšali, sa zameriavali na nájdenie optimálneho prístupu pre detekciu ciferníkov na vodomeroch. Zoznámime sa s datasetom, na ktorom sme trénovali viaceré modely a testovali na ňom naše algoritmy. Jednotlivé prístupy si teraz ukážeme a detailnejšie popíšeme.

4.1 Dataset

Dataset, ktorý sme využívali, vznikol pre potreby riešenia automatizácie odpočtu stavu vodomerov, ktorý sme v krátkosti opísali v kapitole 1. Dataset bol pre účely tejto práce zapožičaný firmou Cognexa, ktorá sa venuje riešeniam na báze umelej inteligencie pre potreby automatizácie a optimalizácie procesov. Ako už teda bolo spomenuté, dataset obsahuje telefónom odfotené snímky vodomerov v prirodzenom prostredí. Obrázky z prirodzeného prostredia predstavujú špeciálnu skupinu dát, ktorá sa vyznačuje veľkou variabilitou svojich parametrov. To spôsobuje komplikácie pre algoritmy, ktoré ich majú spracovávať. Jedná sa o to, že obrázky zhotovené v teréne, môžu trpieť veľkým množstvom nedokonalostí ako napríklad: slabý jas fotografie alebo prílišné presvetlenie obrázka, rozmazanie, priveľká vzdialenosť od objektu, zlá perspektíva a mnohé ďalšie. Všetky tieto faktory sa môžu, ale aj nemusia objaviť na obrázku a algoritmus, ktorý takýto obrázok spracováva, by mal byť ideálne na všetky tieto faktory rezistentný. Problému segmentácie prirodzených obrázkov sa v posledných rokoch venuje pomerne vysoká pozornosť. Existuje množstvo algoritmov, ktoré sa zapájajú do súťaží (Kaggle, ILSVRC, a iné) s cieľom, čo najlepšie segmentovať obrázky verejne prístupných datasetov. Takéto datasety obsahujú desiatky tisícov prirodzených obrázkov zobrazujúcich rôzne objekty v rôznych prostrediach. Najlepšie algoritmy sú na týchto datasetoch

schopné s vysokou úspešnosťou detegovať objekty ako: auto, človek, pes, mačka, atď. (viď obrázok 4.1)

Obr. 4.1: Ukážka segmentovaných objektov na prirodzenom obrázku [60].

Jeden z najznámejších datasetov je **ImageNet** [61], ktorý obsahuje viac ako 14 miliónov prirodzených obrázkov zobrazujúcich objekty, ktoré možno klasifikovať do viac ako 21 tisíc kategórií. Takéto množstvo dát dáva dobrý základ komplexným riešeniam využívajúcim hlboké neurónove siete. Náš dataset obsahuje takmer 1500 anotovaných snímkov vodomerov, ktoré sa snažia čo najvernejšie demonštrovať podmienky, za akých by v praxi tieto snímky vznikali (viď obrázok 4.2). Anotácia snímkov predstavuje osmicu čísel, ktoré reprezentujú súradnice rohov hľadaného orámovania ciferníka na vodomeri. Pre účely trénovania modelov boli obrázky datasetu v rámci predspracovania preškálované na rozmery 600×600 pixelov.

Obr. 4.2: Ukážka obrázkov z datasetu vodomerov s dobre identifikovateľným ciferníkom.

V porovnaní s veľkými datasetmi ako je napríklad ImageNet, je veľkosť datasetu vodomerov zanedbateľná a preto je otázne, či takéto množstvo trénovacích obrázkov bude dostatočné pre natrénovanie spoľahlivého segmentačného modelu. Jednou z výhod

v našom prípade je fakt, že na obrázku je potrebné segmentovať iba jediný objekt, resp. oblasť (okolie ciferníka vodomeru). Detegovanie tejto plochy je však pomerne nejednoznačná úloha.

4.1.1 Augmentácie

Veľmi užitočným v boji s limitáciou, akou je malý dataset, sa ukázalo použitie augmentácií. Augmentácia datasetu je technika, ktorá pomáha rozšíriť dataset o nové obrázky. Táto metóda sa využíva práve v prípadoch, keď máme pre účely trénovania modelu k dispozícii iba malé množstvo dát. Nové obrázky vzniknú aplikovaním rôznych transformácií a deformácií pôvodných snímkov. Na obrázku 4.3 môžeme vidieť ukážku deformovaných snímkov z datasetu. Takéto rozšírenie trénovacieho datasetu môže pomôcť pri generalizácii klasifikácie vstupných dát. Rozlišujeme dva typy procesov augmentácie podľa spôsobu ich aplikácie na dataset. V prípade **offline augmentácie** sú transformácie na dáta aplikované ešte pred samotným trénovaním modelu. Pri **online augmentácii** sú transformácie aplikované na obrázky v jednotlivých dávkach počas ich trénovania. V prípade datasetu s vodomermi bola použitá offline augmentácia, v rámci ktorej boli využité transformácie rôznych typov (rozostrovanie, zmena jasu, zašumenie obrázka, preškálovanie, zmena perspektívy, zmena kontrastu, rotácia).

Obr. 4.3: Ukážka snímkov z datasetu vodomerov so slabšie čitateľným stavom.

4.2 Morfologická segmentácia

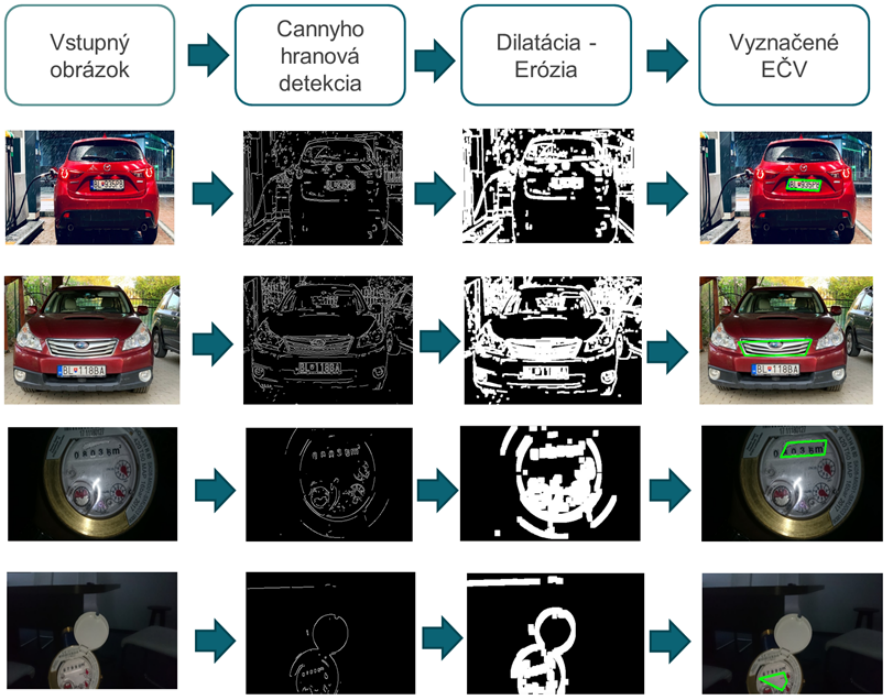
Prvým z testovaných prístupov je algoritmus využívajúci morfologické operácie pre detegovanie kontúr na obrázku. Inšpiráciou nám boli pomerne jednoduché implementácie algoritmov na detekciu tabuľky s EČV na fotografiách s automobilmi. Ako sa aj neskôr ukáže, kľúčovým bodom našich experimentov je vhodná interpretácia hľadanej

oblasti na obrázku. Morfologické operácie, ako sme sa aj dozvedeli v kapitolách 3.2.4 a 3.2.5, pracujú predovšetkým s tvarmi a obrysmi objektov na obrázku a podľa nich sú aj detegované. Z tohto hľadiska je definícia ciferníka na vodomeri jednoduchá, lebo sa jedná o obdĺžnik vyplnený číslicami. Podobnosť s problémom detekcie EČV je z morfologického hľadiska zjavná.

Algoritmus teda bude na obrázku detegovať signifikantné obdĺžnikové kontúry. Skôr, ako sa však pustíme do detekcie kontúr na obrázku, je potrebné vstupný obrázok vhodne predspracovať. Náš algoritmus sa skladá z nasledujúcich krokov:

1. Obrázok transformujeme do šedotónovej podoby.
2. Na šedotónový obrázok aplikujeme Gaussov filter pre vyhladenie šumu.
3. Detegujeme hrany pomocou Cannyho detektora 3.2.3.
4. Na binárny obrázok s detegovanými hranami aplikujeme štandardné morfologické operácie, akými sú dilatácia a erózia. Veľkosťou kernela a tiež počtom opakovaní jednotlivých operácií vieme z obrázka extrahovať kontúry, v ktorých sa číslice na EČV ale aj na ciferníku vodomera zlejú do jedného obdĺžnika, ktorý tak vieme v ďalšom kroku nájsť.
5. Hľadáme kontúry na obrázku. Obrys výslednej kontúry sa musí skladať zo štyroch vrcholov a kontúra musí mať veľkosť väčšiu ako definovaný prah.

Vďaka fixným parametrom sa dá očakávať, že algoritmus bude veľmi citlivý na zmeny vstupných obrázkov. Na obrázku 4.4 môžeme vidieť jednotlivé kroky spracovania vstupných obrázkov algoritmom. Dá sa všimnúť, že vo všetkých prípadoch, bez ohľadu na to, či je na vstupe automobil s EČV alebo vodomer s ciferníkom, algoritmus začína mať problémy, ak sa na obrázku vyskytne aj iná kontúra spĺňajúca jednoduché kritériá štyroch vrcholov a dostatočnej veľkosti. Môžeme však tento algoritmus použiť ako základný výsledok pri vyhodnocovaní algoritmov nevyužívajúcich hlboké učenie.



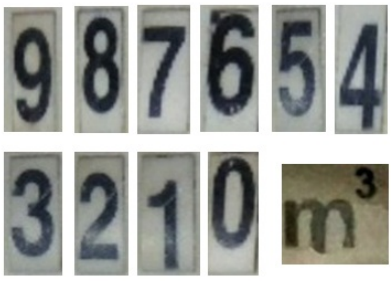
Obr. 4.4: Porovnanie výsledkov algoritmu morfologickej detekcie na obrázkoch s vozidlami a obrázkoch s vodomermi.

4.3 Hľadanie zhody v obraze

V tejto časti si ukážeme niekoľko prístupov, ktoré k segmentácii ciferníka pristupujú prostredníctvom hľadania zhody medzi vzorovým a referenčným obrázkom. Pre tieto účely sme využili poznatky, ktoré sme do veľkej miery popísali v kapitolách 3.3 a 3.4. V kapitole 3.3 sme si objasnili princípy takzveného Naivného Template matchingu. Ten porovnáva vzorový obrázok na všetkých možných pozíciách referenčného obrázka a pri každom porovnaní vypočíta zhodu podľa zvolenej korelačnej funkcie. Hlavným problémom sa ukázalo, že takýto algoritmus nie je invariantný na škálovanie, rotáciu, či skosenie. Z toho vyplýva, že zhodu medzi obrázkami je algoritmus schopný presne detegovať len v prípade, že hľadaná scéna vzorového obrázka sa na referenčnom obrázku nachádza v podobnej veľkosti, pod podobným sklonom aj natočením. Takýto prístup je pochopiteľne v podmienkach prirodzených fotografií trocha neuplatniteľný vzhľadom k ich variabilite spomenutých parametrov. Skôr, ako si však povieme aj o jeho použiteľnejších implementáciách, využijeme túto jednoduchú implementáciu v kombinácii s menej náročnými obrázkami datasetu na odladenie správnej voľby vzorového obrázka. Ako sa ukázalo aj v predchádzajúcom prístupe popísanom v kapitole 4.2, správne a pre algoritmus zreteľné definovanie hľadanej oblasti sa ukazuje ako kľučový faktor segmentačnej úlohy, ktorú riešime. Naskytuje sa nám niekoľko variánt, na ktorých je

možné postaviť logiku nášho algoritmu. Pozrime sa teda na dva spôsoby, akými možno k výberu vzorového obrázka pristúpiť:

Každý znak samostatne
Jeden prístup je, že môžeme na referenčnom obrázku s vodomerom hľadať jednotlivé číslice ciferníka, prípadne symbol mierky m^3, ktorý je na ciferníku predtlačený (viď obrázok 4.5), takže máme istotu, že by na obrázku mal byť detegovaný.

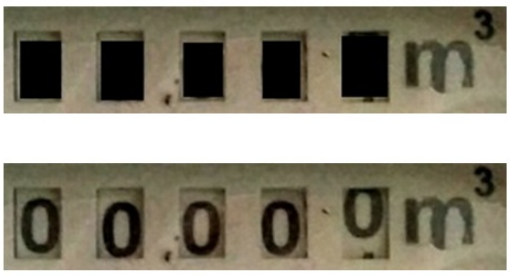


Obr. 4.5: Variant vzorového obrázka so samostatnými znakmi.

Takýto prístup so sebou prináša viacero komplikácií. Prvou je fakt, že nevieme, ktoré číslice a v akom počte budú na ciferníku v momente zhotovenia obrázka zobrazené. Ďalšou komplikáciou je prítomnosť aj iných číslic na vodomeri, ktoré napríklad popisujú model vodomeru, nehovoriac o potenciálnych čísliciach nachádzajúcich sa na nejakých objektoch v okolí vodomeru. Tretím potenciálnym problémom, alebo skôr komplikáciou by mohlo byť skladanie výsledného orámovania ciferníku na základe detekcie jednotlivých čísel.

Ciferník ako celok
Druhý spôsob je zamerať sa na ciferník v celku, tak ako je to zobrazené na obrázku 4.6. Tento prístup by automaticky odbúral problém s usporadúvaním a prepájaním rámikov jednotlivých znakov. Aj tu sa však nájdu potenciálne problémy, ktorých validnosť sa prejaví až pri testoch na datasete. Jeden problém môže spôsobiť fakt, že rôzne vodomery v momente zhotovenia fotografie zobrazujú rôzne hodnoty. Pri porovnaní s jedným vzorom, ktorý prirodzene zobrazuje na ciferníku len jeden stav (viď obrázok 4.6), dochádza k rozdielnosti obrázkov na pozíciách číslic. To samozrejme do značnej miery ovplyvní hodnotu korelácie medzi porovnávanými ciferníkmi. Riešením by mohlo byť použitie takzvaného „deravého“ ciferníku, tak ako je zobrazený na obrázku 4.6.

Implementovali sme ešte dva ďalšie algoritmy fungujúce na princípe template matchingu, ktoré by sa mali aspoň čiastočne vysporiadať s nesprávnym škálovaním alebo rotáciou medzi porovnávanými obrázkami. Tieto dva algoritmy sme pomenovali „RS-



Obr. 4.6: Variant vzorového obrázka s kompletným ciferníkom vodomeru.

Invariant Template matching“ a „S-Invariant Template matching“. S-Invariantný algoritmus funguje jednoduchým spôsobom, v ktorom je aplikovaná funkcia template matchingu pre viacero úrovní priblíženia referenčného obrázka. RS-Invariantný algoritmus funguje rovnako ako S-Invariantný s tým rozdielom, že okrem viacerých možných škálovaní referenčného obrázka aplikuje aj viaceré možné rotácie. V tomto prípade je pre každý testovaný obrázok potrebné urobiť $n * m$ behov template matchingu, kde n je počet aplikovaných škálovaní na obrázok a m je počet aplikovaných rotácií obrázku. Takýmto spôsobom sa nám enormne zvýšila časová náročnosť algoritmu. Oba algoritmy predstavujú takzvaný brute force prístup. Z dôvodu časovej zložitosti je preto podstatné zvoliť len prijateľné množstvo škálovaní a rotácií.

V kapitole 3.4 sme popisovali prístup takzvaného Feature matchingu, teda algoritmu, ktorý taktiež hľadá zhodu medzi vzorovým a referenčným obrázkom. Rozdiel oproti dvom metódam, ktoré sme doteraz popisovali v tejto kapitole, spočíva v spôsobe hľadania zhody dvoch obrázkov. Algoritmus Feature matchingu si najprv na vzorovom obrázku definuje kľúčové body a tie sa potom snaží detegovať na referenčnom obrázku. Náš zámer detegovať ciferník vodomeru je teda závislý od počtu nájdených kľučových bodov. Aby sme boli schopní vyznačiť na obrázku oblasť ciferníka vodomeru na základe detegovaných bodov, je potrebné, aby algoritmus našiel aspoň 4 kľúčové body, ktoré by definovali ohraničujúci obdĺžnik. Pre všetky výsledky s menším počtom nájdených kľúčových bodov ako je štyri, prehlásime výsledok za neúspešný. V tomto prípade ako vzorový obrázok použijeme iba varianty s kompletným ciferníkom.

4.4 Predikcia rohových súradníc

Ďalším prístupom, ktorý si teraz predstavíme a ktorý sme testovali, sa posunieme do skupiny algoritmov, ktoré sú postavené na princípoch hlbokého učenia. S detailami problematiky, ktorú okolo vodomerov riešime, sme boli v tejto kapitole už viackrát oboznámení a preto sa rovno pozrime na ideu tohto nového prístupu. Hlavnou myš-

lienkou bude v tomto prípade určiť súradnice rohov oblasti, ktorá zobrazuje časť vodomeru s ciferníkom. Pre účely otestovania tohto prístupu sme použili dva typy neurónovej siete, na ktorých implementáciu sme využili knižnicu Tensorflow [54]. V prvom prípade sme implementovali menšiu viacvrstvovú perceptrónovú sieť. Druhý spôsob využíval konvolučnú neurónovú sieť. V oboch prípadoch majú modely predikovať osmicu čísel $(A_x, A_y, B_x, B_y, C_x, C_y, D_x, D_y)$, ktoré predstavujú súradnice rohových bodov A, B, C, D v 2D matici obrázka. Rovnakým spôsobom sú anotované aj obrázky v datasete s vodomermi. Pre natrénovanie oboch sietí sme použili približne 65% trénovacích obrázkov z datasetu s vodomermi, ktoré sme preškálovali na rozmery 100×100 pixelov.

Viacvrstvová neurónová sieť

Takýto typ siete sa dnes už síce na spracovanie obrazu často nepoužíva, ale predikcia súradnic bodov by na tento problém mohla fungovať. Pokúsili sme sa preto natrénovať model, ktorý sme optimalizovali algoritmom Adam s využitím chybovej funkcie MSE (Mean Square Error). Pred začatím trénovania bolo potrebné vstupné obrázky vhodne predspracovať, aby boli použiteľné ako vstup pre viacvrstvovú neurónovú sieť. Po tom, ako sme obrázky transformovali na šedotónové a preškálovali na rozmer 100×100 pixelov, sme ich pre potreby siete transformovali na vektor dĺžky 10000, ktorý sme posielali sieti na vstup. Táto sieť sa skladá zo siedmich plne-prepojených neurónových vrstiev. Detailnejší popis architektúry siete môžeme vidieť v tabuľke 4.1.

Poradie vrstvy	Typ vrstvy (aktivačná funkcia)	Popis vrstvy
1.	Plne-prepojená (Sigmoid)	vstupná vrstva s rozmermi 1x4000
2.	Plne-prepojená (Sigmoid)	rozmery vrstvy 1x1000
3.	Plne-prepojená (Sigmoid)	rozmery vrstvy 1x624
4.	Plne-prepojená (ReLU)	rozmery vrstvy 1x312
5.	Plne-prepojená (ReLU)	rozmery vrstvy 1x128
6.	Plne-prepojená (ReLU)	rozmery vrstvy 1x64
7.	Plne-prepojená (Sigmoid)	výstupná vrstva s rozmermi 1x8

Tabuľka 4.1: Architektúra viacvrstvovej neurónovej siete predikujúcej súradnice rohov ciferníka.

Konvolučná neurónová sieť

Konvolučné siete sú pre prácu s obrázkami vhodnejšou voľbou, čo vyplýva z ich schopnosti extrahovať z obrázka podstatné príznaky. Pre účely predikcie súradníc a pre porovnanie s MLP-riešením sme si zostrojili konvolučnú neurónovú sieť, ktorá na vstupe dostane trojkanálové obrázky s veľkosťou 100×100 pixelov. Na

rozdiel od MLP siete, nie je potrebné vstupné obrázky transformovať do šedotónovej podoby. V tabuľke 4.3 môžeme vidieť, že prvé dve vrstvy siete sú konvolučné vrstvy s 32 a 64 filtrami, všetky s rozmermi 5×5. Aktivačná funkcia oboch vrstiev je ReLU. Po konvolučných vrstvách je použitá jedna Max-poolingová vrstva s maticou veľkosti 2×2. Za ňou nasleduje špeciálna dropout vrstva, ktorú sme už detailnejšie popísali v kapitole 3.6. Táto vrstva v sieti pomáha, aby sa model počas trénovania nepretrénoval. Čize, aby sa príliš nenaviazal na danú množinu vstupov a stratil tak schopnosť spracovávať nové dáta. V prípade dropoutu sa jednotlivé neuróny predchádzajúcej vrstvy vypínajú s určitou pravdepodobnosťou p. V prípade nášho modelu sa $p = 25\%$. Po dropout vrstve sú zapojené ešte štyri plne-prepojené neurónové vrstvy tak, ako to je uvedené v tabuľke 4.3. Pri trénovaní konvolučnej siete sme tiež využili optimalizačný algoritmus Adam s chybovou funkciou Categorical Cross-Entropy.

Poradie vrstvy	Typ vrstvy (aktivačná funkcia)	Popis vrstvy
1.	Konvolučná vrstva (ReLU)	počet filtrov 32, keras 5×5
2.	Konvolučná vrstva (ReLU)	počet filtrov 64, keras 5×5
3.	MaxPooling	matica 2×2
4.	Dropout	25%
5.	Plne-prepojená (ReLU)	rozmery vrstvy 1×312
6.	Plne-prepojená (ReLU)	rozmery vrstvy 1×128
7.	Plne-prepojená (ReLU)	rozmery vrstvy 1×64
8.	Plne-prepojená (Sigmoid)	výstupná vrstva s rozmermi 1×8

Tabuľka 4.2: Architektúra konvolučnej neurónovej siete predikujúcej súradnice rohov ciferníka.

4.5 Predikcia masky

Pri ďalšom z vyskúšaných prístupov sme sa na problém segmentácie ciferníka vodomeru pozreli na úrovni pixelov. Pri sémantickej segmentácii na úrovni pixelov tak, ako sme ju aj popísali v kapitole 2.2.2, môžeme výslednú segmentáciu obrázkov s vodomermi reprezentovať ako binárnu masku. Na takejto binárnej maske budú všetky pixely, ktoré nereprezentujú hľadanú oblasť s hodnotou 0 (čierna farba) a všetky pixely, ktoré oblasť s ciferníkom reprezentujú, budú s hodnotou 255 (biela farba), tak ako je to zobrazené na obrázku 4.7. Pokúsili sme sa preto natrénovať model, ktorý by takúto masku predikoval. Pre tieto účely sme použili konvolučný autoenkóder, ktorý na vstupe dostane farebný obrázok s rozmermi 600×600 pixelov a na výstupe vráti 2 kanály, z ktorých jeden predstavuje pravdepodobnostnú mapu určujúcu každému pixelu pravdepodob-

nosť, či daný pixel reprezentuje ciferník. Druhý kanál predstavuje pravdepodobnostnú mapu, ktorá pre každý pixel definuje pravdepodobnosť, že daný pixel nie je ciferníkom vodomeru. Pomocou aktivačnej funkcie sigmoid sme z týchto dvoch kanálov vypočítali binárnu masku. Takto predikovaná maska sa následne bit po bite porovná s binárnou maskou vypočítanou z anotovaného obrázka.

binárnu masku aká bola popísana vyššie.

Obr. 4.7: Predikovanie binárnej masky ciferníka vodomeru.

Konvolučný autoenkóder sa skladá z kódovacej časti a dekódovacej časti. Náš autoenkóder má kódovaciu časť zloženú zo šiestich konvolučných vrstiev s kernelom veľkosti 3×3 a troch max-poolingových vrstiev s poolingovým oknom veľkosti 2×2. Detailnejší popis vrstiev a architektúry kódovacej časti je popísaný v tabuľke . Vzhľadom k tomu, že tento prístup preukazoval najlepšiu schopnosť naučiť sa segmentovať na obrázku oblasť s ciferníkom vodomeru, pokúsili sme sa odladiť model autoenkódera tak, aby dosahoval čo najlepšie výsledky. Model sme pri trénovaní optimalizovali pomocou metriky F1, ktorú detailnejšie ešte popíšeme v kapitole 5.1. V princípe však išlo o metriku, ktorá porovnávala binárnu masku skutočnéj polohy ciferníka na fotke s binárnou maskou predikovanej pozície ciferníka. Ukázalo sa, že pridávanie ďalších vrstiev do architektúry siete od istého počtu konvolučných vrstiev v sieti prestalo byť efektívne. Sieť sa trénovala zbytočne pomaly s relatívne malým zlepšovaním na presnosti. Pri trénovaní sme využili optimalizačný algoritmus Adam s hodnotou učiaceho koeficientu (learning rate) = 0.0001. Pri určovaní učiaceho koeficientu sme pre dosiahnutie prijateľnej hodnoty trocha experimentovali. Pri menších hodnotách učiaceho koeficientu sa model trénoval príliš dlho, naopak pri vyšších mal problém skonvergovať.

Trénovanie autoenkóderu sme testovali aj s využitím optimalizačného algoritmu RMSProp. Výsledky však boli porovnateľné, ako pri použití algoritmu Adam. Na implementáciu sme použili opensource knižnice *cxflow* a *cxflow tensorflow* [55].

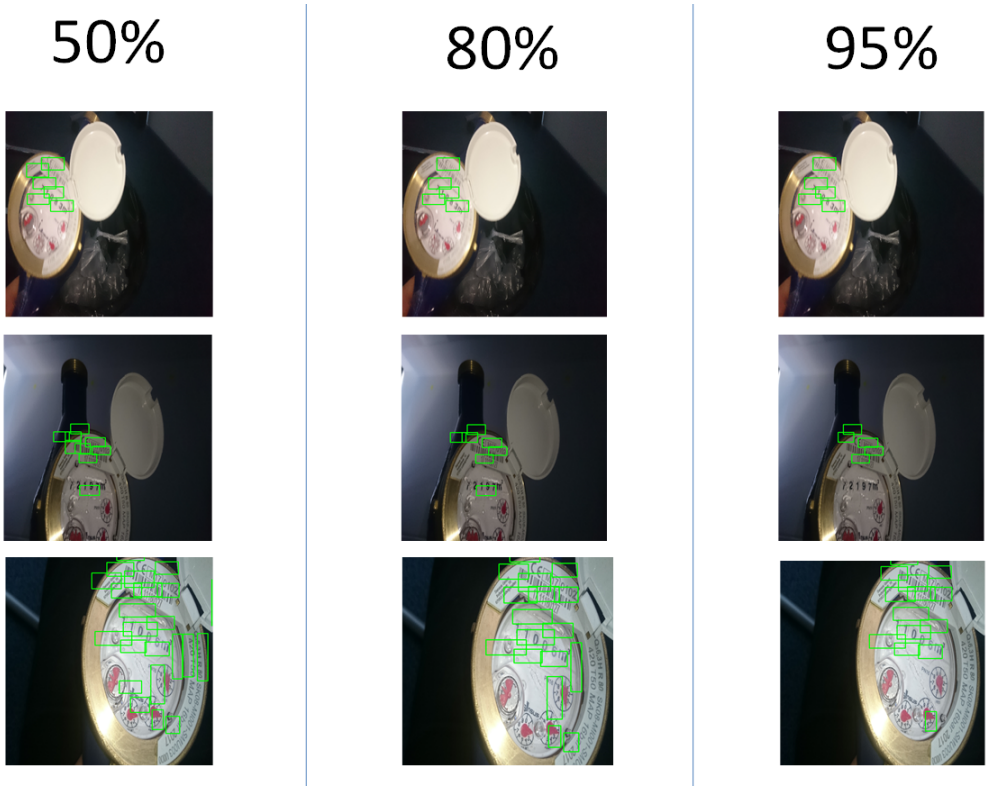
Poradie vrstvy	Typ vrstvy(aktivačná funkcia)	Popis vrstvy
1.	Konvolučná vrstva	počet filtrov 16, keras 3×3
2.	MaxPooling	matica 2×2
3.	Konvolučná vrstva	počet filtrov 24, keras 3×3
4.	MaxPooling	matica 2×2
5.	Konvolučná vrstva	počet filtrov 32, keras 3×3
6.	MaxPooling	matica 2×2
7.	Konvolučná vrstva	počet filtrov 40, keras 3×3
8.	Konvolučná vrstva	počet filtrov 48, keras 3×3
9.	Konvolučná vrstva	počet filtrov 56, keras 3×3

Tabuľka 4.3: Architektúra kódovacej časti konvolučného autoenkódera.

4.6 Detekcia textu v obraze

V ďalšom testovanom prístupe sme sa na problém segmentácie ciferníka vodomeru pokúsili namapovať na problém detekcie textu na obrázku. Ciferník na vodomeri sa skladá z piatich do riadku zarovnaných číslic. Číslice môžeme interpretovať ako znaky, ktoré ako celok možno vnímať ako kus textu na obrázku. Za týmto účelom sme použili predtrénovaný EAST (Efficient and Accurate Scene Text detection pipeline) model, určený na detekciu textu v obraze. Inšpiráciou v tomto prípade bol pre nás článok [57], v ktorom autori popisujú EAST model, ktorý si ešte detailnejšie predstavíme. Algoritmus, ktorý sme implementovali, postupne prechádza testovacie obrázky datasetu a na každom obrázku pomocou EAST modelu, ktorý sme implementovali pomocou knižnice OpenCV [56], deteguje a vyznačí rámikom potenciálnu oblasť s textom. Každá vyznačená oblasť má priradenú pravdepodobnostnú hodnotu p z intervalu (0,1), ktorá určuje pravdepodobnosť, s akou si model myslí, že na danom vyznačenom mieste sa nachádza text. V prípade nášho testovacieho algoritmu sme pôvodne pravdepodobnostný prah nastavili na hodnotu 0.5. V prípade, že model nedetegoval na obrázku žiadnu textovú plochu s pravdepodobnosťou aspoň 50%, pre daný obrázok sme vrátili nulovú zhodu. Avšak v prípade vodomerov, ktoré obsahovali na sebe viaceré textové plochy, bolo treba každý vyznačený rámik ešte dodatočne vyhodnotiť. Na väčšine obrázkov sme totiž dostali viacero odpredikovaných orámovaní textu. Prvým krokom, čo sme skúsili, bolo posunutie prahu z 50% na vyššiu hodnotu. To nám pomohlo zredukovať počet orámovaní, avšak nie tak výrazne, aby nás to zbavilo nutnosti nejakého postprocesingu (pozri obrázok 4.8).

Aby sme však tento prístup mohli vyhodnotiť, bolo potrebné definovať v každom prípade práve jeden rámik. Vyskúšali sme viaceré verzie, ako vyberať kandidáta dete-



Obr. 4.8: Výsledky detekcie textu pri prahoch 50% , 80%, 90%.

govaného ciferníka.

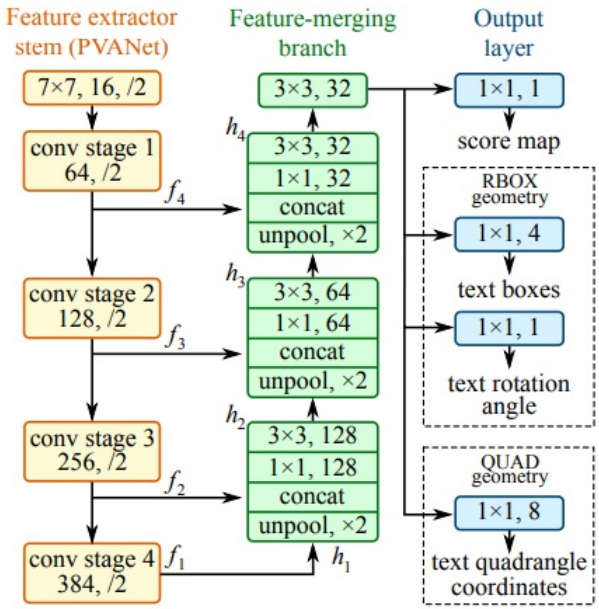
Prvým spôsobom bolo vybrať orámovanie, zahŕňajúce všetky orámovania detegovaných textov s najvyššou hodnotou „confidence“ (>99%). Druhým otestovaným spôsobom identifikácie ciferníka bol výber orámovania na obrázku s najväčšou hodnotou confidence, teda také textové pole, o ktorom je model najviac presvedčený, že sa jedná o text. Vychádzali sme z faktu, že čísla ciferníku by mali byť vo väčšine prípadov najväčším a teda najpresvedčivejším textovým poľom na obrázku. Výsledky týchto experimentov si uvedieme v kapitole 5.

4.6.1 EAST model

V článku [57] autori prezentujú EAST ako rýchly a presný model, určený na detekciu textu na obrázkoch, ktorý pozostáva z dvoch hlavných krokov.

V prvom kroku pomocou plne-konvolučnej neurónovej siete, ktorej architektúru môžeme vidieť ilustrovanú na obrázku 4.9, sú na obrázku predikované oblasti s textom, vyznačené obdĺžnikovým rámikom. Na výstupe tejto siete je niekoľko kanálov, pričom jeden predikuje pravdepodobnostnú mapu na úrovni pixelov, pričom každý pixel nadobúda hodnoty z intervalu [0,1]. Zvyšné kanály na výstupe siete predikujú ohraničenia pre jednotlivé textové polia. V druhej fáze EAST algorimu sa na detegovaných orámo-

vaniach aplikuje NMS (Non-Maximum Suppresion). Táto metóda slúži na to, aby jeden objekt bol na obrázku detegovaný iba raz. Najskôr vyberie všetky ohraničenia, ktoré boli predikované s pravdepodobnosťou väčšou ako definovaný prah (metóda prahovania). Následne vyberie ohraničenie s najväčšou pravdepodobnosťou pre dané textové pole a v poslednom kroku všetky ostatné ohraničenia predikované pre daný objekt odstráni.



Obr. 4.9: Architektúra plne konvolučnej neurónovej siete na predikciu textu. [57]

Kapitola 5

Vyhodnotenie

V tejto kapitole si porovnáme výsledky jednotivých experimentov, ktoré sme popísali v kapitole 4 a vyhodnotíme ich celkovú efektivitu. Všetky prístupy, ktoré sme vyskúšali, sa pomocou rôznych metód snažili detegovať alebo segmentovať ciferník vodomeru, ktorý bol zachytený na prirodzenom obrázku. V tabuľke 4 môžeme vidieť zoznam jednotlivých algoritmov s popisom ích prístupu k problému, ktorý sme skúmali. V tabuľke je uvedená aj kategória, do ktorej daný algoritmus patrí. Testované algoritmy sme si rozdelili do dvoch kategórii, podľa toho, či daný prístup na segmentáciu využíval hlboké neurónové siete alebo nevyužíval.

Názov algoritmu	Prístup k problému	Kategória
Morfologická segmentácia	detekcia tvarov v obraze	bez HU
Naivný template matching	hľadanie zhody v obraze	bez HU
S-invariantný template matching	hľadanie zhody v obraze	bez HU
RS-invariantný template matching	hľadanie zhody v obraze	bez HU
Feature matching	hľadanie kľučových bodov v obraze	bez HU
MLP prediktor súradníc	predikovanie súradníc rohov	s HU
Konvolučný prediktor súradníc	predikovanie súradníc rohov	s HU
Konvolučný autoenkóder	predikovanie binárnej masky	s HU
EAST text detektor	detekcia textu v obraze	s HU

Tabuľka 5.1: Zoznam testovaných algoritmov riešiacich segmentáciu ciferníka vodomeru. Označenie **HU** značí hlboké učenie

5.1 Metrika

Úspešnosť jednotlivých metód budeme vyhodnocovať na základe podobnosti binárnych masiek, pričom pri každom testovanom obrázku porovnáme reálnu (anotovanú) masku

s predikovanou maskou, na úrovni pixelov pre daný obrázok. Pre tieto účely použijeme metriku **F1 skóre**, ktorú formálne definujeme nasledovne:

$$F1 = 2 \star \frac{Precision \star Recall}{Precision + Recall} \tag{5.1}$$

kde:

$$Precision = \frac{TP}{TP+FP}$$

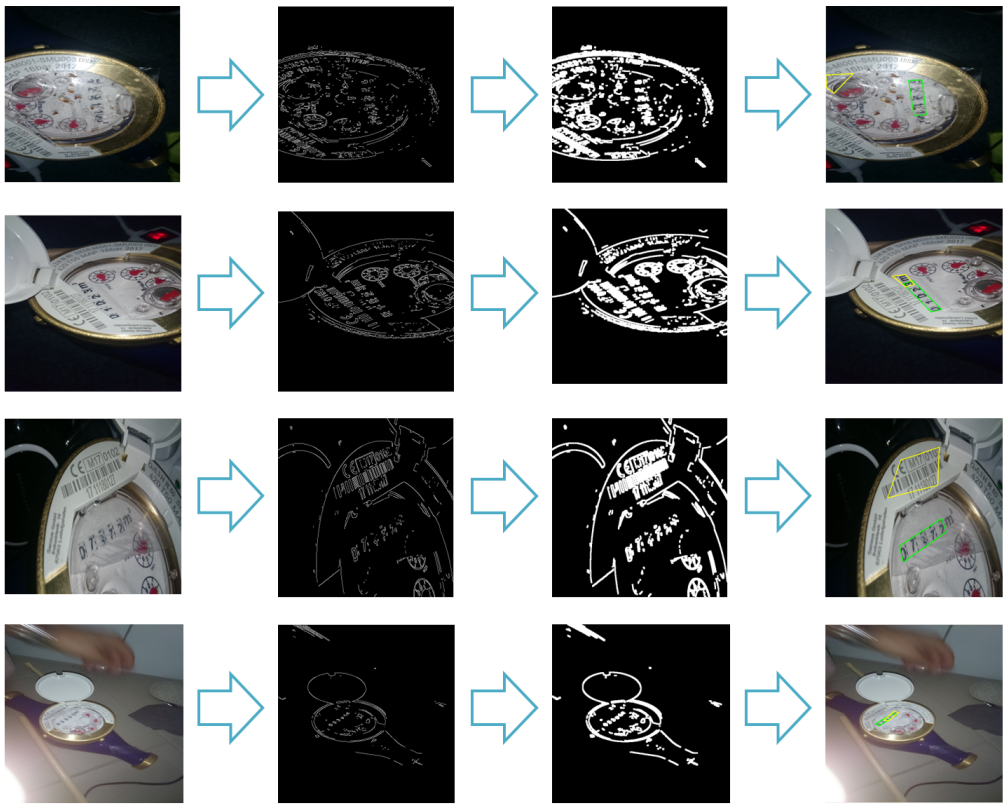
$$Recall = \frac{TP}{TP+FN}$$

Počet True positive pixelov (pixely, ktoré mali byť predikované ako ciferník a boli tak predikované), je označený **tp**. Počet False positive pixelov (pixely, ktoré nemali byť predikované ako ciferník, ale boli tak predikované), je označený **fp** a počet False negative pixelov (pixely, ktoré mali byť predikované ako ciferník, ale neboli tak predikované) predikovanej masky je označený ako **fn**. Pomocné metriky *precision* a *recall* nám zodpovedajú jednoduché otázky, pomocou ktorých sme schopní kvantifikovať úspešnosť jednotlivých predikcií. Kým jedna metrika sa pýta koľko pixelov z tých, ktoré boli predikované sú skutočne relevantné (precision). Druhá metrika určuje, koľko relevantných pixelov bolo predikovaných správne (recall) [58]. Táto metrika je užitočná v tom, že okrem toho, že nás informuje o celkovej veľkosti chyby, ktorú daný model pri predikcii urobil a to prostredníctvom metriky F1-skóre (vzorec 5.1), tak vďaka hodnotám precision a recall vieme aj určiť, či model predikuje ako ciferník priveľkú alebo primalú oblasť z obrázka.

5.2 Úspešnosť metód bez využitia hlbokého učenia

Vo všeobecnosti sa testované metódy, nevyužívajúce hlboké učenie, ukázali byť pomerne slabo adaptovatelné na spracovanie prirodzených obrázkov, ktoré sú veľmi variabilné, čo sa týka rotácie, priblíženia, skosenia, atď. Algoritmus morfologickej segmentácie, ktorý sa zameriava na tvary v obraze a objekty, z obrázka segmentuje na základe ich obrysov a tvarov, je obzvlášť senzitívny na takéto zmeny a jeho výsledky sú silno závislé od toho, aký tvar nadobúdali objekty zobrazené na obrázku. Na obrázku 5.1 vidíme výsledky segmentácie štyroch obrázkov. Z tohto testu vidieť, že pri rôznych pozíciach a polohách vodomeru v rámci obrázka je definovanie ciferníka ako kontúry so štyrmi vrcholmi príliš nejednoznačné.

Na náročnom datasete akým je dataset vodomerov, sú výsledky (pozri tabuľku 5.2) úspešnosti morfologického prístupu pri detekcii ciferníka pochopiteľne neuspokojivé. Tieto výsledky nám však poslúžili ako základný výsledok pri hodnotení ďalších prístupov.



Obr. 5.1: Ukážka výsledkov segmentácie s využitím morfologických operácii. Zelené orámovanie na obrázkoch predstavuje skutočnú anotáciu obrázkov a žlté orámovanie je výsledok morfologickej segmentácie.

Názov algoritmu	Precision	Recall	F1-skóre
Morfologická segmentácia	0.0403	0.0938	0.0257

Tabuľka 5.2: Úspešnosť morfologickej segmentácie

Ďalším prístupom, ktorý sme testovali, boli algoritmy detegujúce zhodu medzi vzorovým a referenčným obrázkom. Prvý z testovaných algoritmov tohto typu bol Naivný Template matching a jeho rôzne obmeny. Naivný template matching je typ algoritmu, ktorý tiež môžeme označiť ako korelačné algoritmy. Jeho nedostatkom však je neinvariantnosť voči rotáciam, škálovaniu a perspektíve. Tento problém sme sa pokúsili eliminovať implementovaním dvoch brute force algoritmov (S-Invariantnt Template matching, RS-Invariant Template matching), ktoré skúšajú aplikovať jednoduchú metódu Template matchingu (ako sme ho popísali v 3.3) pre všetky možné kombinácie preddefinovaných rotácií a škálovaní (v prípade RS-Invariant Template matching algoritmu), alebo len pre všetky preddefinované škálovania (v prípade S-Invariantnt Template matching). Algoritmy by tak mali detegovať na obrázku vzor aj napriek tomu,

že je na referenčnom obrázku iných rozmerov prípadne inak natočený. V tabuľke 5.3 vidíme výslednú úspešnosť na 510 testovacích obrázkoch z datasetu vodomerov pri použití rôznych korelačných funkcií. Ako vzorový obrázok bol použitý „plný“ ciferník. V rámci testov sa neukázali výraznejšie rozdiely vo výsledkoch medzi použitím plného a deravého vzoru ciferníka.

Názov algoritmu	Korelačná funkcia	Precision	Recall	F1-skóre
Naivný TM	CCOEFF_NORMED	0,0283	0,0529	0,0325
Naivný TM	CCOEFF	0,0139	0,0251	0,0153
Naivný TM	CCORR_NORMED	0,0311	0,0494	0,0347
Naivný TM	CCORR	0,0433	0,1014	**0,0531**
S-invariantný TM	CCOEFF_NORMED	0,0226	0,0470	0,0269
S-invariantný TM	CCOEFF	0,0159	0,0267	0,0161
S-invariantný TM	CCORR_NORMED	0,0322	0,0491	0,0350
S-invariantný TM	CCORR	0,0433	0,1014	**0,0531**
RS-invariantný TM	CCOEFF_NORMED	0,0161	0,0253	0,0171
RS-invariantný TM	CCOEFF	$< 0,01$	0,0105	$< 0,01$
RS-invariantný TM	CCORR_NORMED	0,0246	0,0322	0,0253
RS-invariantný TM	CCORR	0,0459	0,1244	**0,0594**

Tabuľka 5.3: Tabuľka výsledkov jednotlivých variánt template matchingových (TM) algoritmov s použitím rozdieľných korelačných funkcií.

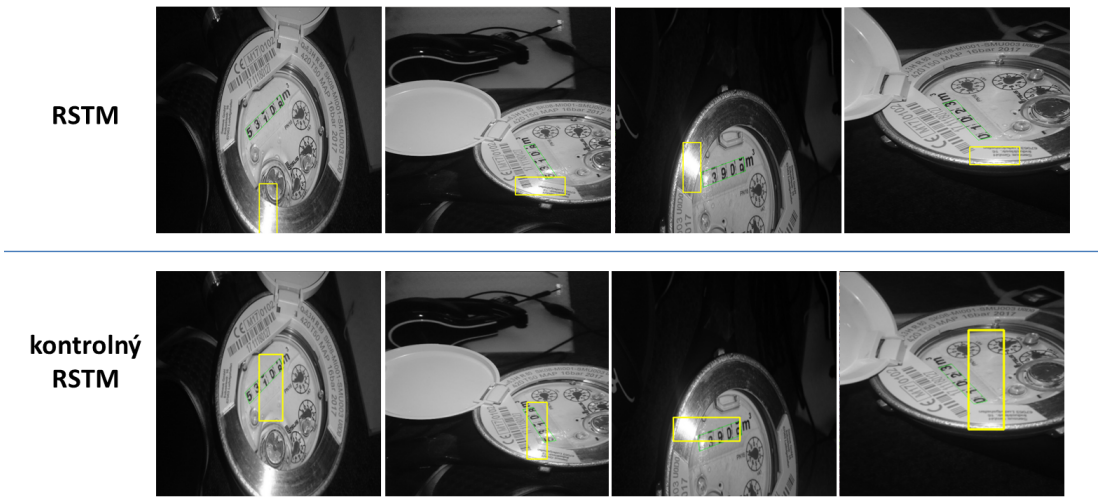
Z výsledkov vidno, že skúšaním väčšieho počtu kombinácií natočenia a priblíženia sa algoritmus nie vždy zlepšoval v presnosti, pričom by sme to intuitívne očakávali. Najlepšie výsledky v prípade každého z algoritmov boli dosiahnuté vyhodnocovaním pomocou obyčajnej krížovej korelácie, pričom normalizácia v tomto prípade výsledky nezlepšovala. Urobili sme preto ešte jeden dodatočný, nie však úplne regulérny test, aby sme si overili správanie algoritmov. Nazveme ho **kontrolný test**.

Názov algoritmu	Korelačná funkcia	Precision	Recall	F1-skóre
Naivný TM	CCORR	0.0283	0.0529	0.0325
S-invariantný TM	CCORR	0,1344	0,2546	0,1515
RS-invariantný TM	CCORR	0,2479	0,4668	0,2786

Tabuľka 5.4: Tabuľka výsledkov jednotlivých variant template matchingových (TM) algoritmov s priebežným vyhodnocovaním výsledkov.

V kontrolnom teste sme výsledky každej kombinácie natočenia a priblíženia referenčného obrázka vyhodnocovali porovnaním s maskou reálneho ohraničenia. Dostali

sme tak pre každý obrázok tú správnu voľbu, ktorú by algoritmus označil za najdôverihodnejší výsledok, ak by jeho metrika fungovala podľa očakávaní tejto úlohy. Výsledky tohto testu vidno v tabuľke 5.4. Z nich vyplýva, že algoritmus v jednotlivých variáciach lokalizoval ciferník zakaždým lepšie, tak ako sme to očakávali. Ukazuje to, že jeho korelačná funkcia tieto zhody vyhodnotila nesprávne. Na obrázku 5.2 vidíme porovnanie niekoľkých výsledkov reálneho a kontrolného testu, pričom v oboch prípadoch sme použili krížovú koreláciu, ktorá dosiahla v testoch najlepšie výsledky.



Obr. 5.2: Porovnanie niekoľkých výsledkov testu RSTM(Rotate-Scale Template Matching) algoritmu a jeho kontrolného ekvivalentu. Zelené orámovanie na obrázkoch predstavuje skutočnú anotáciu obrázkov a žlté orámovanie je výsledok daných algoritmov.

Posledná z testovaných metód, ktorá nevyužíva pre segmentáciu hlboké učenie, bol algoritmus Feature matchingu využívajúci SIFT metódu. Tento algoritmus hľadá kľúčové body v referenčnom aj vzorovom obrázku a jednotlivé body v oboch obrázkoch sa snaží spárovať. Našim zámerom bolo detegovať minimálne 4 kľučové body na referenčnom obrázku a na základe nich vytvoriť orámovanie oblasti, ktoré by sme potom mohli transformovať na binárnu masku. Binárnu masku vyhodnocujeme pomocou metriky popísanej v kapitole 5.1. Neočakávaným problémom tohto prístupu bola jeho neschopnosť detegovať dostatočný počet kľúčových bodov pre vytvorenie orámovania. Algoritmus sme testovali s rôznymi vzorovými obrázkami (celý ciferník, deravý ciferník, znak m^3) v šedotónovej aj farebnej verzii. Na testovacích obrázkoch sa nám však nepodarilo vo väčšine prípadov detegovať dostatočný počet bodov. Výsledky tohto algoritmu boli preto nerelevantné.

5.3 Úspešnosť metód s využitím hlbokého učenia

Testované algoritmy, využívajúce hlboké neurónové siete, prejavili väčšiu schopnosť pochopiť a zvládnuť úlohu segmentácie ciferníka vodomeru aj napriek náročnosti použitého datasetu. Aj v ich prípade však platilo, že vhodné zadefinovanie problému zohráva podstatnú úlohu. Podstatnejším problémom v niektorých testovaných prístupoch bola neschopnosť natrénovať daný model.

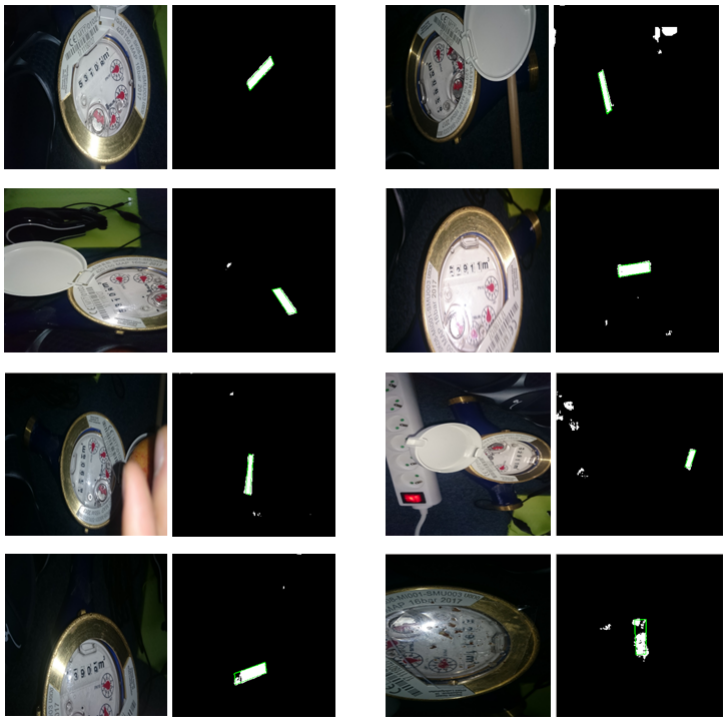
Prvý testovaný prístup s využitím neurónových sietí bol postavený na predikovaní súradníc rohových bodov orámovania ciferníka. Pokúsili sme sa natrénovať dva typy neurónovej siete na 981 trénovacích obrázkoch. V týchto prípadoch sme zvolili menšie architektúry siete, ktoré na vstupe dostali obrázky s rozmermi 100×100. Architektúru oboch sietí sme popísali v sekcii 4.4. Klasická dopredná neurónová sieť mala počas trénovania problém skonvergovať a keď sme trénovanie ukončili po 30 epochách, výsledky jej predikcie sa hýbali na úrovni algoritmu morfologickej segmentácie. Na obrázku 5.3 môžeme vidieť, že sieť sa naučila predikovať nejaký tvar(vždy rovnaký) v blízkosti ciferníka vodomeru.

Obr. 5.3: Ukážka výsledkov predikcie súradníc rohov ciferníka pomocou viacvrstvovej doprednej neurónovej siete. Zelené orámovanie na obrázkoch predstavuje skutočnú anotáciu obrázkov a žlté orámovanie je výsledok predikcie siete.

Verzia s konvolučnou sieťou mala pri trénovaní taktiež problémy. Pri väčšom počte epoch sa naučila predikovať identitu a pri menšom počte bola predikcia tejto siete nepoužiteľná. Tento test ukazuje, že natrénovanie neurónovej siete pre úlohu tohto typu si vyžaduje buď väčšie množstvo dát alebo komplexnejší model.

Druhým testovaným prístupom bolo použitie konvolučného autoenkódera za účelom predikcie binárnej masky. Ideu tohto experimentu sme popísali v sekcii 4.5. V tomto prípade sme sa poučili z neúspešnej implementácie jednoduchších modelov predikujúcich súradnice rohov a model použitého autoenkóderu, ktorý sme popísali v predošlej kapitole s experimentami. Model autoenkóderu sme trénovali na vstupných obrázkoch

s rozmermi 600×600 a po viac ako 1000 epochách nam chyba klesla pod hodnotu 0,001. Model sme otestovali rovnako, ako pri predošlých metódach na 510 testovacích obrázkoch datasetu. Ukážku segmentovaných ciferníkov môžeme vidieť na obrázku 5.4



Obr. 5.4: Ukážka výsledkov segmentácie pomocou konvolučného autoenkódera.

Na testovacich dátach (510 obrázkov) sme touto metódou dosiahli najlepšie výsledky z pomedzi vyskúšaných algoritmov (pozri tabuľku 5.5).

Názov algoritmu	Precision	Recall	F1-skóre
Konvolučný autoenkóder	0.7844	0.9704	0.8478

Tabuľka 5.5: Výsledky testovania konvolučného autoenkódera

Ďalšia testovaná metóda využívala predtrénovaný model plne konvolučnej neurónovej siete EAST, určený na detekciu textu na obraze. V kapitole 4.6 sme si popísali dve heuristiky výberu orámovania, ktoré sú založené na medzivýsledku, ktorý vracia EASt text detektor. Ten definuje na obrázku orámovania na miestach, kde s určitou pravdepodobnosťou predpokladá výskyt textu na obrázku. Porovnanie jednotlivých prístupov je vidno na obrázku

Prvá heuristika vybrala orámovanie ako zjednotenie všetkých nájdených textov na obrázku. Ako je vidieť v tabuľke 5.6, tento prístup dosiahol veľmi vysokú hodnotu recall, ale príliš malú hodnotu preccision. To znamená, že väčšinou musel tento algoritmus vyberať príliš veľké orámovanie v porovnaní s realitou. Druhá heuristika, ktorá

Obr. 5.5: Ukážka výsledkov metódy detekcie textu na obraze. Zelené orámovania predstavujú detegovaný text pomocou modelu EAST. Červené orámovanie je najväčšie detegované textové pole (druhá heuristika) a žlté orámovanie je najmenšie pokrytie detegovaných prámovaní (prvá heuristika).

z pomedzi textových polí, ktoré boli detegované v prvom kroku tohto algoritmu, vyberala iba jedno textové pole s najväčšími rozmermi. Tento postup nedosiahol žiadne spoľahlivé výsledky, pretože orámovania textu s najväčším rozmerom sa na väčšine obrázkov nachádzali mimo ciferníka. Ukážku výsledkov jednotlivých heuristík je vidieť na obrázku 5.5. Z nich môžeme vidieť, že model EAST na obrázkoch datasetu predikoval množstvo orámovaní, ktoré sa žiadneho textu na obrázku netýkali. Rovnako môžeme vidieť že model nepredikoval vo väčšine prípadov číslice ciferníka ako text na obrázku.

Názov algoritmu	Precision	Recall	F1-skóre
EAST textový detektor(zjednotenie orámovaní)	0.0285	0.7390	0.0525

Tabuľka 5.6: Výsledky testovania textovej detekcie na obrázku. Výsledky heuristiky zjednotenia detegovaných orámovaní.

Detekcia textu na obrázku sa pre potreby segmentácie ciferníka ukázala byť v tejto podobe nevyužiteľná. Orámovania textu, ktoré EAST model vygeneruje síce v mnohých prípadoch vyznačí určitú časť ciferníka vodomeru, ale bez vhodného postprocesingu nám algoritmus neposkytne jednoznačné výsledky.

Na záver sme sa ešte pokúsili urobiť jeden test, v ktorom sme sa nechali inšpirovať yolo algoritmom. Ideu tohto algoritmu sme už popísali v kapitole 2.2.1. Trénovacie

obrázky (600×600 pixelov) sme si rozdelili podľa mriežky veľkosti N×N (kde N je počet riadkov/stĺpcov na ktoré sa rozdelí obrázok) na sadu menších obrázkov s rozmermi (600/N) × (600/N) pixelov. Zámerom bolo pomocou takto rozsekaných vstupov natrénovať model konvolučnej neurónovej siete, ktorá by každý dielik klasifikovala, či sa na ňom ciferník vodomeru nachádza alebo nie. Na výstupe siete bola umiestnená plne-prepojená neurónová vrstva s N×N neurónmi. Každý neurón tejto vrstvy slúžil na klasifikáciu jedného dieliku pôvodného obrázka. Po spätnom poskladaní dielikov by sme dostali binárnu masku predikujúcu oblasť výskytu ciferníka na obrázku. Z časového hľadiska sa nám však tento experiment nepodarilo doviesť do prezentovateľnej podoby.

Kapitola 6

Záver

V tejto práci sme sa zaoberali problémom segmentácie ciferníka vodomeru zachytenom na prirodzenom obrázku. V úvode sme sa oboznámili s problematikou a vysvetlili si motiváciu nájdenia optimálnej segmentačnej metódy, ktorá by mohla prispieť k automatizácii procesu odčítania stavu vodomerov. V ďalšej kapitole sme si následne ukázali niekoľko existujúcich riešení, ktoré sa venovali téme segmentácie obrazu. Výskum posledných rokov nazačuje, že trendom v tejto problematike, ale aj v oblasti počítačového videnia celkovo, sa ukazuje byť používanie konvolučných neurónových sietí. V kapitole 3 zameranej na použité technológie, sme si vysvetlili technické detaily štandardných metód, používaných v rámci segmentácie obrazu či detekcie objektov na ňom.

Naším cieľom v tejto práci bolo jednak zanalyzovať populárne segmentačné prístupy a overiť, či ich aplikovaním vieme riešiť segmentáciu ciferníka na vodomere s dostatočnou presnosťou, ale tiež sme chceli vyskúšať alternatívne postupy a vzájomne ich porovnať. Pre potreby našich experimentov sme mali k dispozícii dataset s 1491 obrázkami vodomerov odfotených telefónom, ktoré boli zhotovené s cieľom simulovať náročné podmienky prostredia, v ktorom je potrebné vodomer odfotografovať. Dataset sa skladal zo snímkov rôznej kvality. Obsahoval obrázky so zreteľne čitatelným vodomerom, ale aj obrázky, ktoré zobrazovali vodomer rozostrene, presvetlene, tmavo, atď. Všetky nami testované metódy sme v práci rozdelili do dvoch kategórií. Jedna skupina boli algoritmy, ktoré za účelom segmentácie obrazu využívajú hlboké učenie. Druhú skupinu tvorili algoritmy, ktoré segmentujú obraz bez využitia hlbokého učenia. Podľa očakávaní sa použité algoritmy nevyužívajúce hlboké učenie ukázali byť menej schopné vysporiadať sa so spracovaním informácie z prirodzených obrázkov. Naopak, neurónové siete preukázali schopnosť extrahovať z obrázku potrebnú informáciu aj v náročnejších podmienkach. Ich limitáciou sa však zatiaľ ukazuje byť potreba dostatočného množstva kvalitne anotovaných dát a výpočtová sila potrebná pre trénovanie modelu. Z pomedzi testovaných prístupov sme najlepšie výsledky dosiahli pomocou konvolučného autoenkódera, ktorý na výstupe vrátil binárnu masku predikujúcu po-

lohu ciferníka vodomeru v rámci obrázka. Táto metóda na 510 testovacích obrázkoch dosiahla chybovosť $< 16\%$. Viaceré z testovaných prístupov nedosahovali úroveň použiteľných výsledkov. Pri niektorých prístupoch sa tak stalo z dôvodu, že dosiahli svoje limity v rámci použitého datasetu, ale pre viaceré metódy ostal do budúcnosti priestor na zlepšenie výsledkov. Vhodnejšie predspracovanie, ktoré by určilo správnu orientáciu vodomeru na obrázku, by mohlo výrazne pomôcť pri následnom aplikovaní základného Template matching algoritmu. Algoritmy predikujúce súradnice rohových bodov ciferníka zlyhali pri trénovaní modelov, čo ukázalo potrebu využitia väčšej architektúry siete. Otvorenou otázkou ostáva, aké výsledky by bol tento prístup schopný dosiahnúť pri použití väčšieho modelu a väčšej výpočtovej sily. Ďalším predmetom práce do budúcna ostáva aj implementácia experimentu s klasifikáciou segmentov obrázka, ktorý sme popísali v závere kapitoly 5.

Literatúra

[1] LeCun, Y., et al. (1990). Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems, (pp. 396-404).

[2] Kowsari, K., et al. (2018). Rmdl: Random multimodel deep learning for classification. In Proceedings of the 2nd International Conference on Information System and Data Mining (pp. 19-28). ACM.

[3] Shapiro, L. G., Stockman, G. C. (2001): "Computer Vision", pp 279-325, New Jersey, Prentice-Hall, ISBN 0-13-030796-3

[4] History of image segmentation. URL: *https://medium.com/@eaifundoffical/history-of-image-segmentation-655eb793559a*, (1.5.2019).

[5] Roberts, L. (1965): "Machine Perception of Three-Dimensional SolidsÏn Optical and Electro-Optical Information Processing, pp. 159-197, J. T. Tippett, et al., (Ed.'s), MIT Press, Cambridge, Mass.

[6] Prewitt, J. M. (1970). Object enhancement and extraction. Picture processing and Psychopictorics, 10(1), (pp. 15-19).

[7] Canny, J. (1987). A computational approach to edge detection. In Readings in computer vision (pp. 184-203). Morgan Kaufmann.

[8] Chudasama, D., Patel, T., Joshi, S., Prajapati, G. I.. Ïmage segmentation using morphological operations.Ïnternational Journal of Computer Applications 117, no. 18 (2015).

[9] Martinsky, O. (2007). Algorithmic and mathematical principles of automatic number plate recognition systems. In Brno University of technology.

[10] Sarvaiya, J. N., Patnaik, S., Bombaywala, S. (2009). Image registration by template matching using normalized cross-correlation. In 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies (pp. 819-822). IEEE.

[11] Kim H.Y., de Araújo S.A. (2007) Grayscale Template-Matching Invariant to Rotation, Scale, Translation, Brightness and Contrast. In: Mery D., Rueda L. (eds) Advances in Image and Video Technology. PSIVT 2007. Lecture Notes in Computer Science, vol 4872. Springer.

[12] Parthasarathy, D. (2017). A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN. URL: *https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4*, (1.5.2019)

[13] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[14] Prakash, J. (2018) Image Classification Architectures review. URL: *https://medium.com/@14prakash/image-classification-architectures-review-d8b95075998f*, (30.4.2019)

[15] Hu, J., Shen, L., Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

[16] Pröve, P. L. (2017). Squeeze-and-Excitation Networks: Setting a new state of the art on ImageNet. URL: *https://towardsdatascience.com/squeeze-and-excitation-networks-9ef5e71eacd7*, (29.4.2019).

[17] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[18] Ren, S., He, K., Girshick, R., Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[19] He, K., Gkioxari, G., Dollár, P., Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[20] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[21] Badrinarayanan, V., Handa, A., Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1505.07293.

[22] Badrinarayanan, V., Kendall, A., Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), 2481-2495.

[23] Noh, H., Hong, S., Han, B. (2015). Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE international conference on computer vision (pp. 1520-1528).

[24] Kumar, M., Saxena, R. (2013). Algorithm and technique on various edge detection: A survey. Signal Image Processing, 4(3), 65.

[25] Jain, R., Kasturi, R. Schunck, B.G. (1995). Machine vision (Vol. 5, pp. 385-395). New York: McGraw-Hill.

[26] Blázsovits, G. Digital Image Processing, Interaktívna učebnica spracovania obrazu. In Katedra aplikovanej informatiky FMFI UK Bratislava. URL: *http://dip.sccg.sk/predspra/predspra.htm*, (25.4.2019).

[27] Woods, R.E., Gonzalez, R.C. (1992). Digital Image Processing. Publisher: Prentice Hall, Upper Saddle River, New Jersey.

[28] Convolutional operation. URL: *http://datahacker.rs/002-cnn-edge-detection/*, (30.4.2019).

[29] Sahir, S. (2019). Canny Edge Detection Step by Step in Python Computer Vision. URL: *https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123*, (15.4.2019).

[30] OpenCV. Canny Edge Detection. URL: *https://docs.opencv.org/3.4.0/d7/de1/tutorial_js_ca*
(1.5.2019)

[31] Kečkéš, M. (2010). Äplikace pro implementaci morfologických operací v digitálních obrazech", Bakalárska práca, Vysoké učení techniké v Brně.

[32] Ruman, J. (2010) "Binárne morfologické operácie v spracovaní obrazov", Diplomová práca, SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE.

[33] Sternberg, S.R. (1986). Grayscale Morphology. Computer Vision, Graphics and Image Processing, 35, (pp. 333-355).

[34] Sonka, M., Hlavac, V., Boyle, R. (2014). Ïmage Processing, Analysis, and Machine Vision", Cengage Learning.

[35] Swaroop, P., Sharma, N. (2016). An overview of various template matching methodologies in image processing. International Journal of Computer Applications, 153(10), 8-14.

[36] Karami, E., Prasad, S., Shehata, M. (2017). Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. arXiv preprint arXiv:1710.02726.

[37] David G. Lowe (2004): "Distinctive Image Features from Scale-Invariant Keypoints",International Journal of Computer Vision

[38] OpenCV. Introduction to SIFT (Scale-Invariant Feature Transform). URL: *https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.htm* (1.5.2019).

[39] Koenderink, J.J. (1984|. The structure of images. Biological Cybernetics, 50(5), (pp. 363-396).

[40] Lindeberg, T. 1994. Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics, 21(2):224-270.

[41] Michael, A. N., (2015). "Neural Networks and Deep Learning", Determination Press, 2015

[42] Hladík, J, (2016). Nástroj pro návrh hluboké neuronové sítě, Bakalárska práca, Vysoké učení techniké v Brně.

[43] Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Cornell Aeronautical Laboratory, Psychological Review, 65, (pp. 386-408).

[44] Arthur, A. First neural network for beginners explained (with code). URL: *https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf*, (1.5.2019).

[45] Berger, Ch. (2016). Perceptrons - the most basic form of a neural network. URL: *https://appliedgo.net/perceptron/*, (1.5.2019).

[46] Torres, J. (2018). Learning process of a neural network - How Do Artificial Neural Networks Learn?. URL: *https://towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7*, (1.5.2019).

[47] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[48] Krajčovičová, M. (2015). Konvoluční neuronová síť pro zpracování obrazu, Diplomová práca, Vysoké učení techniké v Brně.

[49] Nair, V., Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

[50] TinyMind. How does ReLU compare. URL: *https://www.tinymind.com/learn/terms/relu*, (1.5.2019).

[51] Bhandare, A., Bhide, M., Gokhale, P., Chandavarkar, R. (2016). Applications of convolutional neural networks. International Journal of Computer Science and Information Technologies, 7(5), (pp. 2206-2215).

[52] Oppermann, A. (2018). Deep Autoencoders For Collaborative Filtering: Predicting the Rating a User would give a Movie - A practical Tutorial. URL: *https://towardsdatascience.com/deep-autoencoders-for-collaborative-filtering-6cf8d25bbf1d*, (1.5.2019).

[53] I. Goodfellow, Y. Bengio, and A. Courville, (2016). Deep Learning. MIT Press, 2016. URL: *http://www.deeplearningbook.org*, (1.5.2019).

[54] Abadi, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).

[55] Cognexa cxflof Tenserflow. URL: *https://github.com/Cognexa/cxflow-tensorflow*, (1.5.2019)

[56] Bradski, G., Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc.

[57] X. Zhou, C. Yao, H. Wen, Y. Wang,S. Zhou, W. He, J. Liang (2017). EAST: An Efficient and Accurate Scene Text Detector. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[58] Precision and recall. URL: *https://en.wikipedia.org/wiki/Precision_and_recall*, (1.5.2019)

[59] Le, J. (2018). How to do Semantic Segmentation using Deep learning, URL: *https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef*, (1.52019)

[60] Thalles, S. (2018) Deeplab Image Semantic Segmentation Network URL: *https://sthalles.github.io/deep_segmentation_network/* (1.5.2019)

[61] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[62] Navladni, A. (2019). Neural Network Models in R. URL: *https://www.datacamp.com/community/tutorials/neural-network-models-r*, (1.5.2019)

[63] Budhiraja, A. (2016). Dropout in (Deep) Machine learning, URL: *https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5*, (1.5.2019)

[64] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

V prílohe k tejto práci sú na CD, ktoré je súčasťou práce, nahraté súbory s použitými testovacími algoritmami.

Zoznam súborov:

- *corner_predictor_convnet.py* - testovací skript využívajúci konvolučnú neurónovú sieť na predikciu súradníc rohov ciferníka (Python 3.6.3).
- *corner_predictor_mlp.py* - testovací skript využívajúci doprednú neurónovú sieť na predikciu súradníc rohov ciferníka (Python 3.6.3).
- *feature_matching.py* - testovací skript párujúci príznaky porovnávaných obrázkov (Python 2.7.14).
- *mask_predictor_autoencoder.py* - testovací skript využívajúci konvolučný autoenkóder za účelom predikcie binárnej masky (Python 3.6.3).
- *morphological_segmentation.py* - testovací skript s algoritmom morfologickej segmentácie (Python 2.7.14).
- *naive_template_matching.py* - testovací skript porovnávajúci obrázky pomocou jednoduchého Template matching algoritmu (Python 2.7.14).
- *s_template_matching.py* - testovací skript so škálovateľným Template matching algoritmom (Python 2.7.14).
- *rs_template_matching.py* - testovací skript so škálovateľno-rotačným Template matching algoritmom (Python 2.7.14).
- *s_comparing_template_matching.py* - testovací (kontrolný) skript so škálovateľným Template matching algoritmom (Python 2.7.14).
- *rs_comparing_template_matching.py* - testovací (kontrolný) skript so škálovateľno-rotačným Template matching algoritmom (Python 2.7.14).
- *text_detector.py* - testovací skript detegujúci text na obrázku pomocou predtrénovaného EAST modelu (Python 3.6.3).