TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Anni Uutma

Ülevaade optilise märgituvastuse meetoditest

Bakalaureusetöö (9 EAP)

Juhendaja: Artjom Lind, MSc

Tartu 2016

Ülevaade optilise märgituvastuse meetoditest

Töös kirjeldatakse kahte meetodit, mille abil on võimalik optilise märgituvastuse abil autode numbrimärke tuvastada. Samuti selgitatakse funktsioonide tööd, mille abil pilte töödeldakse ja näidatakse, kuidas kasutada neid nii, et neist võimalikult palju kasu oleks.
See töö annab ülevaate lihtsamatest meetoditest, mida optilises märgituvastuses kasutatakse ja annab huvitatud inimesele hea stardipunkti, kust oma edasist uurimist jätkata.
Võtmesõnad: Raalnägemine, optiline märgituvastus, mustrisüsteem, numbrimärgituvastus, pilditöötlus.
CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Overview of optical character recognition methods

In this thesis two methods for optical licence plate recognition are shown. In addition the functions for image processing are explained and the use cases are demonstrated.
This work is a complete overview of basic methods of optical character recognition and is a good starting point for the newcomers in this topic.
Keywords: Computer vision, optical character recognition, pattern matching, licence plate recognition, image processing.
CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

1 Sissejuhatus

Masinate abil märkide ja teksti tuvastamine on teinud võimalikuks rakendused, mis muudavad elu tunduvalt lihtsamaks: QR-koodi skanneerimine saadab inimese sekunditega veebilehele, kus ta saab vajalikku informatsiooni; paberkujul materjalide masintekstiks muutmine ja nende digitaalsel kujul säilitamine ja levitamine on tänu sellele lihtne ja soodustab infosüsteemidele üleminekut, kust kõigil on palju lihtsam vajalikke andmeid kätte saada.
Huvipakkuvast objektist või tekstist pildi tegemine ja selle abil info otsimine on palju mugavam kui otsingusõnade sissetrükkimine, kuid selle meetodi töötamise eelduseks on see, et seade on võimeline aru saama, mis pildil on.
Peamiseks probleemiks on märgituvastuse algoritmidel olnud haruldastes fontides või käsikirjas kirjutatud tekstidest märkide äratundmine. Samuti ei saa alati olla kindel, et pilt on tehtud hea kvaliteediga ja ühtlases valguses, mis raskendab märkide äratundmist veelgi. Aja jooksul on aga algoritmides kasutatavad meetodid muutunud keerukamaks ning tänu sellele on nende universaalsus ja täpsus kasvanud. Praeguseks on olemas mitmeid rakendusi, mis võimaldavad piltidelt teksti eraldamist ja masintekstiks muutmist, paljud neist säilitavad selle käigus ka teksti algse formaadi.
Märgituvastuse algoritmi saab jagada neljaks etapiks: pildi saamine, pildi eeltöötlemine, märgi avastamine ja märgi tuvastamine. Esimeses etapis antakse algoritmile ette pilt, millest see vajadusel huvipakkuva osa eraldab. Teine etapp on algoritmi üks olulisemaid – selles töödeldakse pilti nii, et märgid oleks võimalikult hästi eristatavad ja neid oleks lihtsam avastada. Funktsioonid, mida selleks pildile rakendatakse on natuke varieeruvad, kuid tavaliselt on nende hulgas pildi halltoonidesse muundamine, hägustamine, teravustamine, pildi binaarseks muundamine, pildi suuruse muutmine ja vajadusel ka pildi mingil määral keeramine, juhul kui see ei ole otse. Kolmandas etapis üritatakse pildilt märgi asukoht tuvastada, selleks liigutakse pildil pikselhaaval ja otsitakse tumedaid piksleid, mis võiks vastata märgile. Viimases etapis üritatakse tuvastada märki, mis asub eelmises etapis leitud positsioonil, seda on võimalik teha mitmel erineval viisil: mustrisüsteemiga, struktuurisüsteemiga, omadussüsteemiga või masinõppemeetodi abil. Mustrisüsteem võrdleb leitud märki malliga ja kui nende sarnasus on piisavalt suur, siis loetakse märk tuvastatuks. Selle meetodi kasutamiseks on vajalik mall, millel olevad märgid on tehtud võimalikult sarnaseks märkidele, mida võib esineda pildil. Struktuurisüsteem jaotab mallil olevad märgid gruppideks vastavalt sarnastele omadustele, tuvastab leitud märgilt erilised omadused, nt kaared või ringid, ja võrdleb seda seejärelt selle grupi märkidega, mille omadused on samasugused. Omadussüsteem otsib avastatud märkidelt erilisi omadusi ja võrdleb neid mallil olevate märkide omadustega, kui leitakse vastavus, siis loetakse märk tuvastatuks. Masinõppemeetodit kasutades tuleb seda kõigepealt õpetada märke ära tundma

ja seejärel saab teda kasutada märgi ära tundmiseks. Kuidas see protsess täpselt välja näeb, sõltub sellest, kas tegemist on tehisnärvivõrguga, tugivektormasinaga või mõne muu meetodiga.
Bakalaureusetöös antakse ülevaade ja kirjeldus meetoditest, mida olemasolevates märgituvastuse algoritmides eelpool kirjeldatud etappides kasutatakse eesmärgiga luua eestikeelset õppematerjali optilise märgituvastuse kohta ja rakendada üks optilise märgituvastuse meetod.

2 Taust ja varasemad tööd

Peatüki esimeses osas selgitatakse mõningaid termineid, mis on olulised selles bakalaureusetöös kirjeldatavate algoritmide töö mõistmiseks. Teises osas kirjeldatakse täpsemalt optilise märgituvastuse algoritmi töö etappe ja meetodeid, mida varasemates töödes nendes etappides on kasutatud.

2.1 Terminid

Alampeatükis selgitatakse kõigepealt termineid, mida kasutatakse pildi iseloomustamiseks, hiljem selliste funktsioonide tööd, mida kasutatakse pildi töötlemiseks.

2.1.1 Pildi iseloomustamine

Värvi gradient näitab, kui suur on pildi punktis üleminek heledast tumedale toonile või vastupidi. Mida suurem on üleminek, seda suurem on gradient. Seda kasutatakse töötlemisel pildil olevate objektide äärte tuvastamiseks [1].

2.1.2 Pildi töötlemiseks kasutatavad funktsioonid

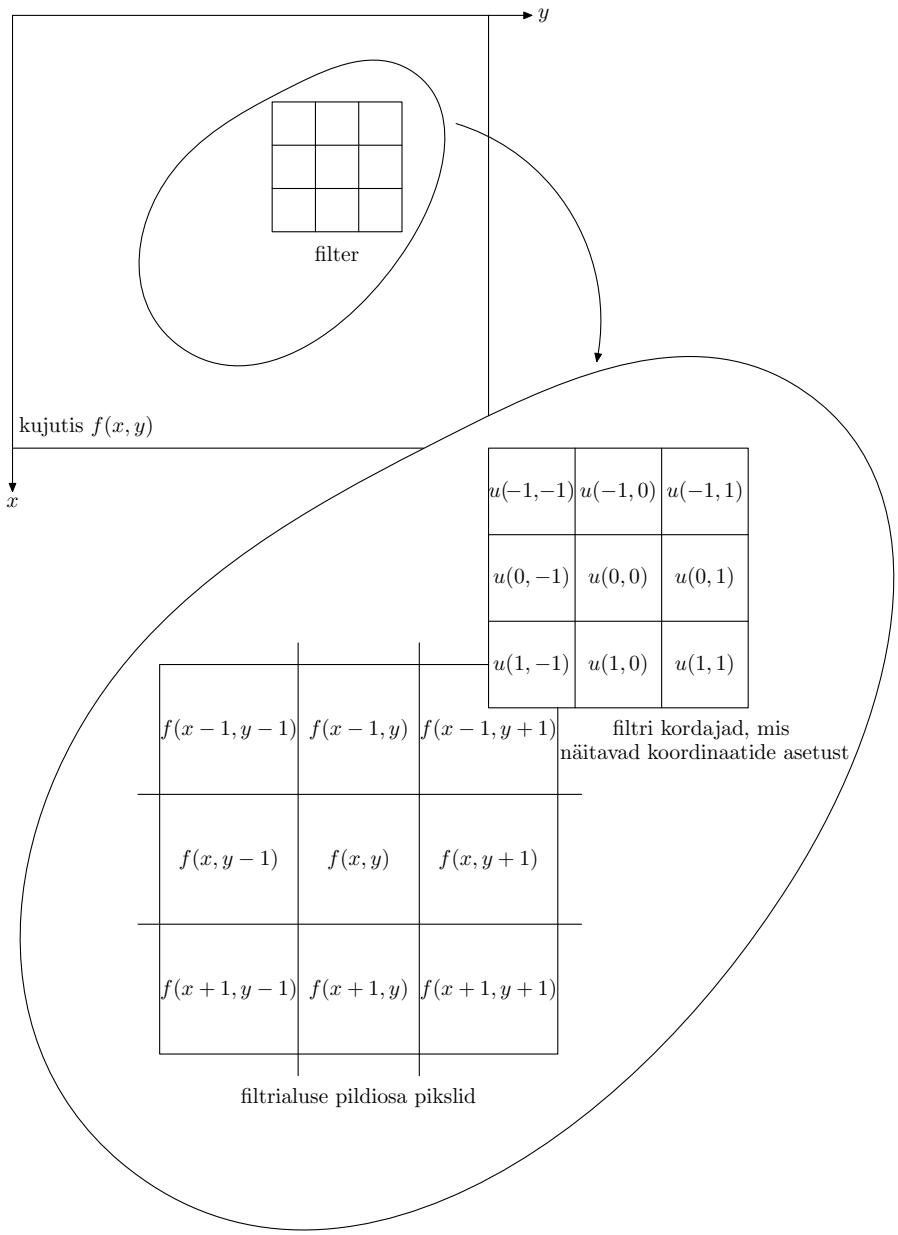
Halltoonidesse muundamine on raamatu *"Digital Signal Processing: Fundamentals and Applications"* [2] järgi funktsioon, mis arvutab piksli rohe-sini-puna-väärtusele (edaspidi RGB) vastava heledus-faas-kvadratuur-väärtusele (edaspidi YIQ) väärtuse ja seab piksli uueks väärtuseks tulemuse heleduse komponendi. Seal on selgitatud ka piksli RGB ja YIQ väärtust, millest kõike siin töös välja ei tooda. Halltoonidesse muundamise funktsiooni töö mõistmiseks on oluline, et raamatu põhjal näitab heleduse komponent piksli valguse intensiivsust. Raamatus on välja toodud järgmine valem, mille abil on võimalik arvutada YIQ väärtust:

$$\begin{pmatrix} Y(m,n) \\ I(m,n) \\ Q(m,n) \end{pmatrix} = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ 0,596 & -0,274 & -0,322 \\ 0,212 & -0,523 & 0,311 \end{pmatrix} \begin{pmatrix} R(m,n) \\ G(m,n) \\ B(m,n) \end{pmatrix},$$

kus konstantmaatriksis olevad väärused vastavad BT.601 standardile ülerealaotusega analoogvideo (*interlaced analog video*) kodeerimiseks digitaalse video vormi [3].

Teravustamist kirjeldatakse raamatus *"Digital image processing"* [4] kui funktsiooni, mille eesmärgiks on tuua pildil välja objektid, mis on algselt hägustatud. Teoses kirjeldatakse, et selle protsessi läbiviimiseks moodustatakse filter suurusega $m \times n$, mida liigutatakse pildil punktist punkti ja igas punktis arvutatakse

vastavate filtri elementide ja pildi pikslite korrutis, mis liidetakse lõpuks kokku ja tulemus saab selle punkti väärtuseks, mis oli filtrialuse pildiosa keskmine piksel. Sellist tegevust nimetatakse raamatus ruumiliseks filtreerimiseks (*spatial filtering*) (joonis 1).



Joonis 1. Ruumiline filtreerimine [4]

Teravustamiseks kasutatavad filtrid põhinevad raamatu järgi esimest ja teist

järku tuletistel. Neil tuletistel ei ole ühest definitsiooni, kuid iga definitsioon, mille järgi selliseid esimest järku tuletisi soovitakse arvutada, peab vastama kolmele tingimusele: see peab olema konstantse väärtusega piirkonnas null; see ei tohi olla null kallaku või astme alguses; see ei tohi olla null kallaku peal. Samuti on raamatus välja toodud sarnased tingimused teist järku tuletise jaoks: see peab olema konstantse väärtusega piirkonnas null; ei tohi olla null kallaku või astme alguses ja lõpus; see peab olema null konstantse kaldega kallaku peal. Raamatus kasutatakse teravustamiseks teist järku tuletist, sest see suudab paremini pisemaid detaile esile tõsta. Täpsemalt kasutatakse Laplace'i operaatorit:

$$\triangle^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Mille diskreetne vorm kahe muutuja jaoks on

$$\triangle^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Gaussi hägustamine on raamatu [4] põhjal pildi diskreetset Fourieri transformatsiooni muutev protsess, kus selleks kasutatakse Gaussi madalpääsufiltrit. Madalpääsufilter tähendab raamatu põhjal seda, et selline filter laseb läbi ainult madalad sagedused ja kõrvaldab kõrged sagedused. Kuna kõrged sagedused väljendavad järsku muutust pildi värvides, siis tähenda selline filtreerimine, et need järsud muutused kaotatakse ja tulemuseks on pilt, mille värvid on ühtlasema üleminekuga. Raamatus kasutatakse selliseks filtreerimiseks valemit:

$$H(u,v) = e^{-\frac{D^2(u,v)}{2\sigma^2}}$$

Kus $D(u,v)$ on kaugus keskpunktist.

Globaalne pildi binaarseks muundamine on raamatu *"Digital Image Processing"* [4] järgi funktsioon, mis suurendab pildi kontrastsust, muutes selle binaarseks. Seal kirjeldatakse funktsiooni tööd järgnevalt: tegemist on funktsiooniga kujul

$$g(x,y) = \begin{cases} 1, & f(x,y) > T \\ 0, & f(x,y) \leq T \end{cases}$$

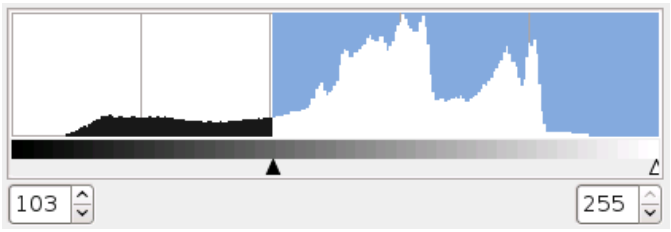
Kus $f(x,y)$ on pildil olev piksel, T on lävend ja $g(x,y)$ on binaarseks muundatud pildil olev piksel. Joonistel 2.1, 2.2 ja 2.3 on näha erinevad versioonid pildist, kus joonis 2.1 on algne pilt, joonis 2.2 on halltoonidesse muundatud pilt ja joonis 2.3 on binaarseks muundatud pilt.

Joonis 2.1

Joonis 2.2

Joonis 2.3

Joonisel 3 on joonise 2.2 histogramm mille x-teljel on halltoonid mustast valgeni ehk vahemik $[0, 255]$ ja y-teljel on pikslid. Mida kõrgem on tulp vastava värvitooni kohal, seda rohkem on seda tooni piksleid pildil. Must nool tähistab lävendit T.



Joonis 3. Histogramm

Lokaalne pildi binaarseks muutmine erineb raamatu *"Digital Signal Processing"* [4] põhjal globaalsest pildi binaarseks muutmisest selle poolest, et kui globaalses variandis oli üks lävend, siis lokaalses variandis arvutatakse iga $m \times n$ suurusega pildiosa jaoks eraldi lävend. Selleks arvutatakse raamatus kõigepealt välja pildiosa pikslite väärtuste standardhälve $\sigma_{x,y}$ ja keskmine $m_{x,y}$ ning seejärel võrreldakse nendega iga piksli väärtust. Kõigepealt koostatakse teoses predikaat:

$$Q(\sigma_{x,y}, m_{x,y}) = \begin{cases} \text{tõene}, & \text{kui } f(x,y) > a\sigma_{x,y} \text{ JA } f(x,y) > bm_{x,y} \\ \text{väär}, & \text{kõigil teistel tingimustel} \end{cases}$$

kus a ja b on mittenegatiivsed konstandid ning seejärel antakse pikslitele väärtused:

$$g(x,y) = \begin{cases} 1, & \text{kui } Q(\sigma_{x,y}, m_{x,y}) \text{ on tõene} \\ 0, & \text{kui } Q(\sigma_{x,y}, m_{x,y}) \text{ on väär} \end{cases}$$

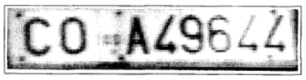
2.2 Varasemad tööd

Alampeatükis selgitatakse optilise märgituvastuse algoritmi tööd. Selleks on peatükk jaotatud neljaks osaks, mis vastavad algoritmi etappidele. Etappides toimuvat selgitatakse läbi näidete, mis on toodud autori poolt loetud töödest, kus kirjeldatakse erinevaid optilise märgituvastuse algoritme.

2.2.1 Pildi saamine

Itaalia tolli jaoks mõeldi välja automaatse numbrimärgituvastuse algoritm [5], mida siin ja edaspidi selles bakalaureusetöös nimetatakse numbrimärgituvastuse algoritmiks. Sellesse algoritmi sisestatakse pilt tolliväravast läbi sõitva auto tagaosast, mis on tehtud tollikaamera poolt (joonis 4.1). Pildist numbrimärgi eraldamiseks otsib algoritm kõigepealt suurima lokaalse kontrasti, mis tõenäoliselt vastab ristkülikule, mis sisaldab numbrimärki. Väidet põhjendatakse sellega, et numbrimärgil on valgel taustal mustad numbrid, mistõttu on kontrast seal kohas suur. Värvi gradiendi analüüsiga tuvastatakse ristküliku hinnanguline asukoht ja see eraldatakse pildist (joonis 4.2).

Joonis 4.1. Pilt auto tagaosast [5]



Joonis 4.2. Pildilt eraldatud numbrimärk [5]

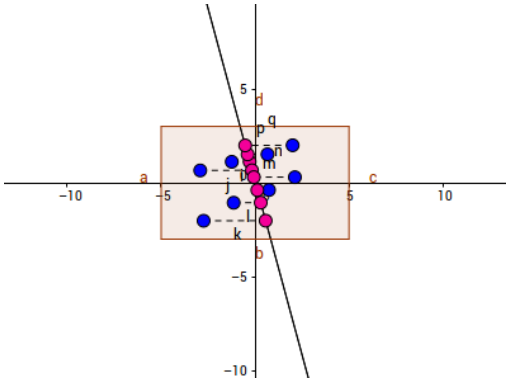
2.2.2 Pildi eeltöötlemine

Numbrimärgituvastuse algoritmis on pildi eeltöötlemine jagatud kolme osasse: 1) Töötlemine ja histogrammi venitamine selleks, et parandada pildi kvaliteeti; 2) märgi suuruse ja numbrimärgi kalde hindamine; 3) pildi standardseks muutmine. Esimeses osas rakendatakse pildile Gaussi hägustamist. Seda eelistatakse pildi binaarseks muundamisele, sest algoritmi autoritel on soov säilitada võimalikult suur hulk pildil olevast infost, lisaks muudab valgustuse ebaühtlus binaarseks muundamise väga keeruliseks.
Tulemusele rakendatakse halli astmete teisendamist, mis suurendab pildi kontrasti ja parandab seeläbi detailide nähtavust. Halli astmete teisendamine seisneb selles,

et olgu pildil olevate halltoonide vahemik [a, b] ja tulemuseks soovitav vahemik [A, B], nii et $[a, b] \subset [A, B]$. Funktsioon t seab igale pildil olevale halltoonile i vastavusse tooni suuremast vahemikust t(i)=i'. Funktsioon t defineeritakse järgnevalt:

$$i' = \begin{cases} \frac{B-A}{b-a}i + \frac{Ab-Ba}{b-a}, & a \leq i \leq b \\ A, & i < a \\ B, & i > b \end{cases}$$

Kus A = 0 ja B = 255.
Teises osas arvutatakse kõigepealt välja lokaalsed maksimum- ja miinimumpunktid, selleks võrreldakse piksli väärtust temast kahel pool horisontaalselt kõrval asuvate kahe piksli väärtusega, kui antud piksel on suurima väärtusega, siis on tegemist lokaalse maksimumiga, vastasel juhul lokaalse miinimumiga. Selliste punktide otsimist põhjendatakse sellega, et need asetsevad tavaliselt märgi sees ja kahe märgi vahel. Seejärel üritatakse hinnata numbrimärgi pildi kallet ja märgi kõrgust. Pilt asetatakse koordinaatteljestikule nii, et pildi keskpunkt on teljestiku keskpunktis ja y-telge hakatakse nurga α võrra keerama. Pärast iga keeramist projitseeritakse varasemalt leitud lokaalsed huvipunktid y-teljele ja pildi kaldenurgaks loetakse selline nurk α, mille korral on punktide hajuvus vähim. Sellise otsuse aluseks on, et kui numbrimärk tõesti on sellise nurga all, siis on punktid vähem hajusalt kui siis, kui numbrimärk sellise nurga all ei ole. Joonisel 5 on näha pildil asuvad punktid (sinised) ja need punktid, kui need on projitseeritud y-telje suhtes nurga α võrra kallutatud joonele (roosad).



Joonis 5. Punktide projitseerimine joonele

Viimases etapis rakendatakse pildile kõigepealt geomeetrilist teisendamist, mille käigus kallutatakse vajadusel pilti vastavalt eelnevas punktis leitud kaldenurgale, lisaks suurendatakse või vähendatakse pildil märke vastavalt vajadusele. Kui suurendamisel või vähendamisel tekib tühje piksleid, siis nende täitmiseks sobiva halli

astmega rakendatakse bilineaarset interpoleerimist (joonisel 6.1 on suurendatud pilt ja joonisel 6.2 on suurendatud pilt pärast bilineaarset interpoleerimist). Tulemuseks saadav pilt on standartselt 20 x 230 pikslit.

Joonis 6.1 [4]

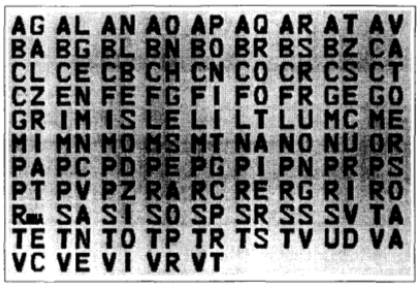
Joonis 6.2 [4]

2.2.3 Märgi avastamine

Numbrimärgituvastuse algoritmis eristatakse Itaalia numbrimärgis kahte osa, esimesed kaks märki on provintsi märgistus ja järgmised kuus on unikaalsed sõiduki identifikaatorid (joonis 7). Provintsi märgistus üritatakse tuvastada korraga, mitte kahe erineva märgina. Selle eraldamiseks lõigatakse pildilt välja osa, kus märgistus peaks loogiliselt asuma ehk võetakse normaliseerimise tulemusena saadud 20 x 230 piksli suurusest pildist 20 x 64 piksli suurune osa.

2.2.4 Märgi tuvastamine

Numbrimärgituvastuse algoritmis kasutatakse märkide tuvastamiseks malle, milles ühel on kõigi provintside märgistused, mida on kokku 95 (joonis 7), ja teisel on võimalikud numbrid ja tähed, mis võivad esineda unikaalses identifikaatoris, neid on kokku 31. Mallidel olevate tähiste ümber on tehtud ka nö varjud, et need oleks halli tasemetelt võimalikult sarnased keskmise numbrimärgi pildil oleva tähise tasemetega.

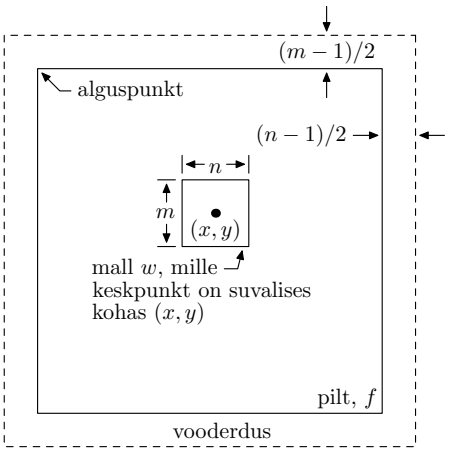


Joonis 7. Mall [5]

Mallil olevat tähist võrreldakse provintsi märgistust sisaldava osaga pildist 1 piksli kaupa ja tulemuseks tekitatakse 95 x 64 maatriks, kus veerud vastavad pildi piksliveergudele ja read sisaldavad malle. Maatriksis sisalduvad numbrid tähistavad vastava pildiosa ja malli osa rist-korrelatsiooni väärtust (*cross-correlation value*), mille abil hinnatakse nende sarnasust. Raamatus *"Digital Signal Processing"* on välja toodud järgmine valem, mille abil arvutatakse normaliseeritud rist-korrelatsiooni kordajat:

$$\gamma(x,y) = \frac{\sum_s \sum_t [w(s,t) - \bar{w}] \sum_s \sum_t [f(x+s, y+t) - \bar{f}(x+s, y+t)]}{\sqrt{\sum_s \sum_t [w(s,t) - \bar{w}]^2 \sum_s \sum_t [f(x+s, y+t) - \bar{f}(x+s, y+t)]^2}},$$

kus f on pilt, w on mall, $\bar{f}$ ja $\bar{w}$ tähistavad vastavalt pildi ja malli keskmist väärtust, mis arvutatakse ainult korra, summeerimise limiidid võetakse f ja w poolt jagatud ala järgi ja $\bar{f}(x+s, y+t)$ on pildi keskmine väärtus alas, mis on malliga ühine. Tulemuseks saadav kordaja on raamatu järgi vahemikus $[-1, 1]$. Joonisel 8 on kujutatud mall suurusega $m \times n$, mille keskpunkt on parasjagu punktis (x, y), pärast seda, kui selles punktis on eelnevalt välja toodud valemi järgi rist-korrelatsiooni kordaja välja arvutatud, siis liigutatakse mall edasi punkti $(x+1, y+1)$ jne.



Joonis 8. Rist-korrelatsiooni arvutamine [4]

Kõigist maatriksis olevatest rist-korrelatsiooni väärtustest valitakse maksimaalne ja kui see on suurem kui fikseeritud lävend, siis loetakse provints tuvastatuks, vastasel juhul mitte. Kui provints on tuvastatud, hakatakse otsima märke alalt, mille laius on [l,213], kus l on oletatav piksel, millest algab numbrimärgi unikaalne identifikaator. Tekitatakse 31 x 163 maatriks, kus igas reas on rist-korrelatsiooni väärtus, mis on arvutatud vastava malli ja selle iga võimaliku positsiooni jaoks antud reas. Kuue märgi täpse positsiooni saamiseks võetakse kõigepealt igast veerust kõige suurema rist-korrelatsiooni väärtusega mall, et saada nimekiri võimalikest lahenditest. Seejärel rakendatakse teadmist, et märkidevaheline kaugus numbrimärgil on 15-19 pikslit ja hinnanguline märgi suurus on 20 x 17 pikslit, nende konstantide abil valitakse välja kuuene komplekt märke, millel on suurim rist-korrelatsiooni väärtus ja mille omavaheline kaugus on sobivas vahemikus. Lõplik tulemus on see, mille märkide keskmine rist-korrelatsiooni väärtus on suurim.

3 Optilise märgituvastuse algoritmi rakendamine numbrimärgituvastuse jaoks

Peatükis kirjeldatakse autori poolt tehtud tööd automaatse numbrimärgituvastuse jaoks mõeldud optilise märgituvastuse algoritmi kirjutamisel. Peatükis selgitatakse töö käiku, ning lõpus antakse ülevaade tulemustest.

3.1 Töövahendid ja meetodite valik

Märgituvastuse algoritmi rakendamiseks kasutatakse Pythonit ja OpenCV teeki. OpenCV on avatud lähtekoodiga raalnägemise ja masinõppe tarkvara teek, mis sisaldab nii klassikalisi kui ka tänapäevaseid algoritme raalnägemise ja masinõppe kasutamiseks programmides [6]. OpenCV teegis olemasolevad algoritmid, mida autori poolt kasutatakse, töötavad nii, nagu eelmises peatükis kirjeldati, kui ei ole öeldud teistmoodi. Järnevalt tuuakse välja kasutatud meetodeid ja põhjendatakse nende valikut. Meetodeid valides ja algoritmi töös üldiselt eeldatakse sisendiks saadava pildi kohta kolme asja: numbrimärk koosneb mustadest numbritest valgel taustal, pildil on võrdlemisi ühtlane valgustus ja pilt on tehtud numbrimärgile suhteliselt lähedalt. Algoritmi kirjeldus on jaotatud sarnaselt etappidega eelmise peatüki optilise märgituvastuse algoritmi kirjeldamisel.

3.1.1 Pildi saamine

Alustuseks loetakse sisse standardsetes mõõtmetes horisontaalse asetusega pilt RGB formaadis ja teisendatakse see halltoonidesse. Seda seetõttu, et mõned järgnevalt pildile rakendatavad funktsioonid nõuavad ühekanalilist sisendpilti. Kui kohe see teisendus ära teha, siis ei ole vaja hiljem järge pidada, millised funktsioonid seda nõuavad ja kas sisend on õiges formaadis. Teiseks põhjuseks on see, et ühekanalilised pildid võtavad vähem ruumi, kuna kolme kanali asemel on neil vaja salvestada infot ainult ühe kohta. Etapi tulemusena on saadud horisontaalse asetusega halltoonides pilt.

3.1.2 Pildi eeltöötlemine

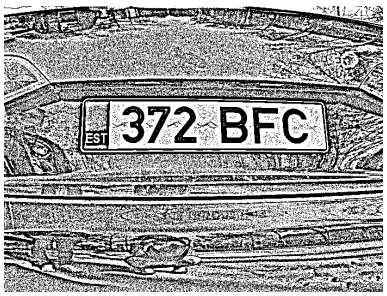
Selle etapi eesmärgiks on töödelda pilti selliselt, et numbrimärk oleks pildilt võimalikult hästi tuvastatav.
Kõigepealt rakendatakse pildile Gaussi hägustamist, et eemaldada pildilt müra,

mis hiljem pildiga töötamist segama hakkab. Filtri suuruseks valitakse 9, sest selgus, et kuna loomulikult on numbrimärgi serv väga selgepiiriline, siis rakendades suuremat filtrit, hakkab see numbrimärgi äärt liiga palju ümbritseva värvi sisse sulatama ja hiljem binaarseks muundamist rakendades on numbrimärgi serv ebaühtlane (joonised 9.1 ja 9.2).

Joonis 9.1. Gaussi filter suurusega 15×15

Joonis 9.2. Gaussi filter suurusega 9×9

Järgmiseks rakendatakse pildile binaarseks muundamist. Selle eesmärgiks on numbrimärk tugevalt ümbritsevast esile tuua. Alguses prooviti kasutada lokaalset binaarseks muutmist, selgus aga, et selle tulemusena jääb pildile palju müra, mida ei ole vaja ja seetõttu otsustati globaalse binaarseks muutmise kasuks (joonised 10.1 ja 10.2). Globaalse binaarseks muutmise eeliseks on siin see, et pildil on vaid üks objekt, mida on vaja säilitada valgena - numbrimärk. Kui valida globaalseks lävendiks toon, mis on mõned astmed madalamal numbrimärgi tausta toonist, on tulemuseks see, et numbrimärgi taust on valge ja numbrimärgi ümbrus must. Töös valiti binaarseks muundamise lävendiks konstant 145, sest katsetamisel selgus, et see sobib enamusele piltidest olenemata sellest, mis värvi on numbrimärgi ümbrus. Probleemid võivad tekkida vaid siis, kui auto on väga heledat tooni.



10.1. Lokaalne binaarseks muutmine

10.2. Globaalne binaarseks muutmine

Selle etapi tulemuseks on binaarseks muudetud pilt, millest on väga lihtne gradienti analüüsides numbrimärgi ääri tuvastada. Järgnevalt on välja toodud esimese kahe etapi kood.

```
def preprocess_image(image):
   img = cv2.imread(image)
   gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
   blur = cv2.GaussianBlur(gray,(9,9),0)
   ret,th = cv2.threshold(blur,145,255,cv2.THRESH_BINARY)

   return th, gray
```

3.1.3 Numbrimärgi asukoha tuvastamine

Eelmises etapis saadud pildile rakendatakse kõigepealt Laplace'i operaatorit, mis numbrimärgi ääred veel selgemalt välja toob ning seejärel üritatakse tuvastada numbrimärgi ääri OpenCV `findContours` funktsiooniga. Raamatu *"Practical OpenCV"* [7] põhjal on see funktsioon OpenCV rakendus Suzuki ja Abe poolt kirjeldatud algoritmist [8], mis tuvastab hulga ühendatud punkte, mis asuvad äärtel ja paigutab nad hierarhiasse. Hierarhiasse paigutamise loogika seisneb selles, et üks kontuur loetakse teise "vanemaks", kui see ümbritseb teist kontuuri. Tulemuseks saadud puu on raamatu järgi organiseeritud nii, et kontuurid, mis asuvad puus kõrgemates tippudes on suurema tõenäosusega pildil asuvate objektide piirjooned ja madalamal asuvad kontuurid on müra, mis tekib hägustest äärtest või muudest pildi soovimatutest omadustest.
Tavaliselt ei ole etapi sisendiks antud pildil kogu numbrimärgi ümbrus vaatamata binaarseks muundamisele ühtlaselt must, heledad laigud võivad tekkida näiteks päikesekiirte peegeldumisest auto värvil või suvalisest objektist taustal. Seetõttu leiab funktsioon `findContours` lisaks numbrimärgi äärtele veel ka teisi objekte, mis ei ole tegelikult huvipakkuvad (joonis 11).

Joonis 11. Punasega on märgitud leitud ääred

Üheks eelduseks, mida sisendiks oleva pildi kohta peatüki alguses mainiti, oli see, et pilt on tehtud numbrimärgile suhteliselt lähedalt. Seega võib arvata, et numbrimärgi ääri tähistav kontuur on kõigi funktsiooni poolt leitud kontuuride

hulgast suurima pindalaga. Selle põhjal võib kõigi kontuuride hulgast välja jätta kõigepealt need, mis ei moodusta kinnist joont. Seejärel arvutatakse ülejäänud kontuuride poolt moodustatud kujundite pindalad ja valitakse neist suurim, mis peaks vastama numbrimärgile. Selle etapi jaoks kirjutatud funktsioon Pythonis on järgmine:

```
def find_subimage(image):
   laplacian = cv2.Laplacian(image,cv2.CV_8U,4)
   _, contours, hierarchy =
       cv2.findContour(laplacian,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)

   #filter contours based on max(area) = licence plate
   areas = {}

   for contour in contours:
      area = cv2.contourArea(contour)
      areas[float(area)] = contour

   contour = areas[max(areas)]
   return contour
```

3.1.4 Numbrimärgi eraldamine pildist

Selle etapi sisendiks on eelmises etapis tuvastatud numbrimärgi kontuuri koordinaadid. Kuna tulemuseks saadud kontuur ei pruugi olla korrektne nelinurk, siis esimese asjana üritatakse kontuuri üldistada kujundisse, millel on vähem tippe. Selleks on OpenCV teegis olemas funktsioon `approxPolyDP`, mis kasutab OpenCV dokumentatsiooni[9] põhjal Ramer-Douglas-Peuckeri algoritmi [10]. Selle algoritmi tööpõhimõte seisneb veebilehe [11] järgi selles, et see tekitab kõigi punktide hulgas, mis moodustavad joonisel kumeruse joone esimese punkti a ja viimase punkti b vahel. Lehel kirjeldatakse, et seejärel kontrollib algoritm, kui suur on kaugus joone ja joonest kõige kaugemal asuva kumeruse peal oleva punkti vahel. Kui kaugus on väiksem kui ε, siis kaotatakse kõik vahepealsed punktid kumerusel ja tulemuseks on sirge joon kumeruse ühest otspunktist teise. Kui aga mingi kumerusel asuva punkti c ja joone vaheline kaugus on suurem kui ε, siis jagatakse kumerusel asuvad punktid kahte hulka $\{a..c\}$ ja $\{c+1..b\}$. Seejärel kutsutakse veebilehel kirjutatu järgi algoritmi rekursiivselt mõlema hulga peal välja ja kui töö nende hulkadega lõpeb, pannakse kummagi hulga vähendatud kumerusega jooned jälle kokku tagasi.
Selle algoritmi kasutamiseks arvutatakse alustuseks arv ε, mis sõltub numbrimärki ümbritseva piirjoone pikkusest ning seejärel rakendatakse piirjoonele eelpool nimetatud funktsiooni `approxPolyDP` (joonised 12.1 ja 12.2).

Joonis 12.1. Enne piirjoone üldistamist

Joonis 12.2 Pärast piirjoone üldistamist

Nagu teiselt pildilt näha, ei pruugi tulemuseks saadud piirjoon samuti ristkülik olla, seetõttu kontrollitakse tavaliselt pärast piirjoone üldistamist algoritmides, kas saadud piirjoonel on 4 nurka. Kui ei ole, siis loetakse numbrimärk pildilt eraldamatuks ja algoritm tööd ei jätka. Pärast mõnda aega kestnud testimist märkas autor, et probleemid nurkade tuvastamisega esinevad ainult numbrimärgi vasakul küljel. See on tingitud sellest, et eesti numbrimärgi vasakul küljel on sinine ala, mille ülemises ääres on Euroopa Liidu logo ja alumises ääres valgete tähtedega Eesti tähis (joonis 13).

Joonis 13. Eesti numbrimärk

Probleem seisnes täpsemalt selles, et algoritm üritas tihti ka Eesti tähist ülejäänud valge numbrimärgi tausta hulka lisada ja seetõttu ei olnud numbrimärgi vasak külg alati sirge. Selle asemel, et selliselt tuvastatud äärtega numbrimärgid kohe kõrvale jätta, kontrollitakse, kui suur on vahe vasakpoolse ja parempoolse ülemise ja vasakpoolse ja parempoolse alumise tipu vahel. Kui tippude erinevuseks on rohkem kui 10 pikslit, siis üritatakse vastavalt kas ülemine või alumine vaskapoolne tipp viia õigesse kohta kasutades alumise või ülemise tipu x-koordinaadi väärtust ja kõrvaloleva tipu y-koordinaadi väärtust.
Järgmiseks arvutatakse tippude poolt moodustatud numbrimärgi piirjoone laius ja kõrgus. Seda infot kasutatakse, et panna paika lõuendi tippude koordinaadid, kuhu algselt pildilt tuvastatud numbrimärk eraldatakse. Seejärel moodustatakse OpenCV funktsiooniga `getPerspectiveTransform` maatriks, mille abil algselt pildilt huvipakkuv osa lõuendile teisendatakse. Funktsiooni poolt leitav maatriks T vastab tingimusele

$$\begin{pmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{pmatrix} = T \cdot \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix},$$

kus lõuend $dst(i) = (x'_i, y'_i)$, algne pilt $src(i) = (x_i, y_i)$ ja $i = 0, 1, 2, 3$ [9]. Selline maatriks T on lineaarteisendus ja selle elemendid on võimalik leida lahendades võrrandisüsteem. Leitud maatriksit T kasutatakse seejärel funktsioonis `warpPerspective`,

mis teisendab huvipakkuva piirkonna algselt pildilt *src* lõuendile *dst*. Seda teeb funktsioon kasutades järgmist maatriksit:

$$dst(x, y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right)$$

Selle etapi tulemusena saavutatakse pilt, millel on algsest pildist eraldatud numbrimärk (joonised 14.1 ja 14.2)

Joonis 14.1. Algne pilt

Joonis 14.2. Eraldatud numbrimärk

Selle etapi jaoks kirjutatud funktsioon Pythonis näeb välja järgmine, autor on kasutanud siin lõike Artjom Lindi jt poolt koostatud Sudoku lahendaja programmist [12]:

```
def extract_subimage(image, contour):
   epsilon = 0.04 * cv2.arcLength(contour, True)
   approximation = cv2.approxPolyDP(contour, epsilon, True)
   box = np.array([item[0].tolist() for item in approximation])

   points = [(p[0], p[1]) for p in box]

   tl = min(points, key=lambda x:x[0] + x[1])
   br = max(points, key=lambda x:x[0] + x[1])
   tr = max(points, key=lambda x:x[0] - x[1])
   bl = min(points, key=lambda x:x[0] - x[1])

   if (br[1]-bl[1])>9:
      bl = (tl[0],br[1])
   if (tl[1]-tr[1]) > 9:
      tl = (bl[0],tr[1])

   #compute width
   width_a = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
   width_b = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
   max_width = max(int(width_a), int(width_b))

   #compute height
   height_a = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
   height_b = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
   max_height = max(int(height_a), int(height_b))

   # destination points array (top-down view) of the image, specifying
   # points in the top-left, top-right, bottom-right, and bottom-left
       order
   dst = np.array([[0, 0],[max_width - 1, 0],
      [max_width - 1, max_height - 1],[0, max_height - 1]], dtype =
         "float32")
   points = np.array([tl, tr, br, bl], dtype = "float32")

   #compute the perspective transform matrix
   perspective_transformation_matrix =
       cv2.getPerspectiveTransform(points, dst)
   #separate licence plate from image
   warped = cv2.warpPerspective(image, perspective_transformation_matrix,
       (max_width, max_height))
   return warped
```

3.1.5 Numbrite tuvastamine

Selles etapis üritatakse tuvastada märke eelmise etapi tulemusena sisendpildist eraldatud numbrimärgilt. Programmi eesmärgiks on tunda ära märke Eesti numbrimärkidelt, mis on standardse fondiga, samuti esitati peatüki alguses nõue, et sisendpilt peab olema tehtud suhteliselt lähedalt, võttes lisaks arvesse ka seda, et tänapäevased kaamerad on väga hea kvaliteediga, otsustati kasutada mustrisüsteemi.
Mustrisüsteemi kasutamiseks on vaja malli, eesti numbrimärkidel kasutatakse DIN 1451 fondi digitaalset versiooni FF DIN [14]. Mall oli võimalik koostada digitaalsete fontide tootja FontShop kodulehel [15] (joonis 15).

Joonis 15. Numbrite mall

Selleks, et numbrimärgi pildil olevaid sümboleid malliga võrrelda, peavad need olema sama suured. Selleks leitakse pildi kõrguse ja pikkuse suhe ning malli kõrgus ja numbrimärgi pilti suurendatakse või vähendatakse nii, et selle kõrgus on sama mis mallil. Seejärel rakendatakse numbrimärgi pildile taas Gaussi hägustamist ja binaarseks muundamist nagu esimeses etapis, aga seekord valitakse Gaussi filtri suurus väiksem kui esimeses etapis, sest kuna pilt on väiksem, siis suur filter võib numbrimärgil olevaid numbreid liiga palju tausta sisse sulatada. Malli töödeldakse samamoodi nagu numbrimärgi pilti ning alustatakse märkide tuvastamisega. Selleks kasutatakse OpenCV funktsiooni `matchTemplate` ja piltide sarnasuse võrdlemiseks rist-korrelatsiooni väärtust, mis OpenCVs arvutab sarnasust sama valemi järgi, mis teises peatükis välja toodi. Tulemuste kontrollimiseks joonistatakse numbrimärgi pildile kast selle koha ümber, kus funktsioon kõige kõrgema sarnasuse leidis. Tulemus, mis saadi võrreldes numbrimärki ja joonisel 15 toodud mallilt number kolme, on toodud joonisel 16.

Joonis 16. Pildilt tuvastatud märk

Nagu näha, on tuvastamine olnud edukas ning number kolm on pildilt tuvastatud. Nii käiakse läbi kõik numbrid ja tähed ning lõpuks valitakse välja kombinatsioon nendest sümbolitest, mille rist-korrelatsiooni väärtus on kõige suurem. Selle etapi jaoks kirjutatud funktsioon Pythonis näeb välja järgmine:

```
def character_recognition(image, template):

   template = cv2.imread(templates,0)
   template = cv2.GaussianBlur(template,(5,5),0)
   ret,template = cv2.threshold(template,170,255,cv2.THRESH_BINARY)

   w = template.shape[1]
   h = template.shape[0]

   img_ratio = int(image.shape[1]) / int(image.shape[0])
   image = cv2.resize(image, (int(img_ratio*template.shape[0]),
       template.shape[0]))
   blur = cv2.GaussianBlur(image,(5,5),0)
   ret,th = cv2.threshold(blur,160,255,cv2.THRESH_BINARY)

   #pattern matching
   res = cv2.matchTemplate(th,template,cv2.TM_CCORR_NORMED)
   min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

   #draw recognised area
   top_left = max_loc
   bottom_right = (top_left[0] + w, top_left[1] + h)
   cv2.rectangle(image,top_left, bottom_right, 50, 2)
   cv2.imshow('warped', image)
   cv2.waitKey(0)

   return max_val
```

3.1.6 Tulemused

Kirjutatud algoritm suudab märke tuvastada võrdlemisi hästi. Enamusel katsetatud piltidest suutis algoritm tuvastada kõik märgid õigesti. Negatiivne külg rist-korrelatsiooni väärtuse järgi sümbolite võrdlemisel on see, et tingituna valemist, mille järgi seda arvutatakse, võib ka sümbol, mida tegelikult pildil ei ole, saada kõrge rist-korrelatsiooni väärtuse, kui pildil on väga heledaid kohti. See probleem tuli välja ka programmi testides: märgid, mida numbrimärgis tegelikult ei sisaldunud, paigutati kas numbrimärgi vasakusse äärde või numbrite ja tähtede vahelisele valgele alale. Selliselt paigutatud märkide rist-korrelatsiooni väärtus oli suur, mõnel korral isegi suurem, kui õigesti tuvastatud märgil. See teeb nende vigade tuvastamise ja selliste märkide kõrvalejätmise keeruliseks. Vea vältimiseks tuleks kasutada teistsugust märgituvastuse meetodit.

4 Kokkuvõte

Bakalaureusetöös kirjeldati, kuidas optilise märgituvastuse algoritm töötab ning rakendati ühte meetodit optiliseks autode numbrimärkide tuvastamiseks. Meetodi rakendamisel selgus, et kindla eesmärgi jaoks mõeldud optilise märgituvastuse algoritmi on võrdlemisi lihtne rakendada. Selle jaoks on oluline teada, mida erinevad pilditöötlusfunktsioonid täpselt teevad, et neid parimal viisil kasutada.
Töö eesmärk on anda ülevaade sellest, optilisest märgituvastusest, et tudengitel ja teistel teemast huvitatud inimestel võimalik lugeda sellest teema kohta eesti keeles. Töö autor mõistis seda tööd koostades, et eestikeelset materjali raalnägemise kohta ei ole eriti võimalik leida. Töö on terviklik ja sobiv materjal teemaga tutvumiseks.

Kirjandus

[1] E. R. Dougherty, R. A. Lotufo, *"Hands-on Morphological Image Processing,"* Bellingham: SPIE–The International Society for Optical Engineering, 2003.

[2] L. Tan, *"Digital Signal Processing: Fundamentals and Applications,"* California: Academic Press, 2007.

[3] Wikipedia artikkel "YUV", *https://en.wikipedia.org/wiki/YUV,* vaadatud 18.04.

[4] R. C. Gonzalez, R. E. Woods, *"Digital Image Processing,"* New Jersey: Prentice Hall, 2008.

[5] P. Comelli, P. Ferragina, M. N. Granieri, F. Stabile, *"Optical recognition of motor vehicle license plates,"* IEEE Trans. Veh. Tech., vol. 44, no. 4, pp. 790–799, 1995.

[6] OpenCV koduleheküIg, *http://opencv.org,* vaadatud 11.05.

[7] S. Brahmbhatt, *"Practical OpenCV,"* New York: Apress, 2013.

[8] S. Suzuki, K. Abe, *"Topological structural analysis of digitized binary images by border following,"* Computer Vision, Graphics, and Image Processing, vol. 30, no. 1, pp. 32-46, 1985.

[9] OpenCV dokumentatsioon, *http://docs.opencv.org,* vaadatud 11.05.

[10] D. H. Douglas, T. K. Peucker, *Älgorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature,"* Cartographica The International Journal for Geographic Information and Geovisualization, vol. 10, no. 2, pp. 112-122, 1973.

[11] Marius Karthausi koduleheküIg, *http://karthaus.nl/rdp/*, vaadatud 11.05.

[12] A. Lind, M. Kamalov, R. Ots, V. Looga, *SudokuOCR - Project Report (MTAT.03.183 Data Mining),* 2010.

[13] K. Safronov, I. Tchouchenkov, H. Wörn, *"Optical Character Recognition Using Optimisation Algorithms,"* Workshop on Computer Science and Information Technologies CSIT'2007, Russia, 2007.

[14] Wikipedia artikkel "DIN 1451", *https://en.wikipedia.org/wiki/DIN_1451,* vaadatud 11.05.

[15] FontShopi koduleheküIg, *https://www.fontshop.com*, vaadatud 11.05.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Anni Uutma (sünnikuupäev: 20.07.1994),

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose:

1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni,

"Ülevaade optilise märgituvastuse meetoditest",

mille juhendaja on Artjom Lind

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartu, 12.05.2016