TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

IDN70LT

Keili Oras132336 IAPMM

# KAUBAMÄRKIDE REPRODUKTSIOONIDE KONTENTANALÜÜS

magistritöö

Juhendaja: Tarmo Veskioja
Doktori kraad
Teadur

Tallinn 2016

# Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Keili Oras

08.05.2016

# Annotatsioon

## Kaubamärkide reproduktsioonide kontentanalüüs

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 63 leheküljel, 6 peatükki, 40 joonist, 8 tabelit. Käesolevas töös uuritakse kaubamärgi reproduktsioonide sisulise võrdlemise metoodikaid. Kirjeldatud metoodikad realiseeritakse näidisprogrammis ning esitatakse loodud programmi testide tulemused ja nende põhjal tehtud järeldused.

# Abstract

Content Analysis of Trademarks Reproductions

The thesis is in Estonian and contains 63 pages of text, 6 chapters, 40 figures, 8 tables. The work researches the methodologies of content based comparison of trademarks reproductions. These methodologies are realized in a demonstration program, tested and presented along with test results and conclusions.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| EPA | Eesti Patendiamet |
| Isik | Antud dokumentatsioonis nimetatakse isikuks taotluse esitajat. |
| Patendivolinik | Patendivolinik on taotleja või omaniku esindaja. Õigusaktide kohaselt võib juriidilist või füüsilist isikut esindada üksnes patendivolinik. Patendivolinik saab esitada avalduse füüsilise isiku või ettevõtte nimel ning selle ise allkirjastada. Välismaistel isikutel, kelle puudub alaline elu- või asukoht Eestis, on kohustuslik omada patendivolinikku. |
| Patendiameti ametnik | isik, kes teostab Patendiametis kaubamärgile menetlust või ekspertiisi |
| Kaubamärgi reproduktsioon | kaubamärgi graafiline kujutis |
| Kaubamärgi ekspertiis | kaubamärgi menetlemise osa, mille käigus teostatakse muuhulgas graafilise kujutise äravahetamise tõenäosuse kontroll |
| Kaubamärgi ristklass | kauba- ja teenusklassidega samaliigiliseks osutuda võiv kauba- ja teenusklass |
| WIPO | Ülemaailmne Intellektuaalsete Omandite Organisatsioon |
| EUIPO | Euroopa Liidu Intelleketuaalomandi Amet |
| Acsepto | Eesti Patendiameti poolt ekspertiisi teostamiseks kasutatav otsinguprogramm |
| RGB-mudel | liitvärvi mudel, milles erinevaid värvitoone saadakse kolme põhivärvuse (punane, roheline, sinine) liitumisel |
| K-NN | masinõppel põhinev karakteristikute klassifitseerimise algoritm |
| Lõpptulemus (testide) | Näidisprogrammiga teostatud testide koondtulemus ehk näidisotsingu efektiivsuse hinnang |

# Sisukord

# Jooniste loetelu

# Tabelite loetelu

# 1 Sissejuhatus

Käesoleva lõputöö eesmärk on välja selgitada, et kuidas on võimalik parendada Eesti Patendiameti kaubamärgi ekspertiisi tööprotsessi, kasutades reproduktsiooni äravahetamise tõenäosuse kontrollimiseks piltide analüüsimisel põhinevat otsingut.

Hetkel on välja pannud avalikuks kasutamiseks pildipõhised kaubamärgi reproduktsiooni otsingud WIPO ja EUIPO. Eesti Patendiameti eksperdid teevad ekspertiisi käigus otsinguid EUIPO ning EPA andmebaasidesse. EPA kaubamärgi andmebaasis[1] pildipõhine otsing puudub ning EUIPO otsing[2] pole Eesti Patendiameti jaoks piisavalt usaldusväärne, et seda oma igapäevatöös kasutada.

Lõputöö esimeses osas antakse lühiülevaade kaubamärkidest, nende liikidest ja klassifikatsioonidest. Samuti kirjeldatakse, kuidas toimub Eesti Patendiametis kaubamärgi reproduktsiooni äravahetamise tõenäosuse kontroll praegu.

Käesoleva lõputöö teises osas kirjeldatakse võimalikke metoodikaid piltide kontekstanalüüsil põhineva kaubamärkide reproduktsioonide otsinguks. Samuti tuuakse ülevaade kaubamärgi reproduktsioonide võrdlemise näidisprogrammist, kus on realiseeritud kirjeldatud metoodikaid ning näidisprogrammiga tehtud testid.

Töö lõpus tuuakse välja lõputöö käigus tehtud järeldused, edasiarendamise võimalused ja kokkuvõte.

[1] http://www2.epa.ee/Patent/mark.nsf/SearchEst?OpenForm

[2] https://euipo.europa.eu/eSearch/

# 2 Kaubamärgid

Kaubamärk on graafiliselt kujutatav tähis, mille abil on võimalik eristada kaupu ja teenuseid. Kaubamärgi abil peab tarbija aru saama, et toodet või teenust pakub mingi kindel ettevõtja. Kaubamärgi abil ei saa kaitsta näiteks kaupade ja teenuste sisu või kaupade tootmise või teenuse osutamise viisi, vaid ainult graafiliselt kujutatavat tähist, mis eristab ühe toote või teenuse teisest sarnasest tootest või teenusest [1].

Kaubamärgi põhiliigid on [1]:

- sõnamärk – tavapärases arvutikirjas sõna või sõnade kombinatsioon, täheühend või tähtede ja numbrite kombinatsioon;
- kujutismärk – koosneb ainult kujundist ning ei sisalda sõnu, tähti ega numbreid;
- kombineeritud märk – sõnad, tähed või numbrid koos kujundi või kujundite kombinatsiooniga. Kujunduseks loetakse ka omapärast kirjaviisi;
- ruumiline märk – Kolmemõõtmeline tähis, mille kohta saab esitada mitu vaadet;
- helimärk – heli või meloodia, mida on võimalik kujutada noodikirjas;

Kaubamärgi eriliigid on [1]:

- kollektiivkaubamärk – isikute ühendusele kuuluv tähis, mida iga liige võib kasutada vastavalt põhikirjas[1] sätestatud tingimustele;
- garantiimärk – eri isikute kaupu või teenused tähistav märk nende kaupade või teenuste ühise omaduse, geograafilise päritolu, tootmisviisi või muu ühistunnuse garanteerimiseks;

[1] https://www.riigiteataja.ee/akt/110012012012?leiaKehtiv

Käesolev lõputöö keskendub kujutismärkide ja kombineeritud märkide reproduktsiooni äravahetamise tõenäosuse kontrollimisele.


Joonis 1. Näide kujutismärgist (a) ning kombineeritud märgist (b).

## 2.1 Kaubamärgi klassifikatsioonid

Kaupade ja teenuste klassifitseerimine on selleks, et võrrelda taotlust eelnevate kaubamärgitaotluste ja –registreeringutega, millega on soovitud kaitset identsetele või samaliigilistele kaupadele ja teenustele [1].

Nizza klassid lisab taotlusele taotleja ning kujundelementide koodid lisab kaubamärgile EPA ametnik, kes teostab kaubamärgi menetlust.

### 2.1.1 Nizza klassifikatsioon

Kõik kaubad ja teenused on rahvusvahelise süsteemi järgi jagatud Nizza klassidesse[1]. Süsteemis on 45 klassi: klassid 1-34 on kaupade jaoks ning 35-45 teenuste kohta. Kaubad on jagatud klassidesse peamiselt materjali, teenused aga tegevusala või eesmärgi järgi [1]. hetkel kehtib kaubamärkide registreerimisel Nizza klassifikatsiooni 10. redaktsiooni 2016. aasta versioon.[2]

### 2.1.2 Kujundelementide klassifikatsioon

Viini klassifikatsioonil [3]põhineva kujundelementide klassifikatsiooni abil tehakse hetkel Eesti Patendiametis otsinguid sarnaste kujundelementide otsimiseks. Kujundelementide

[1] http://www.wipo.int/classifications/nice/en/

[2] http://www.epa.ee/sites/www.epa.ee/files/elfinder/dokumendid/nizza10-2016klass.pdf

[3] http://www.wipo.int/classifications/vienna/en/

klassifikatsioon on jagatud 29 põhikategooriaks, mis koosneb omakorda alamkategooriatest [1].

## 2.2 Kaubamärgi reproduktsiooni äravahetamise tõenäosuse kontrollimine Eesti Patendiametis

Enne kaubamärgi äravahetamise tõenäosuse kontrolli teostamist määrab EPA ametnik kaubamärgile kujundelementide koodid (Vt. ptk. 2.1.2). Otsingu teostamiseks kasutab Eesti Patendiamet programmi nimega Acsepto[1]. Acsepto on tasuline programm, mis võimaldab teha kaubamärgi reproduktsiooni otsingut eelkõige kujundelementide koodide järgi. Acsepto müüb ka pildituvastuse alusel töötavat otsingut, millest kõneleb käesolev lõputöö, kuid kõrge hinna ja väikese usaldusväärsuse tõttu ei ole EPA seda moodulit endale soetanud.

Käesoleval hetkel toimub kaubamärgi äravahetamise tõenäosuse kontroll nii, et EPA ametnik sisestab kaubamärgi Nizza klassid ning kujundelementide koodid menetlustarkvarasse. Acsepto võtab menetlustarkvarast vajalikud andmed ning ekspert koostab nende andmetega asjakohase otsingu. Acsepto kuvab vastusena kõik reproduktsioonid, mis sisaldavad otsingus olevaid klasse ja koode. Kuna kehtib seadus, et registreeritud kaubamärk ei tohi olla identne või ära vahetamiseni sarnane varasema märgiga, mis on saanud õiguskaitse identse või samaliigiliste kaupade või teenuste suhtes, siis teostab ekspert otsingu ka ristklassi sisenditega.

Olnud uurinud kaubamärgi reproduktsiooni äravahetamise tõenäosuse kontrolli toimimist Eesti Patendiametis, märkas lõputöö autor kahte protsessi aeglustavat kohta, kus on ka võimalus vigade tekkele:

1. Kaubamärgi kujundelementide koodid sisestab ametnik. Kuna ametnik on inimene, mitte masin, siis on alati võimalus, et üks ametnik saab reproduktsiooni sisust aru ühte moodi ja sisestab ühed klassid ning teine ametnik veidi teistsugused.

[1] http://intellect.sword-group.com/Home/Acsepto

2. Ekspert teostab otsingu käsitsi sisestatud kujundelementide koodide põhjal, millega ta arvab kaubamärgi reproduktsiooni sarnanevat. Ka siin on võimalus, et mõni klass jääb märkamata, kuna käsitsi toimides pääsevad EPA ametnikud ligi ainult nendele klassidele ja koodidele, mida otsingusse sisestavad. Seega võib mõni klass või kood inimlikust eksimusest märkamata jääda.

Kontekstanalüüsil põhineva kaubamärgi reproduktsiooni otsingu eesmärk on ära kaotada kujundelementide klasside kasutamine, kuid Nizza klassid jäävad ka sellisel juhul kasutusse, kuna kaubamärgid, mis ei kuulu samasse klassi või samasse ristklassi, võivad sisuliselt sarnaneda.

## 2.3 Reproduktsiooni äravahetamise tõenäosuse määramise kriteeriumid

Kaupade ja teenuste samaliigilisuse tuvastamisel tuleb arvestada seda, et ei saa lähtuda vaid identsest sõnastusest, vaid tuleb arvestada ka kaupade ja teenuste sisulist kokkulangevust [1]. Kaubamärgi sarnasuse kohta ei ole kaubamärgi seaduses sõnastatud otsest reeglit, et millised kaubamärgid on sarnased ja millised mitte. Kaubamärkide sarnasust tuleb igal konkreetsel juhul lähtudes võrreldavatest märkidest hinnata ja põhjendada. Kaubamärgi seaduses[1] on kirjas, et õiguskaitset ei saa kaubamärk, mis on varasema kaubamärgiga identne või ära vahetamiseni sarnane. Kaubamärgi seadus ütleb, et kaubamärk on identne või sarnane juhul, kui on tõenäoline kaubamärkide ära vahetamine tarbija poolt identsete või samaliigiliste kaupade või teenuste tähistamisel.

Hetkel kasutatakse kaubamärkide sarnasuse hindamisel järgmiseid üldiseid hindamise põhimõtteid:

- arvestatakse kaubamärgi üldmuljet, mitte üksikuid detaile;
- vaadatakse kaubamärgi domineerivaid, meeldejäävaid ja eristatavaid osasid;
- arvestatakse visuaalset, foneetilist ja semantilist aspekti;

[1] https://www.riigiteataja.ee/akt/128122011004?leiaKehtiv#para10

Toon välja Eesti Patendiameti kodulehel [1] esitatud näited liiga sarnastest kaubamärgi reproduktsioonidest (Joonis 2).


Joonis 2. Näited liiga sarnastest kaubamärkide reproduktsioonidest.

Selles peatükis väljatoodud äravahetamise tõenäosuse määramise kriteeriumid ei ole ainukesed, mida Eesti Patendiamet kaubamärgi reproduktsioonide võrdlemisel kasutab. Lõputöö autoril puudus ülevaade kõikidest võrdlemise kriteeriumitest, mida EPA eksperdid ekspertiisi läbiviimisel kasutavad. Seda põhjusel, et need hindamiskriteeriumid on väga suhtelised ja neid ei ole isegi kaubamärgi ekspertidele kuhugile kirja pandud. Kaubamärgi reproduktsiooni äravahetamise tõenäosuse määramise oskus kujuneb ajas läbi eksperdi töö praktika, tööstusomandi apellatsioonikomisjoni ja kohtute praktika. Seega kasutas lõputöö autor informatsiooni, mida ta leidis Eesti Patendi ameti kodulehelt [1] ning Kaubamärgi seadusest[1].

Uurides kaubamärgi reproduktsiooni äravahetamise tõenäosuse määramise kriteeriumeid selgus, et ei ole olemas üheselt mõistetavaid ärireegleid, mis sätestaksid, et millised reproduktsioonid on omavahel ära vahetamiseni sarnased ja millised mitte - ehk tegemist on ekspertide süsteemiga. Seda väidet toetab tõdemus, et EPA eksperdid ei pea olemasolevaid kujutisel põhinevaid reproduktsiooni võrdlemise otsinguid piisavalt

[1] https://www.riigiteataja.ee/akt/KaMS

usaldusväärseks, kuna nad ei ole teadlikud, milliste äriliste kriteeriumite põhjal see programm vastuseid annab. Eesti Patendiameti ekspertide sõnul ei ole veel teada, kuidas saada pildipõhise kaubamärgi otsinguga ammendav tulemus. Lõputöö autor on arvamusel, et EPA-s kehtivate kaubamärgi reproduktsiooni äravahetamise tõenäosuse määramise põhimõtete alusel tuleb välja töötada täpsed ärireeglid, mida on võimalik kasutada pildipõhises ekspertsüsteemis. Välja tuleb töötada subjektiivsete hinnangute põhise süsteemi asemel reeglipõhine ekspertsüsteem. Kuna taoliste ärireeglite väljatöötamine on mahukas ning vajab vastavaid õigusalaseid teadmisi, siis antud lõputöö keskendub rohkem otsingu teostamise metoodikatele ning toob välja vaid üldised reeglid.

### 2.3.1 Reeglid reproduktsioonide unikaalsuse kontrollimisel pildipõhises otsingus

Kuna hetkel puuduvad piisavalt täpsed unikaalsuse kriteeriumid, mille abil saaks luua pildipõhist otsingut nii, et tulemused oleksid juriidiliselt Eesti Patendiameti jaoks õiged, siis järgnevalt toob lõputöö autor välja mõned kriteeriumid, mida ta kasutab enda loodud näidisotsingus:

- Reproduktsioonide **värvide** võrdlemine (Vt. ptk. 2.3.1.1);
- Reproduktsioonide **tekstide** võrdlemine – lõputöö autor on arvamusel, et kaubamärgi reproduktsioonilt tuleb leida kõik tähe- ja numbrimärgid ning arvutada protsentuaalne kokkulangevuse koefitsient. Leitud koefitsient näitab, et kui palju kaubamärgi reproduktsiooni tekstist langeb kokku võrreldava reproduktsiooni tekstiga;
- Reproduktsioonidel olevate **kujundite** võrdlemine – lõputöö autor on arvamusel, et reproduktsiooni kujundite sisulise kokkulangevuse hindamiseks tuleb võrrelda nii üksikuid detaile kui ka üksikutest detailidest moodustunud suuremaid detaile;

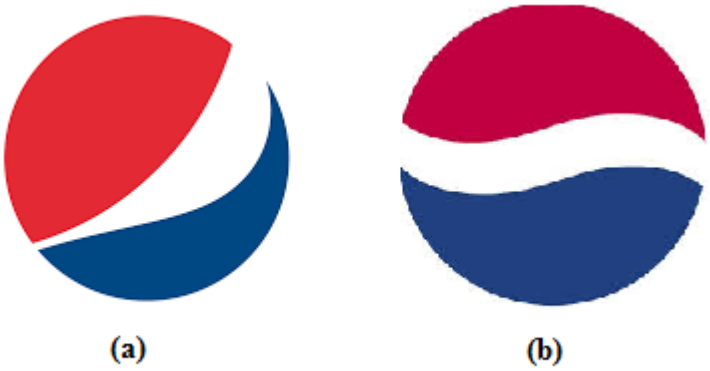
#### 2.3.1.1 Reproduktsioonide värvide võrdlemine

Lõputöö autor on arvamusel, et kui kahel, samalaadsesse kauba-või teenusklassi kuuluval kaubamärgil on sama suur hulk põhivärve ning need kattuvad, siis võib see juba olla tarbija jaoks eksitav sarnasus. Seega tuleb leida igalt reproduktsioonilt põhilised toonid ning kui kattuvad nii põhitoonide arv kui ka toonid, siis tuleb otsingutulemustes arvestada

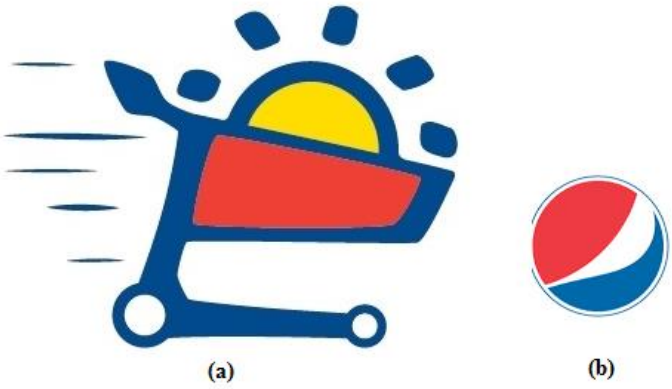
ka värvide kattuvust. Joonistel 3 ning 4 on toodud näited kaubamärkide reproduktsioonide värvide võrdlemisel põhinevast sarnasuse leidmisest.

Põhitoonideks loetakse teostatud näidisprogrammis (Vt. ptk 4) järgmiseid värve:

- must;
- sinine;
- roheline;
- punane;
- lilla;
- kollane;




Joonis 3. Näide: Reproduktsioonide (a) ja (b) korral tuleb otsingu tulemuse määramisel arvestada ka värvide kokkulangevust, sest mõlemal kujutisel on põhitoonideks sinine ja punane.




Joonis 4. Näide: Reproduktsioonide (a) ja (b) korral ei tohi arvestada lõpptulemuse määramisel värvide kokkulangevust, sest reproduktsioonil (a) on kolm põhitooni (kollane, sinine ja punane), kuid reproduktsioonil (b) on kaks põhitooni (punane ja sinine).

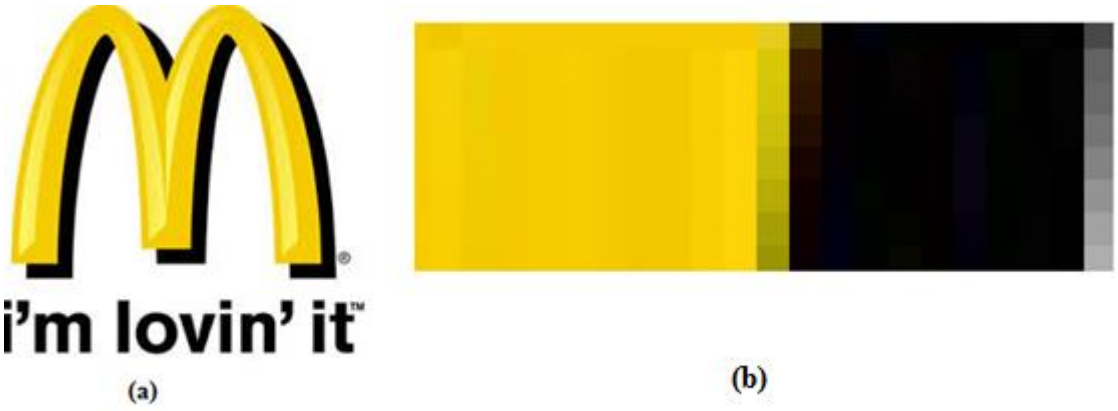
# 3 Piltide sisulise võrdlemise metoodikad

Järgnevalt kirjeldatakse järgmiseid piltide sisulise võrdlemise metoodikaid:

- Värvitoonide võrdlemine (Vt. ptk 3.2);
- Teksti tuvastamine (Vt. ptk. 3.3);
- Kujundite sisuline võrdlemine (Vt. ptk 3.4);

## 3.1 Värvitoonide võrdlemine

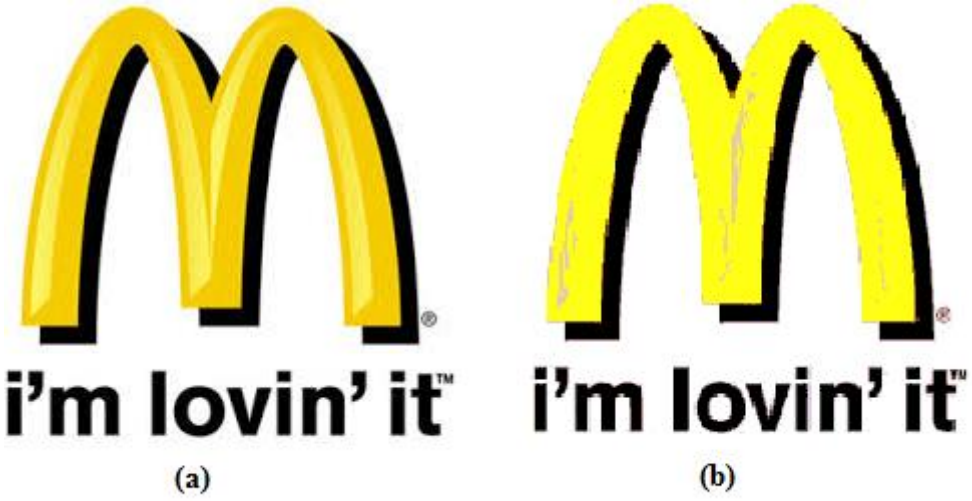
Kahe pildi värvitoonide võrdlemine ei ole triviaalne ülesanne. Pildilt, millel on väga palju erinevaid värvitoone, eristab inimene pilti normaalsuuruses vaadates vaid teatud värve ning erinevad toonid tulevad nähtavale alles pilti oluliselt suurendades.



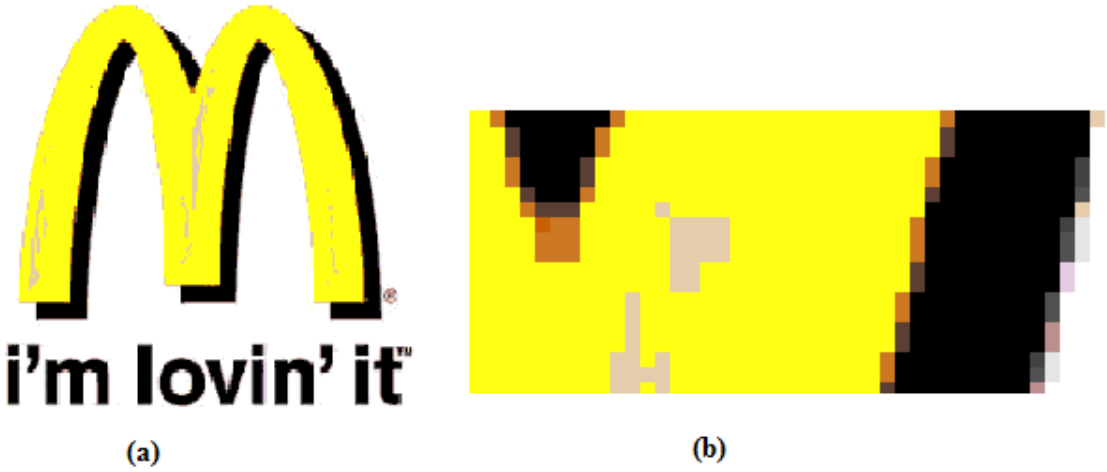

Joonis 5. Näide: Pilt (a) ning suurendatud vaade samast pildist (b).

Joonisel 5 on näha, et kui vaadata pilti normaalsuuruses, siis eristab inimese silm vaid kollast ning musta tooni, kuid tegelikult koosneb see pilt väga erinevat värvi pikslitest, näiteks kollased, pruunid, mustad ja hallid.

Selleks, et arvuti saaks inimesega võimalikult sarnaselt pildi värvidest aru, kasutab lõputöö autor järgmist metoodikat. Programmile antakse ette piiratud hulk värvitoone. Enne kui programm hakkab vaatama piksleid värvide tuvastamiseks, saab iga piksel uue värvi. Uus värv saadakse nii, et võetakse lähim toon uute värvide hulgast, millega originaalpildil olev piksli värv kõige enam sarnaneb.




Joonis 6. Näide: Originaalpilt (a) ning uus pilt (b), kus värvid on muudetud vastavalt uutele etteantud värvitoonidele.




Joonis 7. Näide: Pilt (a) ning suurendatud vaade pildist.

Nii jooniselt 6 kui ka jooniselt 7 on näha, et ka muudetud värvitoonidega reproduktsiooni pildil ei ole ainult kollased ja mustad värvid. Seega loetakse kokku pikslite arv, mis ei ole valged ning lahutatakse see kogu pikslite arvust. Tulemuseks saadakse värviliste pikslite arv. Järgmisena leitakse kõik erinevad toonid, mis on pildil ning pikslite arv, mis on nende toonidega kaetud. Iga värvi tulemus jagatakse kõikide värviliste pikslite arvuga ning saadakse tulemuseks konkreetse värvi pildil esinemise protsent. Valem (1) esitab näite tehtavast arvutusest, kus m tähistab konkreetstet värvitooni, pm on värvi m protsentuaalne esinemise sagedus pildil, nm on värvi m pikslite arv, n on kõikide pikslite arv ning nw on valgete pikslite arv.

$$pm = \frac{\mathrm{nm}}{\mathrm{n-nw}} \qquad (1)$$

Joonise 9-1 kohta saadakse valemi (1) järgi järgmine tulemus:

- musta värvi on 30%;
- kollast värvi on 60%;
- pruuni värvi 1%;
- punast värvi 3%;

Saadud tulemuse järgi saab välja sorteerida väikese protsentuaalse tulemusega värvid ning järele jäävad soovitud põhitoonid. Antud metoodika on lõputöö autori enda välja pakutud ning see annab enamike piltide korral hea tulemuse. Parema tulemuse saamiseks on vajalik toonide ülemineku põhjalikum analüüs.

## 3.2 Teksti tuvastamine

Pildilt teksti tuvastamine ja tähemärkidest aru saamine on üks visuaalse tuvastuse põhiprobleeme. Pildilt teksti tuvastamise keerukus seisneb põhiliselt keeruliste kirjastiiliga tekstide tuvastamises [14]. Kuigi pildilt teksti tuvastamiseks on mitmeid algoritme, on see teema, millele pole siiani ideaalset lahendust leitud.

Teksti tuvastamiseks kasutatakse OCR meetodit. OCR (*Optical Character Recognition*) on trükitud või käsitsi kirjutatud tähtede, numbrite või sümbolite teisendamine masinloetavale kujule. Järgnevalt tuuakse välja kaks võimalikku teksti tuvastamise lähenemist [5] :

- **maatriksite ehk mustrite sobitamine** – põhineb pildi osadeks jagamisel, mille korral tähemärgid eraldatakse ülejäänud pildist ning neid võrreldakse piksli haaval etteantud tähemärkidega (Vt. ptk 3.2.1);

- **masinõppel põhinev teksti tuvastamine** –tähemärkidest saadud karakteristikute vektoreid võrreldakse eelnevalt õpetatud tähemärkide karakteristikute vektoritega (Vt. ptk 3.2.2);

### 3.2.1 Mustrite sobitamise põhine teksti tuvastamine

Järgnevalt kirjeldatakse ühte võimalikku teksti tuvastamise algoritmi pildil oleva teksti muutmiseks masinloetavale kujule. Antud algoritmi tugev külg on see, et pildi eeltöötlusele ei kulu palju aega. Kirjeldatav algoritm koosneb järgnevatest osadest:

- pildi eeltöötlus (Vt. ptk 3.2.1.1);

- karakteristikute eraldamine (Vt. ptk. 3.2.1.2);

- mustrite sobitamine (Vt. ptk. 3.2.1.3);

- saadud tulemuste töötlemine – kuna tekstituvastus ei anna üldjuhul üheselt õiget tulemust, siis tehakse antud lõputöö näidisprogrammis tuvastatud tekstidele sarnasuse koefitsiendi leidmiseks järel töötlust (Vt. ptk 3.2.1.4);

#### 3.2.1.1 Eeltöötlus

Selleks, et maatriksite võrdlemisel põhinev tekstituvastus annaks häid tulemusi, tuleb teisendada pilt binaarkujule, milles valgete pikslitega kujutatavaid alasid hakatakse võrdlema etteantud šabloonidega.

Enne mustripõhist tekstituvastus tehakse pildiga üldjuhul järgmised teisendused:

- Pildi teisendamine hallideks toonideks (Vt. ptk 3.2.1.1.1);
- Hallides toonides pildi teisendamine binaarkujule (Vt. ptk 3.2.1.1.2);

On võimalik teostada ka keerulisemaid teisendusi näiteks pildi tausta eemaldamine[1] ning suure intensiivsusega kohtade välja toomine[2]. Joonistelt 8 ja 9 on näha, et keerulised eeltöötluse tehnikad ei pruugi anda paremaid tulemusi.


Joonis 8. Näide. Pilt (a) teisendati hallideks toonideks ning viidi binaarkujule, saadi tulemus (b).


Joonis 9. Pilt (a) teisendati hallideks toonideks, eemaldati taust, toodi esile tugevama intensiivsusega kohad ning teisendati siis binaarkujule, saadi tulemus (b).

[1] http://se.mathworks.com/help/images/examples/correcting-nonuniform-illumination.html

[2] http://se.mathworks.com/help/images/ref/imreconstruct.html

#### 3.2.1.1.1 Pildi teisendamine hallideks toonideks

Pildi teisendamisel hallideks toonideks muudetakse kõik pildil olevad nähtavad toonid erinevateks hallideks varjunditeks. Kõige tumedam halli varjund on must ning heledaim on valge. Vahepealsed halli varjundid väljendavad kolme põhivärvi (punane, roheline ja sinine) heleduse astet [7].

RGB-mudelis tähistab värvilise pildi korral iga piksel kolme põhitooni (punane, roheline ja sinine) heleduse astet kas arvuga vahemikus 0-255 või binaarkujul esitatuna 00000000 kuni 11111111. Näiteks punase värvi korral R=255, G=0 ja B=0, kuid oranžika värvi korral on liidetud ka rohelist tooni ehk R=255, G=128 ja B=0 [7].

Halli piksli varjundi arvutamiseks on mitmeid erinevaid valemeid ning see, millist valemit on mõistlik rakendada, sõltub soovitud tulemusest. Hallide pikslite väärtuseid saab arvutada näiteks valemitega (2), (3) ja (4) [8].

$$Gray = 0.2989 * R + 0.5870 * G + 0.1140 * B \quad (2)$$

$$Gray = (R + G + B)/3 \quad (3)$$

$$Gray = (R * 0.3) + (G * 0.59) + (B * 0.11) \quad (4)$$


Joonis 10. Näide. Pilt (a) koosneb neljast kollakast pikslist, kus kõigi nelja piksli värvitoonid moodustuvad RGB-mudeli järgi järgmiselt R=255, G=195 ja B=36 ning pilt (b) arvutati valemi (2) järgi, kus iga piksel omab nüüd ümardatud väärtust 195.

#### 3.2.1.1.2 Pildi teisendamine binaarkujule

Kui pilt on teisendatud binaarkujule, siis on pildil olevatel pikslitel ainult kaks erinevat võimalikku väärtust. Tavaliselt valitakse nendeks toonideks must ja valge, kuid tegelikult võivad olla selleks mistahes kaks erinevat tooni [9]. Pildi viimiseks binaarkujule on erinevaid meetodeid, kuid järgnevalt kirjeldatakse hallides toonides oleva pildi viimist binaarkujule.

Selleks, et viia hallides toonides pilt binaarkujule, tuleb valida heleduse lävi, millest heledamad toonid saavad väärtuse 1 ning tumedamad väärtuse 0. Pikslite heledus arvutatakse järgmise valemi (5) järgi [10]. Näiteks kuna joonisel 10 pildi (b) viimisel binaarkujule kui lävi on 0.5 ning heleduse aste on kõikidel pikslitel 0.8, saavad kõik neli pikslit uue väärtuse 1.

$$R' = \frac{R}{255}$$

$$G' = \frac{G}{255}$$

$$B' = \frac{B}{255}$$

$$lightness = \max(R', G', B') \quad (5)$$

Heleduse läve arvutamiseks kasutatakse antud töös Otsu' metoodikat [24]. Selleks leitakse pildi sagedusjaotuse tulpdiagramm (Joonis 11). Leitud tulpdiagramm kirjeldab hallides toonides pildi väärtuste sagedust. Joonise 11 näitel saab öelda, et pildil on kõige rohkem valgeid toone ning vahepealsete hallide toonide sagedus jääb vahemikku 0 kuni 2000 pikslit. Otsu metoodika töötab nii, et leitakse $\sigma_\omega^2$ iga võimaliku läve $x_i$ korral, kus $x_i$ on pildil olev halli tooni väärtus. Iga $x_i$ korral arvutatakse $\sigma_\omega^2$ (7), kus $k_1$ ja $k_2$ sisaldavad väärtuseid, mis jäävad kas alla või üles poole vaadatavat lävi. Valem (5) leiab vastava läve korral pikslite hulga protsentuaalse väärtuse, kus $n_i$ on vahemikku kuuluvate pikslite arv ning N on kõikide pikslite arv. Valem (6) leiab vastava k korral pikslite hulga protsentuaalse väärtuse, kus $n_{i_2}$ on vastavasse k vahemikku jääva konkreetse väärtuse pikslite hulk ning $N_2$ on kõiki vahemikku k jäävate pikslite hulk. Lõpptulemuseks määratakse $\sigma_\omega^2$, mille väärtus on kõikide võimalike $x_i$ väärtuste korral kõige väiksem.

$$p_{i_1} = n_i / N \quad (5)$$

$$p_{i_2} = n_{i_2} / N_2 \quad (6)$$

$$\omega_{k_1} = \sum_{i=1}^{k_1} p_i$$

$$\omega_{k_2} = \sum_{i=1}^{k_2} p_i$$

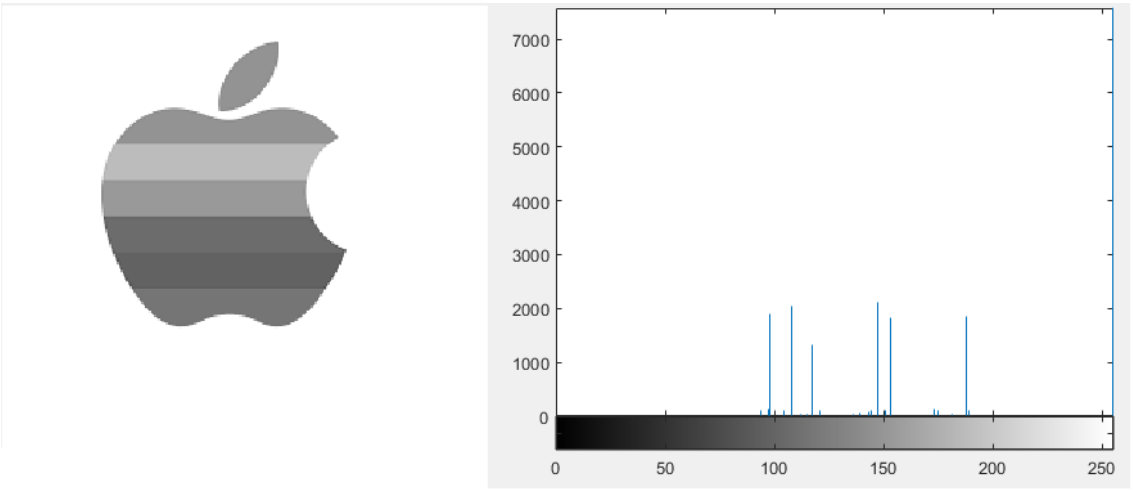
$$\mu_{k_1} = \sum_{i=1}^{k_1} ip_i$$

$$\mu_{k_2} = \sum_{i=1}^{k_2} ip_i$$

$$\sigma_{k_1}^2 = \sum_{i=1}^{k_1} (i - \mu_{k_1})^2 p_i$$

$$\sigma_{k_2}^2 = \sum_{i=1}^{k_2} (i - \mu_{k_2})^2 p_i$$

$$\sigma_{\omega}^2 = \omega(k_1)\sigma_{k_1}^2 + \omega(k_2)\sigma_{k_2}^2 \quad (7)$$




Joonis 11. Näide. Pilt (a) ja selle sagedusjaotuse tulpdiagramm (b).

#### 3.2.1.2 Karakteristikute eraldamine mustrite võrdlemisel põhinevas teksti tuvastuses

Karakteristikute eraldamine mustrite võrdlemisel põhinevas teksti tuvastuses on protsess, mille käigus eraldatakse teksti tuvastusele ette antavad tähemärgid. Järgnevalt kirjeldatakse ühte võimalust karakteristikute eraldamiseks binaarkujul olevast pildist:

- ridade põhine karakteristikute eraldamine (Vt. ptk. 3.2.1.2.1);

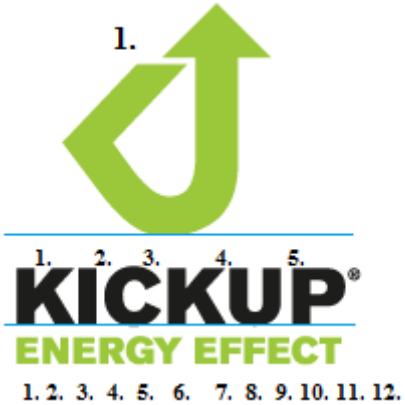
- märgistuste põhine karakteristikute eraldamine (Vt. ptk. 3.2.1.2.2);

Mõlema metoodika korral viiakse leitud karakteristikud kindlasse mõõtu, et neid oleks võimalik võrrelda etteantud tähemärkide maatriksitega. Lõputöö autor on arvamusel, et parema tulemuse saab nimetatud kahte metoodikat kombineerides, sest nagu on näha joonisel 12, siis ei pruugi ridade põhine karakteristikute leidmine alati head tulemust anda. Nimetatud näites leitakse tähemärgid vales järjekorras, kuna eeldatakse, et tegemist on ühe, mitte kahe reaga. Kõige õigem oleks eraldada enne teksti tuvastust teksti sisaldavad piirkonnad ning kujutisi sisaldavad piirkonnad, kuid kuna nimetatud alamülesanne on keeruline ning vajab põhjalikku analüüsi, siis antud lõputöö sellele ei keskendu.


Joonis 12. Näide. Kaubamärgi reproduktsioon sisaldab sageli nii teksti kui ka graafilist kujutist, kusjuures graafiline kujutis võib asuda teksti suhtes erinevates kohtades.

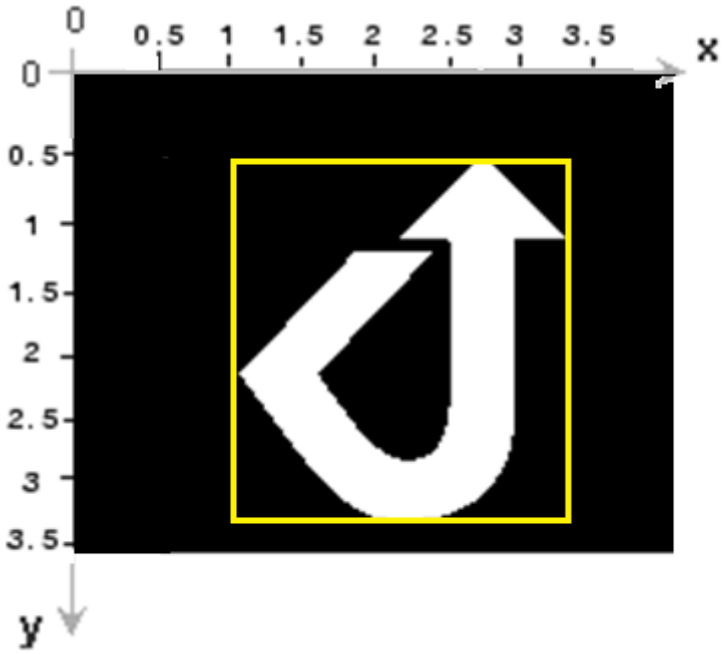
#### 3.2.1.2.1 Ridade põhine karakteristikute eraldamine

Ridade põhise karakteristikute eemaldamise eesmärk on eraldada pildil oleva teksti read ning igalt realt eraldi tuvastada real olevad karakteristikud (Joonis 13).




Joonis 13. Näide. Pildilt tuvastatud read on märgitud sinise joonega ning ridadelt tuvastatud karakteristikute tuvastamise järjekord on nummerdatud.

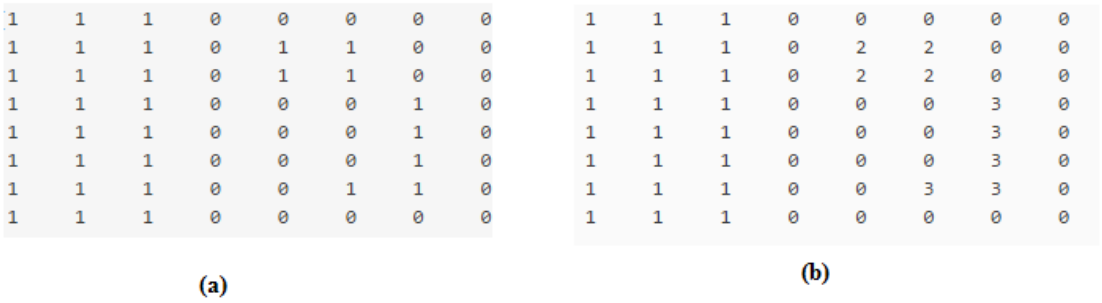
Enne rea tuvastamist lõigatakse pildilt välja see osa, mis sisaldab vajalikku infot ehk valgeid piksleid. Selleks leitakse kõikide pildil olevate valgete pikslite asukohad ning vaadatakse x ja y koordinaatide maksimaalselt ja minimaalselt väärtust (Joonis 14).



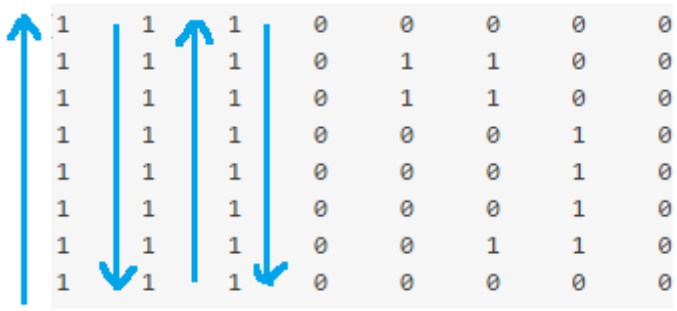

Joonis 14. Näide. y(min)=0.5, y(max)=3.3, x(min)=1, x(max)=3.3, seega vaadatakse ainult seda osa pildist kus 3.3<y<0.5 ning 1<x<3.3.

Järgnevalt vaadatakse läbi kõik välja lõigatud pildi pikslid. Kui ei leita ühtegi rida, kus kõik pikslid on nullid, siis kuvab reatuvastus vastusena selle sama kujutise ehk ühe leitud rea. Kui leitakse rida, kus on kõik nullid, siis kuvab reatuvastus vastusena leitud esimese rea ning ülejäänud pildi osa. Igal järgneval korral antakse reatuvastusele sisendiks alles jäänud pilt, millelt pole veel ridu tuvastatud ning seda protsessi korratakse rekursiivselt kuni kogu pilt on läbi vaadatud.

Iga kord kui leitakse rida, siis märgistatakse selle rea pildil olevad laigud ehk valgete pikslitega kaetud kohad. Märgistamine toimub nii, et käiakse läbi kõik read ning iga laigu pikslitele omistatakse erinev märgistuse number. Pikslite vaatamise järjekord on joonisel 15. Iga valge piksel saab sama väärtuse, mis on tema lähimatel valgetel pikslitel (Joonis 16). Kui valgete pikslite laigud on pildilt leitud, siis lõigatakse pildilt välja kohad, kust laigud leiti ning antakse need üks haaval sisendiks tähemärkide tuvastamiseks.




Joonis 15. Näide. Binaarkujul olevalt pildilt (a) on leitud 3 laiku, mis on märgistatud pildil (b).



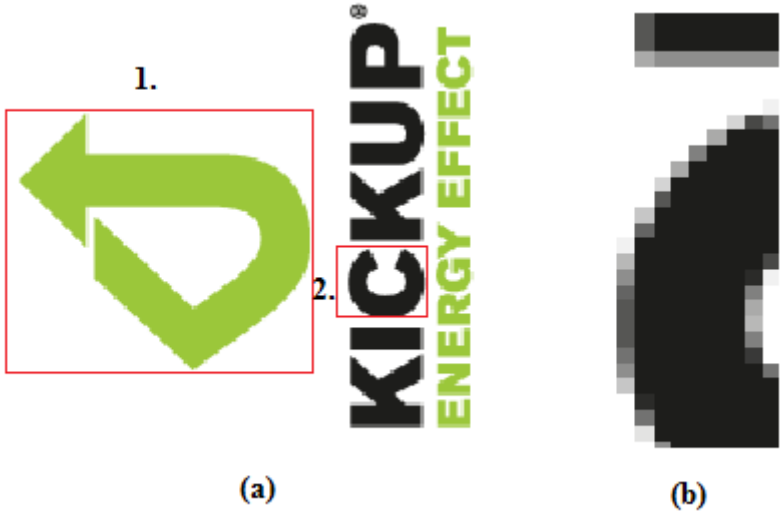

Joonis 16. Ridade läbimise järjekord valgete pikslitega kaetud laikude otsimisel.

#### 3.2.1.2.2 Märgistuste põhine karakteristikute eraldamine

Ka märgistuste põhisel karakteristikute eraldamisel tuleb leida ning ära märgistada valgete pikslitega kaetud kohad ehk laigud. Erinevus reapõhise lähenemisega seisneb selles, et ridu enam ei leita vaid märgistatakse kohe esimese asjana kogu sisendina tulnud binaarkujul olev pilt. Märgistuste järjekord on selle lähenemise korral ülioluline, kuna teksti tuvastuse sisendiks antakse karakteristikud vastavalt märgistuse järjekorra numbrile. Kuna ridade läbimist alustatakse alt vasakust nurgast , siis võib kergesti tekkida olukord, kus karakteristikud tuvastatakse vales järjekorras (Joonis 17).

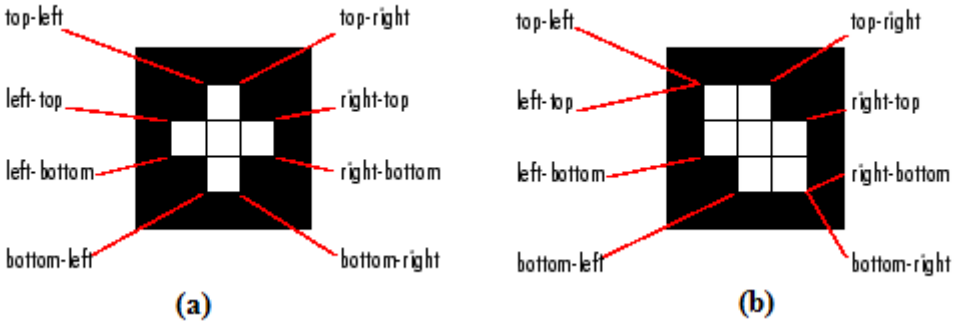

Joonis 17. Näide. Esimesena leitakse karakteristik „K“, järgmisena „E“, siis „N“ jne.

Loogiline samm on nüüd pöörata pilt teist pidi ja eeldada, et tulemus on parem. Tegelikkuses on endiselt järjekord vale (Joonis 18).



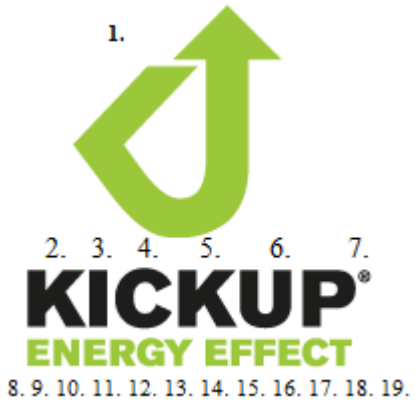

Joonis 18. Näide. Kui vaadata pilti lähemalt (b), siis selgub, et C täht on teistest veidi kõrgem ning seetõttu märgistatakse tuvastatakse see varem kui K ja I.

Parim lahendus on karakteristikute märgistused ära sorteerida [11]. Selleks leitakse iga karakteristiku alumise vasaku nurga koordinaadid ehk ekstreemumid (Joonis 19). Järgnevalt sorteeritakse märgistuste järjekorda nii, et uus järjekord on samasugune nagu leitud ekstreemumite sorteeritud järjestus.




Joonis 19. Näide karakteristikute (a) ja (b) ekstreemumitest.

Ka see lahendus ei märgista karakterisikuid alati õigesti ning antud teema vajaks rohkem analüüsi, kuid enamus piltide korral on tulemus hea (Joonis 20).

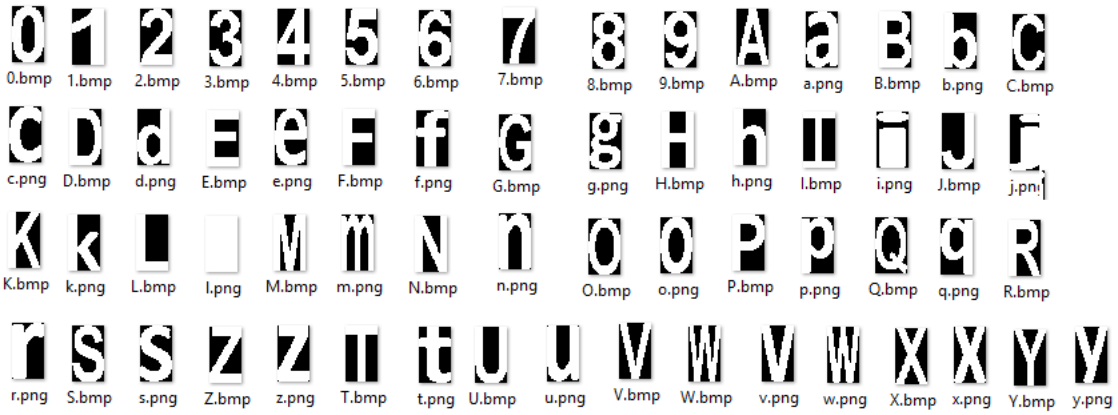



Joonis 20. Näide. Karakteristikute märgistamine pärast märgistuste tulemuste sorteerimist.

### 3.2.1.3 Tähemärkide tuvastamine ehk mustrite sobitamine

Tähemärkide tuvastamiseks kasutatakse mustrite võrdlemise korral ette antud kindlas mõõdus tähemärke ehk šabloone (Joonis 21). Iga leitud karakteristiku maatrikseid (Joonis 22) võrreldakse šabloonide maatriksitega [6]. Selleks leitakse leitud karakteristikute ja šabloonide maatriksite korrelatsiooni koefitsiendid. Korrelatsiooni koefitsient määrab kahe vektori kattuvuse. Valem (8) kirjeldab korrelatsiooni koefitsiendi leidmist, kus r on korrelatsiooni koefitsient, A ja B on võrreldavad pildid, $\bar{A}$ on A maatriksi keskmine väärtus, $\bar{B}$ on B maatriksi keskmine väärtus, m on ridade arv ning n on veergude arv (A ja B on sama suurusega) [12]. Mida suurem on korrelatsiooni koefitsient, seda sarnasemad on maatriksid - seega leitakse maksimaalne väärtus.

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \quad (8)$$




Joonis 21. Näide šabloonidest ehk ette antud tähemärkidest, millega pildilt leitud karakteristikuid võrreldakse.

Joonis 22. Näide. Ette antud tähemärgi ’A’ maatriks.

#### 3.2.1.4 Saadud tulemuste töötlemine

Teksti tuvastuse väljundeid üksüheselt võrreldes ei tuvastata üldjuhul sobivust põhiliselt kahel põhjusel:

- tähemärkide piirkondade määramine ei toimi alati nii nagu peab, seega antakse tekstituvastusele ette ka selliseid piirkondi, mis on sisuliselt graafilised kujutised, mitte tähemärgid. Kuigi näidisprogrammis on määratud maatriksite korrelatsiooni koefitsiendile lävi, milles ainult kõrgemaid väärtuseid vaadatakse, siis võib juhtuda, et graafiline kujutis aetakse segamini tähemärgiga;

- On teatud hulk tähe- ja numbrimärke, mis võivad erinevate fontide korral olla visuaalselt väga sarnased näiteks ’v’ ja ’u’, ’6’ ja ’G’ jne. Seega võib juhtuda, et teksti tuvastus ajab need tähed või numbrid omavahel segamini;

Seetõttu kasutab lõputöö autor näidisprogrammis saadud tulemuste töötlemist - tuvastatud tekstide sümbolite kattuvust võrreldakse alates kahest sümbolist. Leitakse numbriline väärtus, et mitu sümbolit järjest tekstis 1 sisaldub ka tekstis 2. Näiteks tekstide *„Kick Up“* ning *„Kick Up Energy Effect“* kokkulangevuse numbriline väärtus on 6, kuna väiksemast tekstist 6 sümbolit kattusid samas järjekorras suuremas tekstis olevate sümbolitega. Järgmisena leitakse sümbolite kattuvuse protsentuaalne väärtus jagades saadud number väiksema teksti pikkusega. Antud näite korral on saadud koefitsient 6/6 ehk 1, kuna kogu väiksem tekst sisaldub suuremas tekstis. Samas kui võrrelda üks ühele, kas antud tekstid kattuvad, siis sarnasust ei tuvastata. Võimalus oleks otsida ka ainult väiksema sõna sisalduvust suuremas sõnas terviklikult, kuid kuna tekstituvastus ei pruugi leida alati õiget tulemust ning eksitavalt sarnaseks võib muuta teksti ka osaline kattuvus, siis otsitakse väiksema sõna sisalduse protsenti suuremas sõnas. Kuna mõned numbri- ja tähemärgid võivad olla teiste numbrite- ja tähemärkide teksti tuvastuse jaoks liiga sarnased, siis teostatakse tekstide kattuvuse protsendi leidmise protsessi järgmiste sisenditega, kus min on väiksem tekst, max on suurem tekst, min_asendatud on väiksem tekst, kus on sarnased sümbolid asendatud (näiteks ’6’asemel ’G’) ning max_asendatud on asendatud pikem tekst:

- min, max;
- min_asendatud, max;
- min_asendatud, max_asendatud;
- min, max_asendatud;

Kirjeldatud tekstide järel töötluse tulemusena saadakse tekstide kokkulangevuse koefitsient, mis on protsentuaalne väärtus vahemikus nullist üheni. Mida kõrgem on tekstide kokkulangevuse koefitsient, seda sarnasemad on piltidel olevad tekstid.

### 3.2.2 Masinõppel põhinev teksti tuvastamine

See metoodika põhineb masinõppel ning on võimeline tuvastama ka keerulisemaid kirjastiile. Karakteristikute vektorite sobitamisel põhinev tuvastamine võib koosneda näiteks järgmistest osadest:

- **tähemärkide piirkondade tuvastamine** – on mitmeid erinevaid algoritme, mis proovivad seda ülesannet lahendada. Üks lähenemine on analüüsida pildil olevaid vertikaal- ja horisontaaljooni ehk servasid[1]. Samuti on võimalik leida pildil olevad read ning ridadel olevate karakteristikute märgistused (Vt. ptk 3.2.1.2). On mitmeid erinevaid lähenemisi teksti piirkondade tuvastamiseks, kuid kuna kaubamärgi reproduktsioonid on väga erinevad, sisaldades erinevaid kirjastiile ja graafilisi kujutisi, on lõputöö autor arvamusel, et ilmselt ei ole ühte universaalset algoritmi, mis leiab pildilt kõik tähemärgid ning jätab leidmata kõik need karakteristikud, mis ei ole tähemärgid. Antud alamülesanne on keeruline ning vajab põhjalikumat analüüsi - seetõttu jääb antud lõputöö skoobist välja.

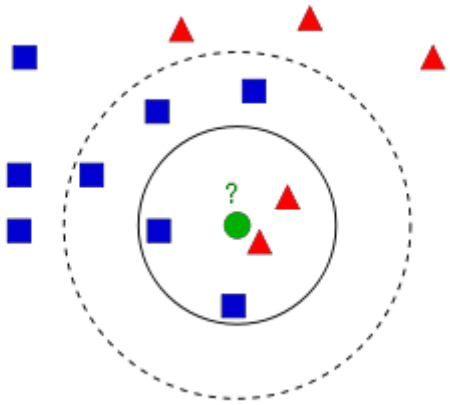
- **tähemärkide karakteristikute leidmine** (Vt. ptk. 3.2.2.2);

- **tähemärkide karakteristikute klassifitseerimine ja klassi kuuluvuse kontrollimine** – Antud lõputöös kirjeldatakse K-NN' i kui ühte võimalikku algoritmi tähemärkide karakteristikute klassifitseerimiseks ja klassi kuuluvuse kontrollimiseks (Vt. ptk 3.2.2.1)

#### 3.2.2.1 K-NN

K-NN *(k-Nearest Neighbours)* on masinõppel põhinev karakteristikute klassifitseerimise algoritm. k-NN'i eesmärk on otsida karakteristikute testandmete hulgast kõige lähedasemaid karakteristikuid [15]. K-NN'i kasutatakse klassifitseerimiseks ja regressiooniks. Teksti tuvastuses kasutatakse k-NN algoritmi klassifitseerimiseks. Selleks antakse algoritmile ette teatud hulk treeningandmeid (täpsemalt nende karakteristikuid) ning samuti uus klassifitseerimata sisend testimiseks. Eesmärk on leida uuele sisendile sobiv klass (Joonis 23). Karakteristikute hulka nimetatakse karakteristikute ruumiks, näiteks kahemõõtmelises karakteristikute ruumis on igal karakteristikul kaks mõõdet – x ja y koordinaadid.

[1] http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/moredge.html




Joonis 23. Näide. Pildil on kahte erinevasse klassi kuuluvad kujundid – sinised ruudud ja punased kolmnurgad ning KNN algoritmi eesmärk on tuvastada, et kumba klassi lisatakse roheline ring.

K-NN'i töötab erinevalt sõltuvalt parameetrist k. Järgnevalt kirjeldatakse algoritmi käitumist erinevate k väärtuste korral [16].

**1. k=1 ehk lähima naabri reegel**

Järgnevalt tuuakse näide joonise 23 põhjal. Oletame, et soovime leida, millisesse klassi kuulub roheline ring. Tema lähim naaber on punane kolmnurk, järelikult kui k=1, siis vastus on, et roheline ring kuulub punaste kolmnurkade klassi. Selline lähenemine ei anna head tulemust kui testandmete hulk on väike, kuid piisavalt paljude testandmete korral on suur võimalus, et leitakse õige klass [16]. Tulemuse sõltuvust testandmete hulga suurusest näitab valem (9), kus P* on Bayes' i veamäär, c on klasside hulk ning P on k-NN' i veamäär. Valemist (9) järeldub, et kui testklasside hulk on piisavalt suur, siis k-NN' i veamäär on alla kahe korra väiksem kui Bayes' i veamäär. Bayes'i veamäär kirjeldab tõenäosust, et kui objektid jaotatakse juhuslikult klassidesse, siis kui suur on võimalus, et objekt klassifitseeriti valesti. Bayes' i veamäär arvutatakse valemi (10) järgi, kus x on vaatlusalune objekt, $C_i$ on võimalikud klassid, $H_i$ on ala, mida klassifitseeriti kui klassi $C_i$.

$$P^* \leq P \leq P^*\left(2 - \frac{c}{c-1}P^*\right) \tag{9}$$

$$p = \sum_{C_i \neq C_{\max}} \int_{x \in H_i} P(x|C_i)p(C_i)\,dx, \tag{10}$$

**2. k>1 ehk k-lähima naabri reegel**

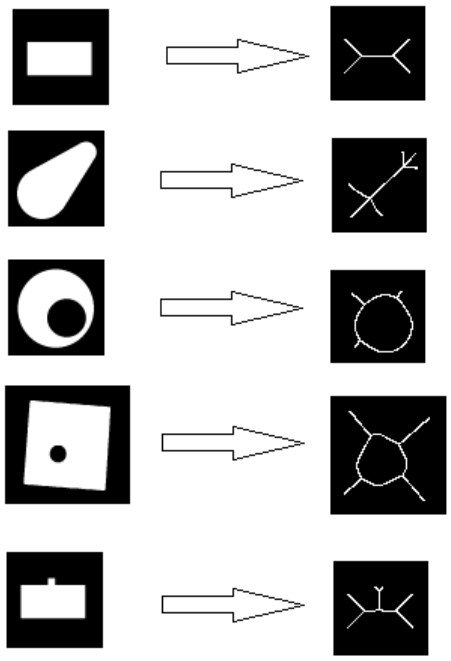
Järgnevalt tuuakse näide joonise 23 põhjal [16] . Kui k=3, siis on lähimad kaks punast kolmnurka ning üks sinine nelinurk ehk ka sellisel juhul lisatakse roheline ring punaste kolmnurkade klassi. Kui k=7, siis on lähimad kaks punast kolmnurka ning 5 sinist nelinurka, seega sellisel juhul lisatakse roheline ring siniste nelinurkadega samasse klassi. Seega k-NN' i klassifitseerimise tulemus sõltub oluliselt k väärtusest. Samuti on mõistlik valida k väärtuseks paaritu number, kuna näiteks kui k=4, siis on lähimad naabrid kaks punast kolmnurka ning kaks sinist nelinurka, seega jääb objekt klassifitseerimata.

**Modifitseeritud k-NN** algoritmi korral võetakse kasutusele koefitsient, mis määrab kui kaugel on teised objektid klassifitseerimise aluseks olevast objektist. Järgnevalt tullakse tagasi näite juurde, kus k=4. Tavalise k-NN algoritmi korral jäi rohelise ringi objekt klassifitseerimata, kuna nii nelinurki kui ka kolmnurki oli naabruses võrdselt. Kuid jooniselt 17 on näha, et kolmnurgad on ringile lähemal kui nelinurgad ning seda eelmine kord arvesse ei võetud. Modifitseeritud k-NN algoritm arvestab ka objektide vahelisi kauguseid ning sellisel juhul ei tekiks k=4 korral olukorda, kus ringi ei klassifitseerita, vaid roheline ring lisatakse punaste kolmnurkadega samasse klassi.

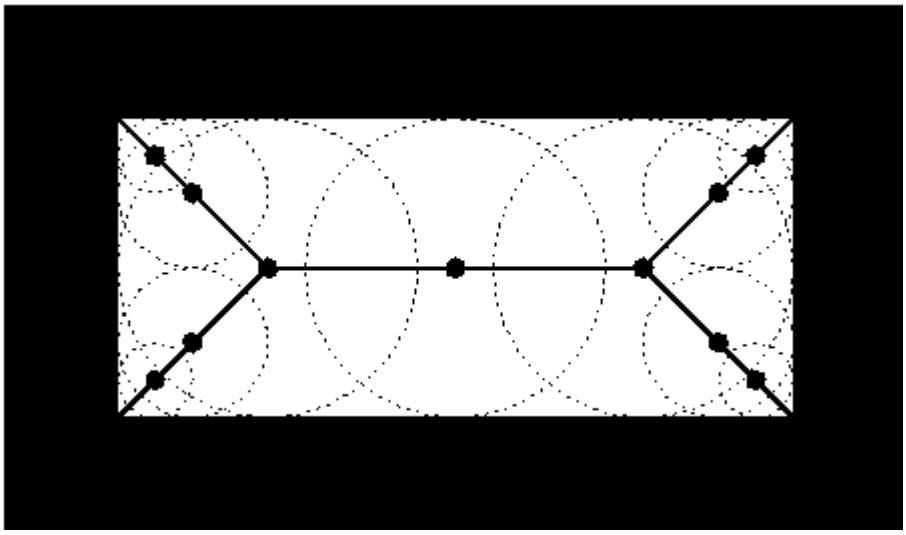
#### 3.2.2.2 Tähemärgi karakteristikute leidmine masinõppel põhinevas teksti tuvastuses

Karakteristikute leidmine on karakteristikute võrdlemisel põhinevas teksti tuvastamises kõige olulisem etapp. Et saada teksti tuvastamisel häid tulemusi, ei tohiks ühe tähemärgi karakteristikute hulk olla väga suur. Samuti on mõistlik arvutuslik keerukus madalana hoida. Järgnevalt kirjeldatakse tähemärgi karakteristikute leidmiseks ühte võimalikku algoritmi, mis leiab alati kindla hulka karakteristikuid ning mille arvutuslik keerukus ei ole suur [17].

Tähemärgi karakteristikute leidmise esimeses osas eraldatakse tähemärgi skelett (Joonis 24). Skeleti eraldamiseks nimetatakse protsessi, mille käigus vähendatakse binaarkujul olevas pildis esiplaanil olevaid piirkondi. Kui kujutisse, mille skeletti soovitakse leida, joonistada mõtteliselt maksimaalsete raadiustega ringid, siis nende ringide keskpunktidest moodustubki skelett (Joonis 25). Leides tähemärkide skeletid, kaovad ära varjud, mis võivad teksti tuvastust segada. Tähemärgist moodustunud skelett on sisendiks järgmisele karakteristikute leidmise etapile [17].



Joonis 24. Näide. Näiteid binaarkujul piltidest ja nende skelettidest.

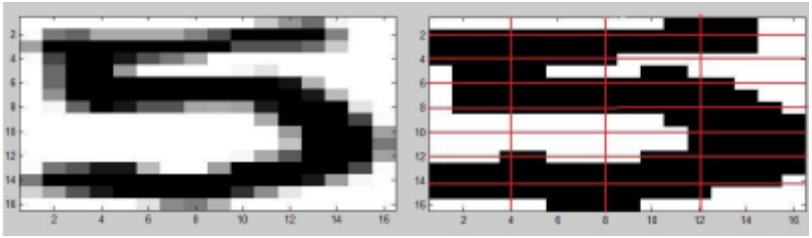


Joonis 25. Näide. Kujutiseks on valge ristkülik ning selle sees on maksimaalsete raadiustega ringid, mille keskpunktid moodustuvad ristküliku skeleti.

Teises etapis leitakse tähemärgi skeletist karakteristikud. Selleks leitakse kõige pealt pildi efektiivne osa ehk lõigatakse välja pildi ääred, mis ei sisalda vajalikku infot (Joonis 26). Seejärel jagatakse efektiivne ehk informatsiooni sisaldav pildi osa ristkülikukujulisteks osadeks (Joonis 27). Pildi jagamine osadeks nelja vertikaal- ning kaheksa

horisontaaljoonega annab kõige täpsema ja kiirema tulemuse [17] (Tabel 1). Iga ristkülikukujulise osa kohta leitakse pikslite keskmine väärtus. Tulemuseks saadakse 32 väärtusest ehk karakteristikust koosnev massiiv ehk karakteristikute vektor.


Joonis 26. Näide. Pildilt (a) eemaldatakse piirkonnad, mis ei sisalda vajalikku informatsiooni ning saadakse pilt (b). Sinised piirjooned tähistavad pildi äärt.



Joonis 27. Näide. Pildi efektiivne ehk infot sisaldav osa on jagatud 32 ristkülikukujuliseks osaks.

Tabel 1. Pildi erinevateks osadeks jaotamise efektiivsus masinõppel põhinevas tekstituvastuses

| Vertikaaljoonte arv | Horisontaaljoonte arv | Täpsus (%) | 2000 pildi töötlemise aeg (s) |
|---|---|---|---|
| 4 | 4 | 87.8 | 0.795 |
| 8 | 4 | 86 | 1.341 |
| 4 | 8 | 92.6 | 1.092 |
| 8 | 8 | 91.8 | 2.512 |

## 3.3 Kujundite sisuline võrdlemine

Kujundite sisulisel võrdlemisel arvestatakse antud töös kahte aspekti:

- Väikeste detailide kokku langemine (Vt. ptk 3.3.2);
- Väikestest detailidest moodustunud suuremate detailide kokku langemine (Vt. ptk. 3.3.3);

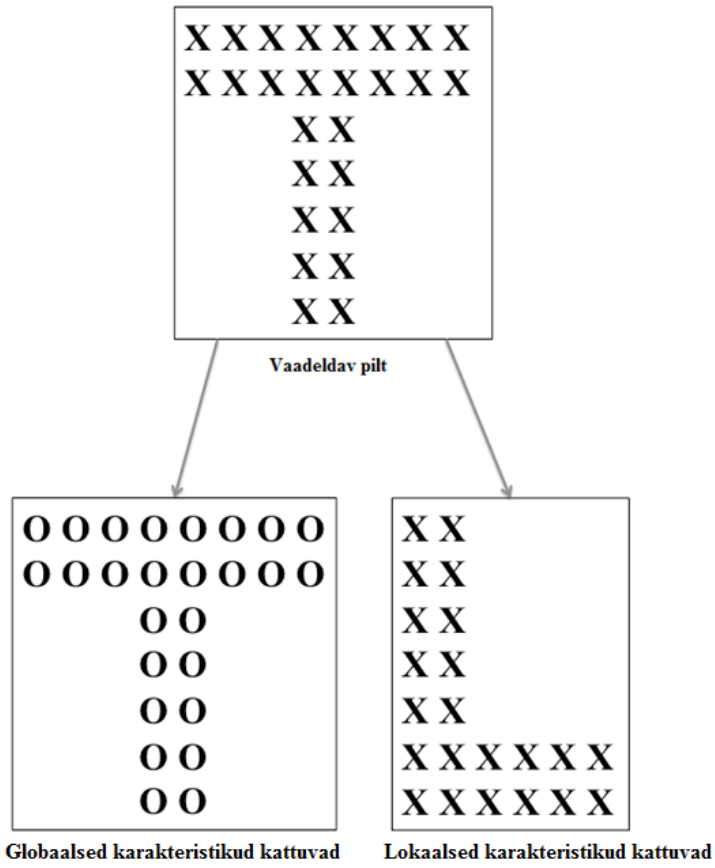
Selleks, et tuvastada piltidel olevate väikeste detailide kokkulangevus, tuleb leida ja võrrelda piltide lokaalseid karakteristikuid. Et tuvastada väikestest detailidest moodustunud kujundite kokkulangevus, tuleb leida ja võrrelda piltide globaalseid karakteristikuid (Vt. ptk 3.3.1).

### 3.3.1 Piltide karakteristikud

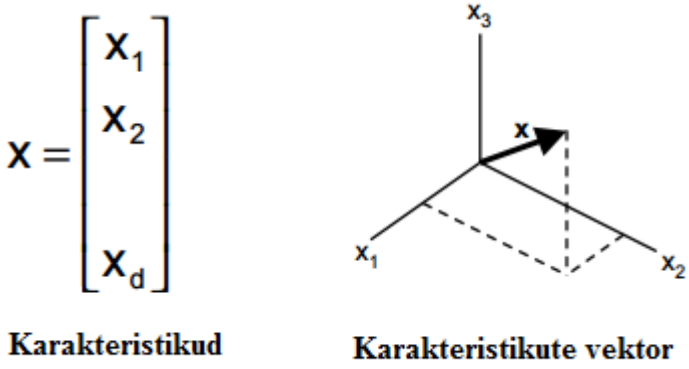
Pildid koosnevad pikslitest, mis on sisuliselt värvitoone kujutavad numbrid. Kui pildi otsingut tehes vaadata igakord läbi kõikide piltide kõik pikslid, siis oleks see liiga aega nõudev ja kulukas protsess. Seega tehakse pikslitega erinevaid arvutusi ning saadakse uued numbrilised väärtused, karakteristikute vektorid, mis kirjeldavad pildil olevaid objekte. Sõnale karakteristik ei ole olemas ühtset definitsiooni, kuid üldiselt nimetatakse pildi tuvastuses karakteristikuks pildi mistahes huvi pakkuvat kohta. Karakteristikud võivad olla näiteks [2]:

- pildil olevad punktid;
- piirjooned ehk servad;
- nurgad;
- laikude tuvastus;

Pildid sisaldavad nii lokaalseid kui ka globaalseid karakteristikuid. Lokaalseteks karakteristikuteks nimetatakse pildil olevaid detaile ja pildi osasid, globaalsed karakteristikud kirjeldavad kogu pilti tervikuna (Joonis 28) [3]. Pildi karakterisikuid kirjeldab n-mõõtmeline karakteristikute vektor, mis koosneb n arvust pildi karakteristikust (Joonis 29) [4]. Karakteristikud on hästi valitud, kui nende abil on võimalik eristada erinevatesse klassidesse kuuluvaid objekte (Joonis 30).




Joonis 28. Näide lokaalsetest ja globaalsetest karakteristikutest.




Joonis 29. Näide: Karakteristikud ja karakteristikute vektor.


Joonis 30. Näide. Sarnastel klassidel on sarnased karakteristikud (a), erinevatel klassidel on sarnased karakteristikud (b).

### 3.3.2 Lokaalsete karakteristikute leidmine ja võrdlemine

Lokaalsete karakteristikute leidmiseks on erinevaid võimalusi, kuid käesolevas lõputöös tuuakse ühe võimaliku näitena välja SURF algoritm *(Speeded Up Robust Features).* SURF algoritm leiab ja kirjeldab pildil olevaid lokaalseid karakteristikuid. Karakteristikute leidmiseks kasutatakse püramiidi metoodikat, mille korral pildi resolutsiooni muudetakse astmeliselt (Joonis 31). See metoodika tagab, et tuvastatud karakteristikud on skaalas muutumatud. SURF algoritm kasutab arvutuste tegemise kiiruse tõstmiseks pildi integraalkuju (Vt. ptk. 3.3.2.1). SURF algoritm koosneb kolmest osast [20]:

- **karakteristikute punktide leidmine** – ruudukujulisi filtreid kasutades rakendatakse pildile Gaussiani filtrit[1]. Järgnevalt leitakse pildilt Hessiani maatriksi[2] abil igale punktile (X,Y) miinimumid ja maksimumid ehk tuvastatakse pildil olevad laigud. Karakteristud leitakse erinevatelt skaaladelt ning selleks, et leitud karakteristik oleksid ka pöörlemisele muutumatud, leitakse igale punktile orientatsioon;

- **karakteristikute vektorite leidmine** – karakteristikute vektor kirjeldab karakteristikute naaberpikslite intensiivsuse jaotust ning iga leitud punkti vektor on 64 dimensiooniline.

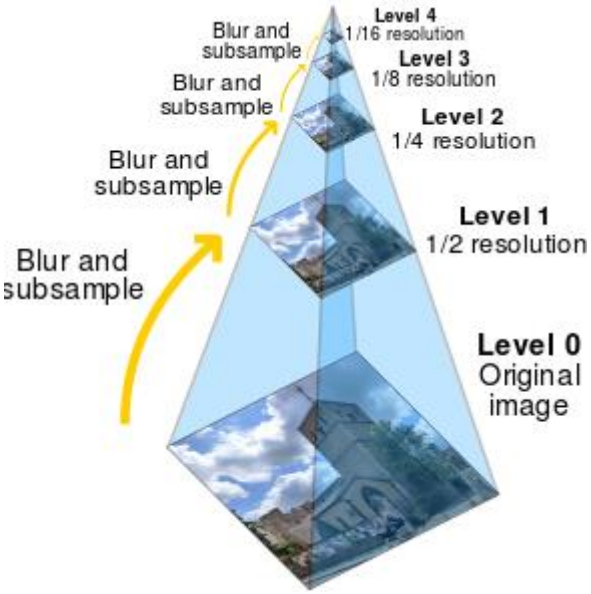
- **karakteristikute sobitamine –** karakteristikute sobitamine on erinevate piltide karakteristikute vektorite võrdlemine. Selleks on erinevaid metoodikaid, kuid antud töös realiseeriti karakteristikute vektorite väärtuste vahelise Euclidean'i

[1] http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

[2] https://www.math.vt.edu/people/dlr/m2k_svb11_hesian.pdf

kauguse leidmine, mida kirjeldab valem (11), kus p ja q on karakteristikute vektorid, ning n ja m on vastavate vektorite pikkused [21];

$$d(p,q) = d(q,p) = \sqrt{(q1-p1)^2\ (q2-p2)^2\ +\cdots+(qn-pn)^2} \qquad (11)$$



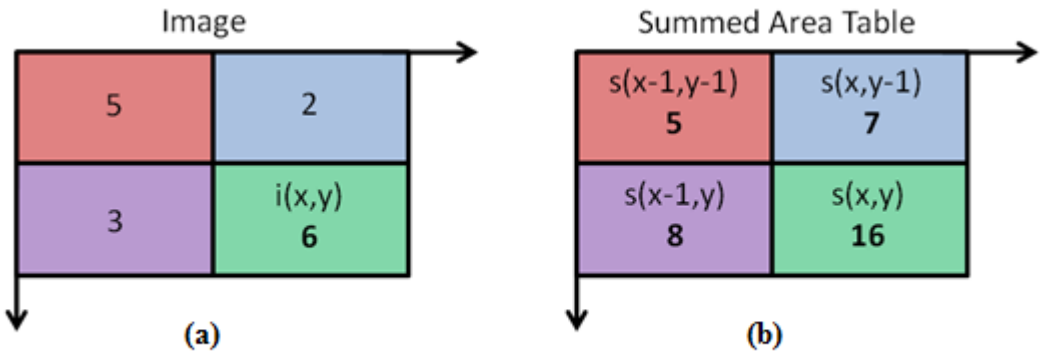

Joonis 31. Püramiidi metoodika lokaalsete karakteristikute leidmiseks.

#### 3.3.2.1 Integraalpilt

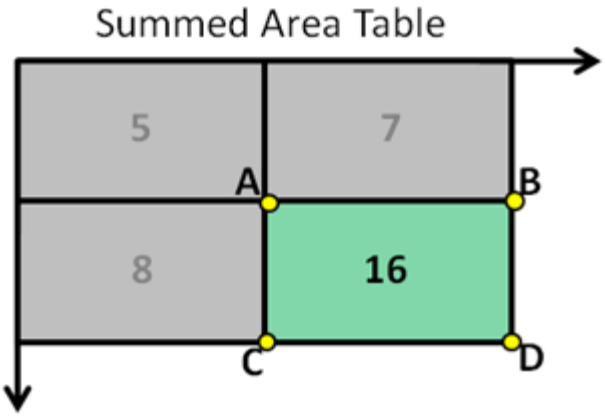
Integraalpildi abil on võimalik arvutada kiirelt ja efektiivselt pildi keskmist intensiivsust ning leida kogu pildi või pildi alamhulga pikslite väärtuste summat. Enne pildi integraalkujule viimist teisendatakse pilt hallideks toonideks (Vt. ptk 3.3.1.1.1).

Et luua integraal pilti, tuleb kõigepealt koostada summeeritud alade tabel [19]. Selle tabeli koostamine nõuab vähe aega, kuna selleks tuleb pildi pikslid ainult ühe korra läbi vaadata. Summeeritud alade tabelis arvutatakse väärtused valemi (12) järgi. Joonise 32 näitel on S(x,y) väärtus summeeritud alade tabelis 16, kuna S(x,y)=6+8+7-5=16.

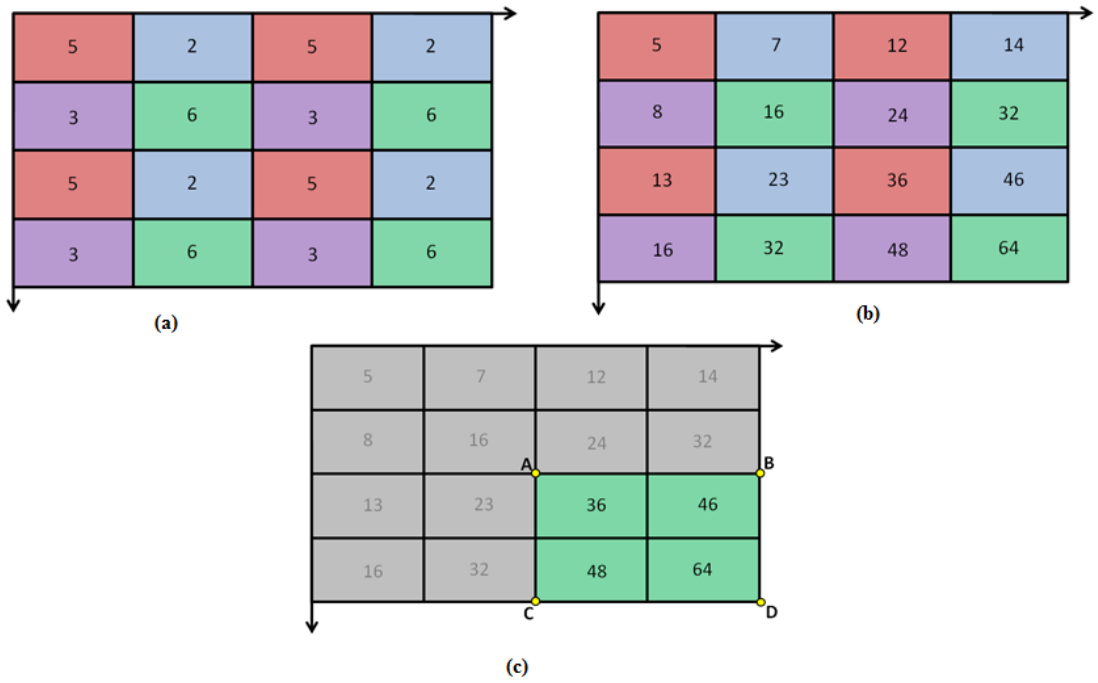
$$s(x,y) = i(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1) \qquad (12)$$




Joonis 32. Näide. Neljast pikslist koosnev pilt (a) ning selle summeeritud alade tabel (b).




Joonis 33. Näide. Summeritud alade tabel, kus iga tabeli väärtus näitab eelnevate väärtust summat.




Joonis 34. Näide. 16 pikslist koosnev pilt (a) ning selle summeeritud alade tabel (b) ning rohelisega kujutatud otsitav ala (c).

Kui summeritud alade tabel on leitud, siis on lihtne leida valemi (13) abil pildi alamhulkade pikslite väärtuste summasid. Näiteks joonise 33 näitel, kui soovitakse leida rohelise ala väärtust siis i($x'$, $y'$)=5+16-8-7=6. Joonisel 34 on samuti näide sellest, kuidas leida soovitud alamhulga väärtuste summat. Lähtudes valemist (13) i($x'$, $y'$)=16+64-32-32=16.

$$i(x',y') = s(A) + s(D) - s(B) - s(C) \quad (13)$$

### 3.3.3 Globaalsete karakteristikute leidmine ja võrdlemine

Järgnevalt kirjeldatakse Zernike momentide leidmist ja võrdlemist kui ühte võimalikku metoodikat globaalsete karakteristikute võrdlemiseks. Pildi momendiks laiemas mõistes nimetatakse teatud funktsiooni järgi hoolikalt kaalutud pikslite intensiivsuse keskmist väärtust [22] või pilti iseloomustavat numbrilist väärtust, mis on saadud teatud valemit kasutades. Pildi momenti võib vaadata kui ühte võimalikku karakteristikut pildi iseloomustamiseks. Enne Zernike karakteristikute leidmist viiakse kõik pildid N x N mõõtu. Seejärel leitakse Zernike karakteristikute vektor, mis võib olla näiteks 36 dimensiooniline (Tabel 2).

Zernike nm-ndat momenti arvutatakse valemite (14), (15) ja (16), (17) ja (18) järgi, kus n on järjekorra number ning m on korduste number (Tabel 2). Arvutuste tegemiseks vaadatakse pilti f(c,r) ühikringina f(ρ,Θ) (Joonis 35). $\lambda_N$ on pikslite arv uuritavas kujundis, j=$\sqrt{-1}$ ning ρ ja Θ arvutamist kirjeldavad vastavalt valemid (17) ja (18) [23].

Vektorite võrdlemiseks on erinevaid metoodikaid, kuid ka Zernike momentidest moodustatud n-dimensioonilisi vektoreid saab võrrelda Euclidean'i kaugust arvutades (Valem 11).

$$R_{n,m} = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s!(((n+|m|)/2)-s)!(((n-|m|)/2)-s)!} \rho^{n-2s} \tag{14}$$

$$V_{n,m}(\rho,\theta) = R_{n,m}(\rho) e^{jm\theta}, \quad |\rho| \leq 1 \tag{15}$$

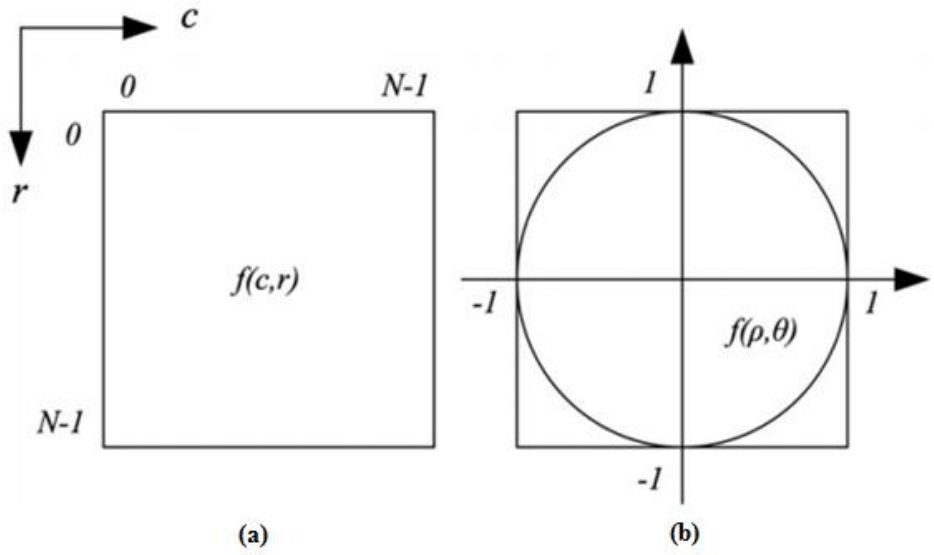
$$Z_{n,m} = \frac{n+1}{\lambda_N} \sum_{c=0}^{N-1} \sum_{r=0}^{N-1} f(c,r) V^*_{n,m}(c,r) \tag{16}$$

$$\rho_{cr} = \frac{\sqrt{(2c-N+1)^2+(2r-N+1)^2}}{N} \tag{17}$$

$$\theta_{cr} = \tan^{-1}\left(\frac{N-1-2r}{2c-N+1}\right) \tag{18}$$

Tabel 2. Zernike polünoomid erinevate järjekorra ja korduste numbrite korral.

| p \ q | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | $R_{0,0}$ | | | | | | | | | | |
| 1 | | $R_{1,1}$ | | | | | | | | | |
| 2 | $R_{2,0}$ | | $R_{2,2}$ | | | | | | | | |
| 3 | | $R_{3,1}$ | | $R_{3,3}$ | | | | | | | |
| 4 | $R_{4,0}$ | | $R_{4,2}$ | | $R_{4,4}$ | | | | | | |
| 5 | | $R_{5,1}$ | | $R_{5,3}$ | | $R_{5,5}$ | | | | | |
| 6 | $R_{6,0}$ | | $R_{6,2}$ | | $R_{6,4}$ | | $R_{6,6}$ | | | | |
| 7 | | $R_{7,1}$ | | $R_{7,3}$ | | $R_{7,5}$ | | $R_{7,7}$ | | | |
| 8 | $R_{8,0}$ | | $R_{8,2}$ | | $R_{8,4}$ | | $R_{8,6}$ | | $R_{8,8}$ | | |
| 9 | | $R_{9,1}$ | | $R_{9,3}$ | | $R_{9,5}$ | | $R_{9,7}$ | | $R_{9,9}$ | |
| 10 | $R_{10,0}$ | | $R_{10,2}$ | | $R_{10,4}$ | | $R_{10,6}$ | | $R_{10,8}$ | | $R_{10,10}$ |




Joonis 35. Näide. (a) on N x N pilt f(c,r) ning (b) on sama pilt kaardistatud ühikringina f(ρ,Θ).

# 4 Kaubamärkide reproduktsioonide võrdlemise näidisprogramm kasutades MATLAB' i

Eelnevalt kirjeldati pildipõhise otsingu teoreetilisi aluseid, kuid selles peatükis antakse ülevaade programmis MATLAB tehtud pildipõhise kaubamärgi reproduktsiooni otsingu näidisprogrammist. Peatükid 4.1-4.5 kirjeldavad erinevaid otsinguid ning nendega teostatud teste. Värvide tuvastamise ning mustrite sobitamisel põhineva teksti tuvastuse korral esitati testi tulemused ka ROC jaotusena. ROC jaotus iseloomustab graafiliselt binaarklassifitseerimissüsteemi suutlikkust ning selle kõver moodustub tõeliste õigete ja -valede suhte arvudest TPR ja FPR [25]. TPR arvutamiseks kasutatakse valemit (19), kus TP on õigesti tuvastatud piltide arv ning FN on tuvastamata jäetud piltide arv. FPR leidmiseks kasutatakse valemit (20), kus FP on tuvastatud piltide arv, mida poleks tegelikult pidanud tuvastama ning TN on tuvastamata jäetud piltide arv, mida ei pidanudki tuvastama [26].

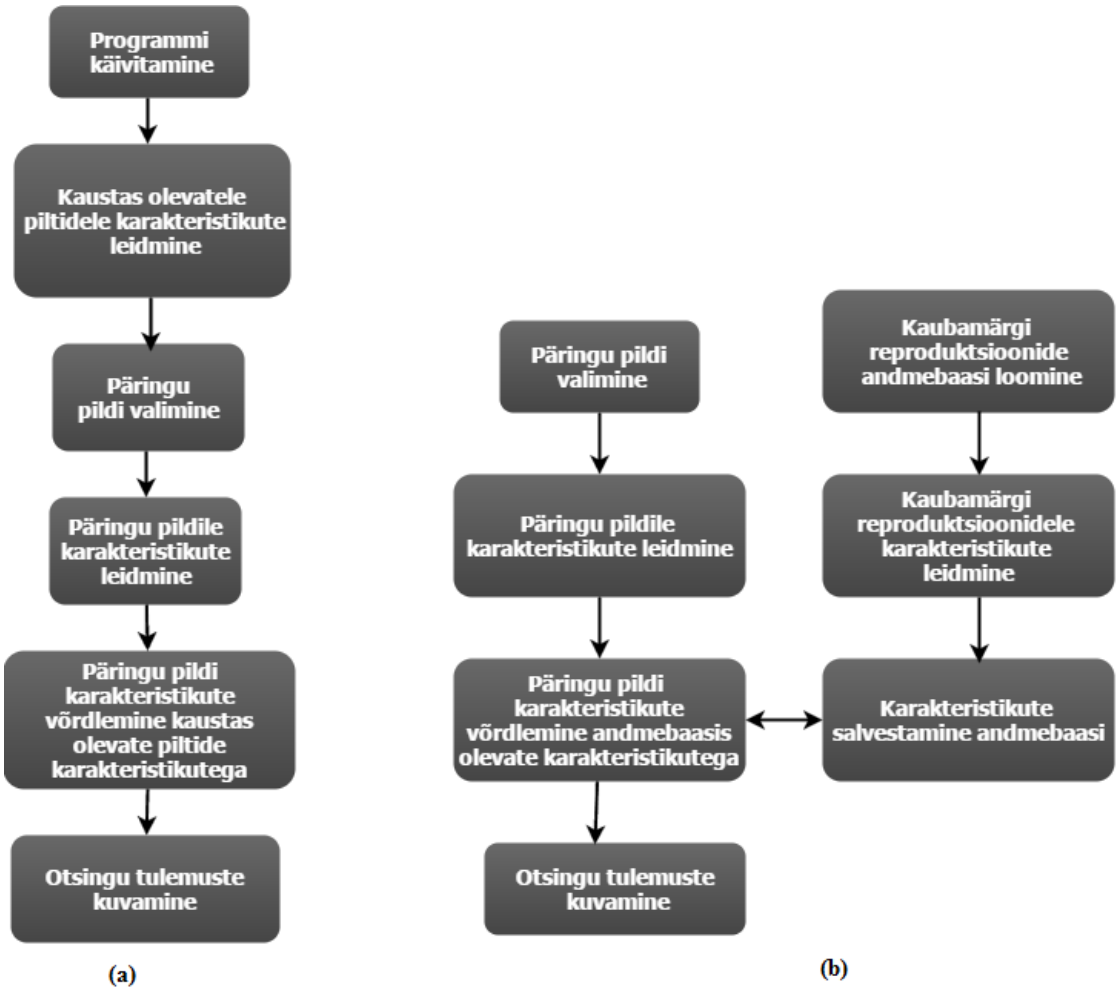
$$TPR = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{19}$$

$$FPR = \frac{\mathrm{FP}}{\mathrm{FP+TN}} \tag{20}$$

Loodud näidisprogramm koosneb viiest eraldi käivitatavast alamprogrammist:

- „runColor.m" – käivitatakse näidisotsing, mis sisaldab keskmiste värvitoonide leidmist ja võrdlemist (Vt. ptk. 4.1);
- „runOCR.m" – käivitatakse näidisotsing, mis sisaldab mustrite sobitamise metoodikal põhinevat teksti tuvastust (Vt. ptk. 4.2);
- „runKNN.m" – käivitatakse näidisotsing, mis sisaldab masinõppel põhinevat tähemärkide tuvastust (Vt. 4.3);
- „runLocal.m" –käivitatakse näidisotsing, mis sisaldab lokaalsete karakteristikute leidmist ja sobitamist (Vt. ptk. 4.4);
- „runGlobal.m" – käivitatakse näidisotsing, mis sisaldab globaalsete karakteristikute leidmist ja võrdlemist (Vt. ptk 4.5);

Joonisel 36 on kaks programmi voogu. Esimene neist on näidisprogrammis teostatud voog, milles andmed salvestatakse massiividesse. Kindlas kaustas olevatele piltidele leitakse karakteristikud ning salvestatakse need massiivi. Seejärel valib kasutaja päringu aluseks oleva pildi, millele leitakse samuti karakteristikud. Lõpptulemus saadakse päringu pildi karakteristikute võrdlemisel massiivi salvestatud karakteristikutega. Selliselt toimivad kõik viis alamprogrammi. Andmete salvestamine massiivi on teostatud näidisprogrammis metoodikate võrdlemiseks ja demonstreerimiseks. Tegelikkuses on soovitav luua programm, mis toimib joonisel 36 näidatud teise voo (b) järgi. Võrdlemise aluseks olevate karakteristikute leidmine ning otsingu teostamine peaksid olema ühise karakteristikute andmebaasiga eraldiseisvad programmid.




Joonis 36. Näidisotsingu voog (a) ning soovitatav voog (b).

## 4.1 Värvitoonide võrdlemine

Värvitoonide võrdlemiseks on loodud kindlatest värvidest koosnev värvikaart. Kõikide piltide jaoks rakendatakse funktsiooni, mis asendab pikslite väärtused lähima väärtusega uues värvikaardis. Järgnevalt loetakse erinevaid toone sisaldavad pikslid kokku ning võrreldakse neid päringu aluseks oleva pildi värvitoonide sisaldusega (Vt. ptk. 3.1).

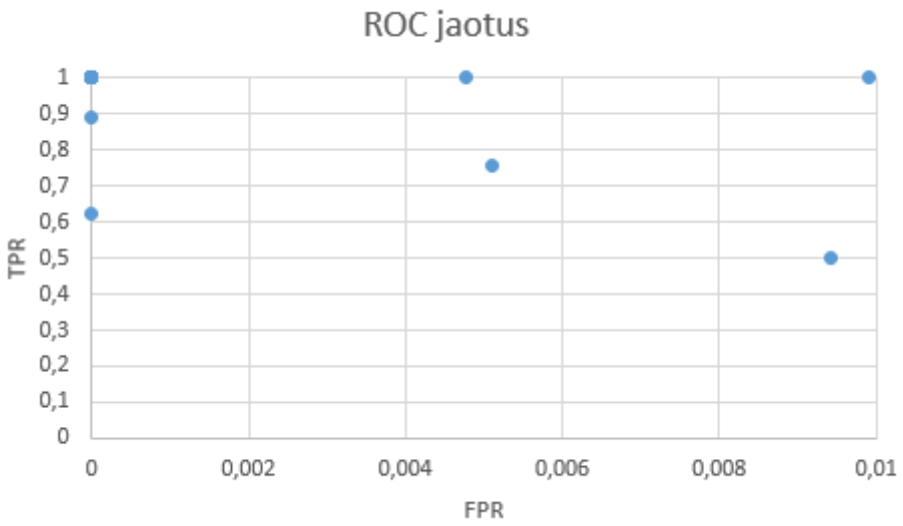
Antud programmi testimiseks käivitab kasutaja programmi ning valib päringu aluseks oleva pildi. Seejärel kuvatakse pildid, mille põhitoonid on sarnased päringu aluseks oleva pildi põhitoonidega. Päringu aluseks olevat pilti võrreldakse kaustas *„color“* olevate piltidega.

### 4.1.1 Näidisprogrammiga teostatud testid

Värvitoonide võrdlemise efektiivsuse tuvastamiseks teostati testid 80 internetist leitud kaubamärgi reproduktsiooniga, mis on jagatud 16 klassi ning igas klassis on 5 lõputöö autori arvates eksitavalt sarnast kaubamärki. Kaubamärgi reproduktsioonide jagamine kategooriatesse erinevate värvide järgi on kirjeldatud tabelis 4. Nii testandmete valimine kui ka värvide kategoriseerimine on lõputöö autori oma looming ning ei pruugi olla juriidiliselt korrektne. Tabelis 3 on veerg „Testide arv“ piltide arv, milles sisalduvad vastavad põhitoonid ning veerg „Piltide arv“ arvutatakse valemi (21) abil. Sellist valemit kasutatakse põhjusel, et teste korratakse seni, kuni iga pilt on olnud päringu aluseks olev pilt ning vastusena kuvatakse kõik õiged pildid, kaasa arvatud päringu aluseks olev pilt.

$$Piltide\ arv = (Testide\ arv) * (Testide\ arv) \tag{21}$$

Värvide tuvastamise korral oli probleemiks põhivärvi valimine olukorras, kus värv on kahe põhivärvi vaheline toon – näiteks rohekas sinine. Samuti tektitasid valesid tulemusi pildid, mis sisaldasid rohkem värve, kui peale vaadates tundub. Näiteks pilt, mis tundub punane ja sinine, sisaldab lähemalt vaadates kollaseid kontuure, mis pildi üldmuljet ei muuda.




Joonis 37. Värvitoonide võrdlemise testi tulemuste ROC jaotus.

Tabel 3. Näidisprogrammiga teostatud testid värvitoonide võrdlemisele.

| Põhitoonid | Testide arv | Piltide arv | Õigesti tuvastatud pildid | Pildid, mida poleks pidanud tuvastama | Pildid, mida ei tuvastatud | TPR | FPR |
|---|---|---|---|---|---|---|---|
| must | 12 | 144 | 144 | 0 | 0 | 1 | 0 |
| sinine | 6 | 36 | 36 | 6 | 0 | 1 | 0,009917355 |
| kollane, must,punane | 4 | 16 | 16 | 0 | 0 | 1 | 0 |
| must,punane | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| roheline,kollane,punane,lilla,sinine | 3 | 9 | 9 | 0 | 0 | 1 | 0 |
| roheline,punane,kollane | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| kollane | 4 | 16 | 16 | 0 | 0 | 1 | 0 |
| roheline | 2 | 4 | 2 | 6 | 2 | 0,5 | 0,009419152 |
| sinine,kollane | 4 | 16 | 16 | 0 | 0 | 1 | 0 |
| sinine,punane,kollane,roheline,must | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| punane,sinine | 17 | 289 | 257 | 0 | 32 | 0,889273 | 0 |
| punane | 4 | 16 | 16 | 0 | 0 | 1 | 0 |
| punane, kollane | 3 | 9 | 9 | 3 | 0 | 1 | 0,004769475 |
| kollane,punane,sinine | 7 | 49 | 37 | 3 | 12 | 0,755102 | 0,005093379 |
| roheline,must | 3 | 9 | 9 | 0 | 0 | 1 | 0 |
| lilla | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| roheline,must,punane | 4 | 16 | 10 | 0 | 6 | 0,625 | 0 |
| roheline,punane | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| lilla,must | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| **KOKKU** | 79 | 635 | 583 | 18 | 52 | 0,91811 | 0,001572327 |
| | | | **92%** | 3% | 8% | **92%** | **0.2%** |
| **LÕPPTULEMUS** | 0.5(TPR)+(1-0.5(FPR)=**95%** | | | | | | |

Tabel 4. Testandmete jagamine kategooriatesse värvide järgi.

| Kaubamärk | Värvid |
|---|---|
| Adidas | 3 x must; 2 x sinine |
| Agip | 3 x kollane,must,punane; 1 x must; 1 x must,punane |
| Apple | 2 x roheline,kollane,punane,lilla,sinine; 1 x must; 1 x ilma värvita, 1 x; roheline,kollane,punane |
| Bing | 2 x sinine,kollane; 1 x kollane; 1 x roheline; 1 x sinine,punane,kollane,roheline,must |
| Cinzano | 4 x punane,sinine; 1 x must |
| Coca-Cola | 3 x punane; 2 x must |
| Heineken | 4 x roheline,must,punane; 1 x roheline,punane |
| Hesburger | 5 x punane,sinine |
| Mc Donalds | 1 x kollane; 1 x must; 2 x punane,kollane; 1 x kollane, punane,must |
| Nike | 1 x punane; 2 x must-valge; 1 x roheline,kollane,punane,lilla,sinine; 1 x kollane,punane |
| Pepsi | 5 x punane,sinine |
| Rimi | 3x punane,sinine; 2 x kollane,punane,sinine |
| Selver | 5 x punane,sinine,kollane |
| Starbucks | 1 x roheline; 3 x roheline,must; 1 x lilla |
| Statoil | 2 x kollane; 2 x kollane,sinine; 1 x lilla,must |
| Twitter | 4 x sinine; 1 x must |

## 4.2 Mustrite sobitamisel põhinev teksti tuvastamine

Selleks, et leida pildil olevat teksti, teisendatakse kõigepealt pilt binaarkujule (Vt. ptk. 3.2.1.1). Selleks, et binaarkujul olevast pildist olulist informatsiooni välja lugeda, tuleb enne pilti õigesti töödelda. Antud näidisprogrammis teostatakse kontroll, et kui pildi esimesel real on üle 90% pikslitest mustad, siis järeldatakse, et pildil on tume taust ning binaarkujul olev pilt muudetakse vastupidiseks (1-d lähevad 0-ks ning vastupidi), eesmärk on eralda informatsiooni sisaldav osa taustast. Teksti tuvastamise protsess on kirjeldatud peatükis 3.2.1.2. Leitakse pildilt read, seejärel märgistatakse realt leitud karakteristikud ning võrreldakse nende maatrikseid ette antud tähemärkide maatriksitega. Kuna mustrite sobitamisel põhinev teksti tuvastus ei sisalda üldjuhul üksühele õiget tulemust, siis teostatakse ka saadud tulemusele järeltöötlus, mis on kirjeldatud peatükis 3.2.1.4.

Antud programmi testimiseks käivitab kasutaja programmi ning valib päringu aluseks oleva pildi. Seejärel kuvatakse tulemused sarnasuse kahanevas järjekorras. Päringu aluseks olevat pilti võrreldakse kaustas *„OCR“* olevate piltidega.

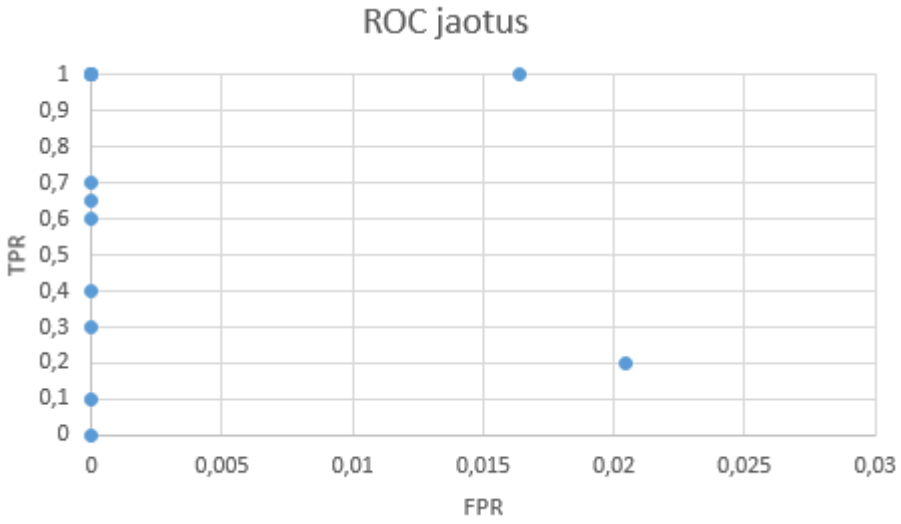
### 4.2.1 Näidisprogrammiga teostatud testid

Et testida mustrite sobitamisel põhinevat teksti tuvastust, siis kasutas töö autor 65 erinevat kaubamärgi reproduktsiooni tekstilist osa. Kuna mustripõhise tekstituvastusega saab tuvastada ainult selliseid tähe- ja numbrimärke, mis on kaustas „letters_numbers“, siis testiti antud programmi toimimist lihtsamate kirjastiilidega. Samuti jäeti välja

kaubamärgi reproduktsioonid, mis sisaldasid pööratud tähe- või numbrimärke. Tabelis 5 on veerg „Testide arv“ piltide arv, mis sisaldavad sarnast teksti ning veerg „Piltide arv“ arvutatakse valemi (22) abil. Sellist valemit kasutatakse põhjusel, et teste korratakse seni, kuni iga pilt on olnud päringu aluseks olev pilt ning päringu aluseks olevat pilti lõpptulemuse arvutamisel ei arvestata, kuna see on alati iseendaga 100% sarnane.

$$Piltide\ arv = (Testide\ arv - 1) * (Testide\ arv) \quad (22)$$

Mustrite sobitamisel põhineva tekstituvastuse korral olid põhiliseks probleemiks väheste kirjastiilide tundmine ning suutmatus tuvastada teksti keerulisemate kujundite seest.




Joonis 38. Mustrite sobitamisel põhineva teksti tuvastamise testi tulemuse ROC jaotus.

Tabel 5. Näidisprogrammiga teostatud testid maatriksite sobitamisel põhineva tekstituvastuse efektiivsuse hindamiseks.

| Tähemärgid | Testide arv | Piltide arv | Õigesti tuvastatud pildid | Pildid, mida poleks pidanud tuvastama | Pildid, mida ei tuvastatud | TPR | FPR |
|---|---|---|---|---|---|---|---|
| Adidas | 5 | 20 | 20 | 0 | 0 | 1 | 0 |
| Bing | 5 | 20 | 8 | 0 | 12 | 0,4 | 0 |
| Stockmann | 5 | 20 | 4 | 5 | 16 | 0,2 | 0,020408 |
| Ferrari | 5 | 20 | 20 | 0 | 0 | 1 | 0 |
| Heineken | 5 | 20 | 0 | 0 | 20 | 0 | 0 |
| Hesburger | 5 | 20 | 20 | 0 | 0 | 1 | 0 |
| Kickup | 5 | 20 | 13 | 0 | 7 | 0,65 | 0 |
| Mc Donalds | 5 | 20 | 12 | 0 | 8 | 0,6 | 0 |
| Nike Vision | 5 | 20 | 20 | 4 | 0 | 1 | 0,016393 |
| Pepsi | 5 | 20 | 20 | 0 | 0 | 1 | 0 |
| Rimi | 5 | 20 | 2 | 0 | 18 | 0,1 | 0 |
| Selver | 5 | 20 | 14 | 0 | 6 | 0,7 | 0 |
| Statoil | 5 | 20 | 6 | 0 | 14 | 0,3 | 0 |
| **KOKKU** | 65 | 260 | 159 | 9 | 101 | 0,611538 | 0,002876 |
| | | | **61%** | 3% | 39% | **61%** | **0.3%** |
| **LÕPPTULEMUS** | 0.5(TPR)+0.5(1-FPR)=**79%** | | | | | | |

## 4.3 Masinõppel põhinev teksti tuvastamine

Masinõppel põhinev tekst tuvastamine on realiseeritud kasutades K-NN algoritmi (Vt. ptk. 3.2.2.1). Kõigepealt õpetatakse programmile selgeks kindel hulk tähemärke, mis asuvad kaustas *„KNN/learning“*. Selleks leitakse kõikidele nimetatud kaustas olevatele piltidele karakteristikud. Karakteristikute leidmise loogika on kirjeldatud peatükis 3.2.2.2 Samuti antakse programmile ette info, millisesse klassi õpetatavad tähemärgid kuuluvad. Järgnevalt luuakse klassifitseerimise mudel, mis sisaldab õpitud tähemärkide karakteristikuid ning nende väärtuseid. Kasutaja valib programmi testimiseks tähemärgi pildi, annab ette k väärtuse ning programm kuvab vastusena, millise tähemärgiga see kõige sarnasem on. Karakteristikute läheduste hindamiseks kasutatakse antud programmis Euclidean’ i kauguse leidmist (Valem 11).

Antud töös ei ole teostatud piltide eeltöötlemist selliselt, et programm oskaks vahet teha tekstil ja graafilisel kujutisel, seega tuleb tähemärkide tuvastamist testida piltidega, kus on vaid tähemärk ning hele taust.

### 4.3.1 Näidisprogrammiga teostatud testid

K-NN algoritmi tähemärkide tuvastamise efektiivsuse tuvastamiseks teostati teste 100 erineva tähemärgiga ning iga tähemärgi korral kolme erineva k väärtusega. Tähemärgid, millega testiti, asuvad kaustas „KNN“. Testi tulemus on alati binaarne ehk õige või vale.

Masinõppel põhineva tekstituvastuse korral olid probleemiks sarnase kujuga tähemärgid. Lõputöö autor on arvamusel, et, see on tingitud sellest,et programmile õpetati selgeks liiga väike hulk ühte liiki tähemärke.

Tabel 6. K-NN' efektiivsuse testimine tähemärkide tuvastamisel.

| Tähemärgid | Testide arv | K väärtus | Õigesti tuvastatud tähemärgid | Valesti tuvastatud tähemärgid |
|---|---|---|---|---|
| A | 10 | 1 | 9 | 1 |
| A | 10 | 5 | 9 | 1 |
| A | 10 | 10 | 9 | 1 |
| B | 10 | 1 | 9 | 1 |
| B | 10 | 5 | 9 | 1 |
| B | 10 | 10 | 9 | 1 |
| C | 10 | 1 | 10 | 0 |
| C | 10 | 5 | 10 | 0 |
| C | 10 | 10 | 10 | 0 |
| D | 10 | 1 | 9 | 1 |
| D | 10 | 5 | 9 | 1 |
| D | 10 | 10 | 9 | 1 |
| E | 10 | 1 | 8 | 2 |
| E | 10 | 5 | 8 | 2 |
| E | 10 | 10 | 8 | 2 |
| F | 10 | 1 | 10 | 0 |
| F | 10 | 5 | 10 | 0 |
| F | 10 | 10 | 10 | 0 |
| G | 10 | 1 | 10 | 0 |
| G | 10 | 5 | 10 | 0 |
| G | 10 | 10 | 10 | 0 |
| H | 10 | 1 | 10 | 0 |
| H | 10 | 5 | 10 | 0 |
| H | 10 | 10 | 10 | 0 |
| K | 10 | 1 | 9 | 1 |
| K | 10 | 5 | 9 | 1 |
| K | 10 | 10 | 9 | 1 |
| S | 10 | 1 | 8 | 2 |
| S | 10 | 5 | 8 | 2 |
| S | 10 | 10 | 8 | 2 |
| **KOKKU** | 300 | | 276 | 24 |
| **%** | | | **92%** | **8%** |

## 4.4 Lokaalsete karakteristikute leidmine ja võrdlemine

Lokaalsete karakteristikute leidmiseks kasutatakse antud programmis algoritmi SURF (Vt. ptk. 3.3.2). Kõigepealt muudetakse pilt hallideks toonideks (Vt. ptk. 3.2.1.1.1), seejärel leitakse pildilt lokaalsed punktid ja nende 64-dimensioonilised vektorid ning salvestatakse need massiivi . Lõpptulemuse saamiseks leitakse SURF punktid ja vektorid päringu aluseks olevale pildile ning võrreldakse neid kaustas *„local“* olevate piltide vektoritega.

Antud programmi testimiseks käivitab kasutaja programmi ning valib päringu aluseks oleva pildi. Seejärel kuvatakse tulemused sarnasuse kahanevas järjekorras. Päringu aluseks olevat pilti võrreldakse kaustas *„local“* olevate piltidega.

### 4.4.1 Näidisprogrammiga teostatud testid

Lokaalsete karakteristikute leidmise efektiivsuse tuvastamiseks teostati testid 54 pildiga. 30 pilti on internetist leitud ning ülejäänud on lõputöö autori enda kombineeritud, et näidata SURF algoritmi võimet tuvastada erinevaid kujundeid, mis on moodustunud sarnastest lokaalsetest osadest. Testida põhjal saab väita, et antud algoritm on sobilik ainult väga sarnaste detailide otsimiseks. Antud programm järjestab tulemused karakteristikute Euclidean' i kauguste järgi ning ei ole ette antud läve, millest üleval pool on õiged vastused ning all pool valed. Seetõttu arvestatakse pildid õigesti tuvastatuks, kui need kuvati vastuses esimestena ning ROC jaotust ei leita. Tabelis 7 on veerg „Testide arv“ piltide arv, mis sisaldavad sarnaseid lokaalseid karakteristikuid ning veerg „Pilte kokku“ arvutatakse valemi (22) abil. Sellist valemit kasutatakse põhjusel, et teste korratakse seni, kuni iga pilt on olnud päringu aluseks olev pilt ning päringu aluseks olevat pilti lõpptulemuse arvutamisel ei arvestata, kuna see on alati iseendaga 100% sarnane.

Lokaalsete karakterisikute tuvastamise korral oli probleemiks see, et SURF algoritm oskab tuvastada vaid väga sarnaseid detaile. Lõputöö autor on arvamusel, et SURF metoodika on sobilikum tavaliselt pildilt kaubamärgi reproduktsiooni otsimiseks. (Joonisv 39).


Joonis 39. Kaubamärgi reproduktsiooni tuvastus tavaliselt pildilt kasutades SURF algoritmi.

Tabel 7. Näidisprogrammiga teostatud testid lokaalsete karakteristikute võrdlemiseks SURF' algoritmiga.

| Kaubamärk | Testide arv | Pilte kokku | Tuvastatud pildid | Pildid, mida ei tuvastatud |
|---|---|---|---|---|
| Adidas_1 | 2 | 2 | 2 | 0 |
| Adidas_2 | 2 | 2 | 2 | 0 |
| Adidas_3 | 2 | 2 | 2 | 0 |
| Agip | 5 | 20 | 20 | 0 |
| Apple | 5 | 20 | 20 | 0 |
| Bing | 5 | 20 | 20 | 0 |
| Cinzano | 4 | 12 | 11 | 1 |
| Coca-Cola | 5 | 20 | 17 | 3 |
| Heineken | 4 | 12 | 11 | 1 |
| Mc Donalds | 5 | 20 | 17 | 3 |
| Nike | 5 | 20 | 15 | 5 |
| Rimi | 5 | 20 | 8 | 12 |
| Starbucks | 5 | 20 | 16 | 4 |
| **KOKKU** | 54 | 190 | 161 | 29 |
| | | % | 84% | 16% |

## 4.5 Globaalsete karakteristikute leidmine ja võrdlemine

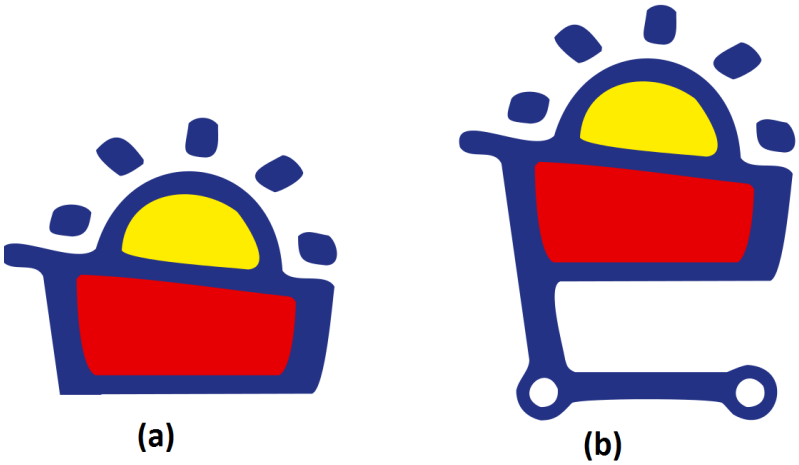
Antud programmis kasutatakse globaalsete karakteristikutena Zernike momente (Vt. ptk. 3.3.3). Zernike momentidest moodustatakse 36 dimensiooniline vektor ning erinevate piltide vektoreid võrreldakse sarnaselt nagu lokaalsete karakteristikute korral leides Euclidean'i kaugust (Valem 11).

Et testida antud programmi toimimist, käivitab kasutaja programmi, valib päringu aluseks oleva pildi ning talle kuvatakse vastavas järjekorras kõige sarnasemad tulemused kaustas *„global“* olevate piltidega. Kuna antud programm leiab ainult globaalseid karakteristikuid, siis nimetatud kaustas olevad pildid ei sisalda teksti osa. Iga pilt sisaldab ainult ühte kujutist ning taust on kõikidel piltidel valge (antud töös ei realiseeritud pildi eeltöötlust selliselt, et saaks globaalseid karakteristikuid leida mistahes pildile).

### 4.5.1 Näidisprogrammiga teostatud testid

Lokaalsete karakteristikute leidmise efektiivsuse tuvastamiseks teostati testid 64 pildiga. Mõned piltidest on leitud internetist ning mõned on töö autori joonistatud, et näidata Zernike momentide võimeikust, tuvastada sarnaseid kujundeid. Antud programm järjestab tulemused karakteristikute Euclidean' i kauguste järgi ning ei ole ette antud läve, millest üleval pool on õiged vastused ning all pool valed. Seetõttu arvestatakse pildid õigesti tuvastatuks, kui need kuvati vastuses esimestena ning ROC jaotust ei leita. Tabelis 8 on veerg „Testide arv“ piltide arv, mis sisaldavad sarnaseid globaalseid karakteristikuid ning veerg „Pilte kokku“ arvutatakse valemi (22) abil. Sellist valemit kasutatakse põhjusel, et teste korratakse seni, kuni iga pilt on olnud päringu aluseks olev pilt ning päringu aluseks olevat pilti lõpptulemuse arvutamisel ei arvestata, kuna see on alati iseendaga 100% sarnane.

Globaalsete karakteristikute tuvastamisel Zernike momentide abil oli probleemiks see, et antud metoodika peab sarnaseks vaid neid kujundeid, mille raskuskese on sarnases kohas (Joonis 40).




Joonis 40. Näide kujunditest, mida Zernike momentidel põhinev otsing sarnaseks ei pea.

Tabel 8. Näidisprogrammiga teostatud testid globaalsete karakteristikute võrdlemiseks Zernike momentide leidmise abil.

| Kaubamärk | Testide arv | Pilte kokku | Tuvastatud pildid | Pildid, mida ei tuvastatud | Tuvastus-protsent |
|---|---|---|---|---|---|
| Adidas 1 | 3 | 6 | 6 | 0 | 100% |
| Adidas 2 | 2 | 2 | 2 | 0 | |
| Apple | 5 | 20 | 20 | 0 | 100% |
| Bing | 5 | 20 | 20 | 0 | 100% |
| Hesburger_täht | 3 | 6 | 6 | 0 | 100% |
| Hesburger_H | 3 | 6 | 6 | 0 | |
| Mc Donalds | 4 | 12 | 12 | 0 | 100% |
| Nike | 6 | 30 | 30 | 0 | 100% |
| Pepsi | 6 | 30 | 26 | 4 | 87% |
| Rimi1 | 2 | 2 | 2 | 0 | 100% |
| Rimi2 | 2 | 2 | 2 | 0 | |
| Starbucks | 3 | 6 | 6 | 0 | 100% |
| Selver | 4 | 12 | 9 | 3 | 75% |
| Statoil 1 | 3 | 6 | 6 | 0 | 100% |
| Statoil 2 | 3 | 6 | 6 | 0 | |
| arrows | 4 | 12 | 10 | 2 | 83% |
| Twitter | 5 | 20 | 16 | 4 | 80% |
| KOKKU | | | 185 | 13 | 93% |

# 5 Edasiarendamise võimalused

Selleks, et arendada otsing, mida saab Eesti Patendiamet kasutada kaubamärkide reproduktsioonide äravahetamise tõenäosuse kontrollimiseks, tuleb põhjalikumalt uurida veel järgmiseid teemasid:

- Tuleb kirjeldada kaubamärgi seadusega [1] kooskõlas olevad reproduktsiooni otsingu ärireeglid, mis vastavad ka Eesti Patendiameti kaubamärgi ekspertiisi teostamise headele tavadele;

[1] https://www.riigiteataja.ee/akt/KaMS

- Vastavalt kirjeldatud ärireeglitele kasutada vajadusel veel mõningaid lokaalseid või globaalseid karakteristikuid leidvaid algoritme ning kombineerida neid omavahel;

- Vastavalt kirjeldatud ärireeglitele kohandada värvide põhist otsingut;

- Heade tulemuste saamiseks on vajalik arendada metoodika(d) tähe- ja numbrimärkide, graafiliste kujutiste ning tausta eristamiseks;

- Üks võimalus teksti tuvastamise tulemuste parandamiseks on arendada masinõppel põhinevat teksti tuvastust nii, et see oskab eristada ka erinevaid kirjastiile;

- Käesoleva töö autor peab vajalikuks kaubamäri reproduktsioonide tekstide sisuliseks võrdlemiseks sünonüümide andmebaasi loomist, kuna reproduktsioonidel kujutatud sõnade tähemärgid võivad küll erineda, kuid sõnad võivad sisuliselt kokku langeda;

- Vastavalt kirjeldatud ärireeglitele tuleb kokku leppida, kuidas kombineeritakse leitud tulemustest terviklik sarnasuse protsent;

Hetkel võrreldakse ühe pildi karakteristikuid teise pildi omadega kasutades selleks Euclidean' i kaugust (Valem 11). See tähendab, et sarnaste karakteristikute leidmiseks vaadatakse läbi kõik punktid. Kuna kaubamärkide reproduktsioonide andmebaas võib koosneda tohutul hulgal piltidest, siis ei ole see väga efektiivne lahendus. Näiteks masinõppel põhinev tähetuvastus võib sisaldada ühe tähemärgi kohta umbes 1000 näidispilti, mida omakorda kirjeldavad näiteks 64 dimensioonilised karakteristikute vektorid. See tähendab, et ühe tähemärgi võrdlemiseks ühe tähemärgiga tehakse juba 1000 x 64 operatsiooni. Seega on mõistlik leida karakteristikute võrdlemiseks kiirem metoodika. Töö autor toob välja mõned märksõnad, mida tasub uurida otsingu kiiruse efektiivsuse tõstmiseks – piltide hashimine, omavektorid, Baire' kaugus ja juhuslikud projektsioonid.

# 6 Kokkuvõte

Käesoleva lõputöö eesmärk oli välja selgitada, et kuidas on võimalik parendada Eesti Patendiameti kaubamärgi ekspertiisi tööprotsessi, kasutades reproduktsiooni äravahetamise tõenäosuse kontrollimiseks piltide analüüsimisel põhinevat otsingut.

Lõputöö käigus selgus, et ei ole olemas üheselt mõistetavaid ärireegleid, mille põhjal saaks öelda, kas kaubamärgi reproduktsioonid on omavahel äravahetamiseni sarnased. Seega peab antud töö autor äärmiselt oluliseks selliste ärireeglite kirjeldamist. Käesolevas töös uuriti kaubamärkide reproduktsioonide sarnasust kolme kriteeriumi järgi: tähe- ja numbrimärkide võrdlemine, värvide võrdlemine ning graafiliste kujutiste võrdlemine.

Näidisprogrammiga teostatud testide põhjal võib öelda, et tekstide võrdlemiseks on soovitatav eelistada mustrite sobitamisel põhinevale tekstituvastusele, mille tuvastusprotsent oli 79%, masinõppel põhinevat teksti tuvastust tuvastusprotsendiga 92%. Masinõppel põhinev tekstituvastus on kaubamärgi reproduktsioonidelt tähe- ning numbrimärkide lugemiseks eelistatum ka sellepärast, et suudab tuvastada keerulisemaid kirjastiile. Globaalsete karakteristikute leidmine Zernike momentide abil andis lõpptulemuse 93% ning lokaalsete karakteristikute leidmine SURF algoritmi kasutades tulemuse 84%. Lokaalsete karakteristikute võrdlemine, kasutades SURF algoritmi, tuvastab vaid eriti sarnaseid detaile, seetõttu tasub uurida veel teisi lokaalsete karakteristikute leidmise metoodikaid. Värvide võrdlemisel põhineva otsingu tuvastusprotsent oli 95%.

Lõputöö autor on arvamusel, et praegune ekspertide süsteem peaks muutuma reeglite põhiseks ekspertsüsteemiks. Selleks, et Eesti Patendiamet saaks kaubamärgi reproduktsioonide kontentanalüüsil põhineva otsingu kasutusule võtta, tuleb kirjeldada kaubamärgi reproduktsiooni äravahetamise tõenäosuse hindamise ärireeglid ning arendada otsingut nii, et see töötaks piisava kiirusega ka suurte andmehulkade korral.

# Kasutatud kirjandus


[1] www.epa.ee (08.05.2016)

[2] http://www.cs.toronto.edu/~jepson/csc420/ (08.05.2016)

[3] J.Davidoff, E. Fonteneau, J. Fagot „Local and Global Processing: Observations from a Remote Culture“ 2008, Elsevier B.V.

[4] http://research.cs.tamu.edu/prism/lectures/iss/iss_l9.pdf (08.05.2016)

[5] https://en.wikipedia.org/wiki/Optical_character_recognition (08.05.2016)

[6] F. Mohammad, J. Anarase, M. Shingote, P. Ghanawat „Optical Cahracter Recognition Implementation Using Pattern Matching“ 2014, ISB&M School of Technology

[7] http://whatis.techtarget.com/definition/grayscale (08.05.2016)

[8] http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/ (08.05.2016)

[9] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT2/node3.html (08.05.2016)

[10] http://www.rapidtables.com/convert/color/rgb-to-hsv.htm (08.05.2016)

[11] http://blogs.mathworks.com/steve/2008/04/14/relabeling-a-label-matrix/ (08.05.2016)

[12] http://se.mathworks.com/help/images/ref/corr2.html (08.05.2016)

[13] http://se.mathworks.com/help/vision/examples/automatically-detect-and-recognize-text-in-natural-images.html (08.05.2016)

[14] A. Coates, B.Carpetner, C.Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, A. Y. Ng „Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning“ Computer Science Department Stanford University

[15] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.html (08.05.2016)

[16] M. Rajalingam, P. Sumari, V. Raman „Text Detection and Extraction from Document Images using K-Nearest Neighbor Rule“ 2014, International Journal of Computer and Information Technolog

[17] W. Wang „Optical Character Recognition, Using K-Nearest Neighbors“ 2014

[18] http://homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm (08.05.2016)

[19] https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/ (08.05.2016)

[20] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool „Speeded-Up Robust Features (SURF)“ ETH Zurich, BIWI

[21] I. Dokmanic, R. Parhizkar, J. Ranieri, M. Vetterli „Euclidean Distance Matrices“ 2015

[22] M.-K. Hui „Visual Pattern Recognition by Moment Invariants“ 1962, Ire Transactions on Information Theory

[23] A. Tahmasbi, F. Saki, S. B. Shokouhi „Classification of Benign and Malignant Masses Based on Zernike Moments“, 2011, Comput Biol Med

[24] Otsu, N., „A Threshold Selection Method from Gray-Level Histograms“ IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.

[25] http://gim.unmc.edu/dxtests/roc2.htm (08.05.2016)

[26] https://www.med.emory.edu/EMAC/curriculum/diagnosis/sensand.htm (08.05.2016)

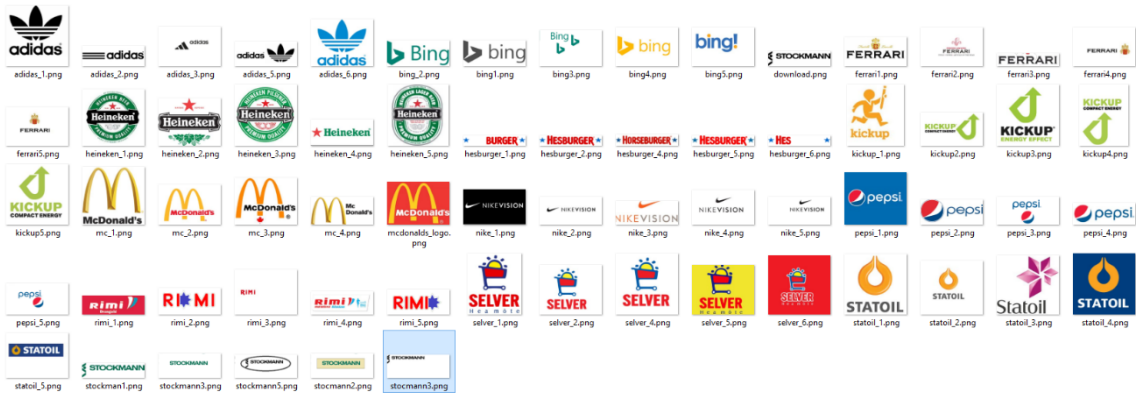
# Lisa 1 – Kasutatud näidisandmed

Kaust „global“

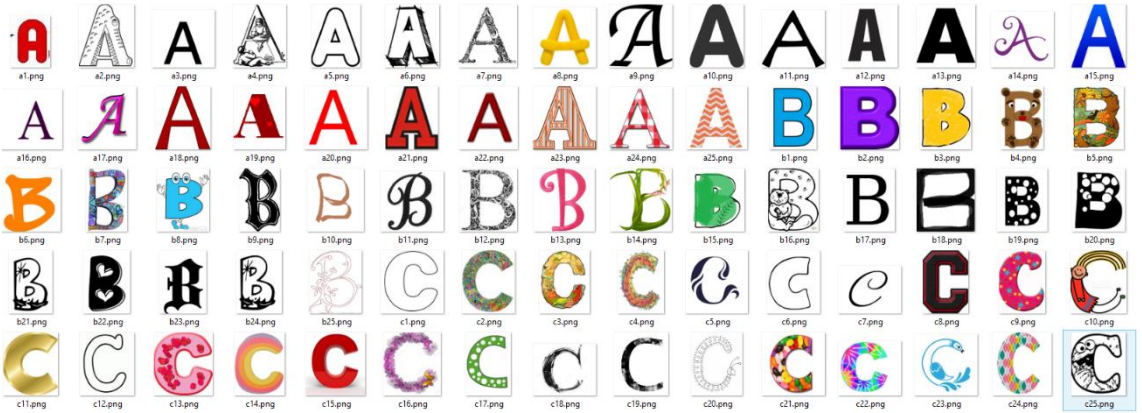
Kaust „color“

Kaust „local“

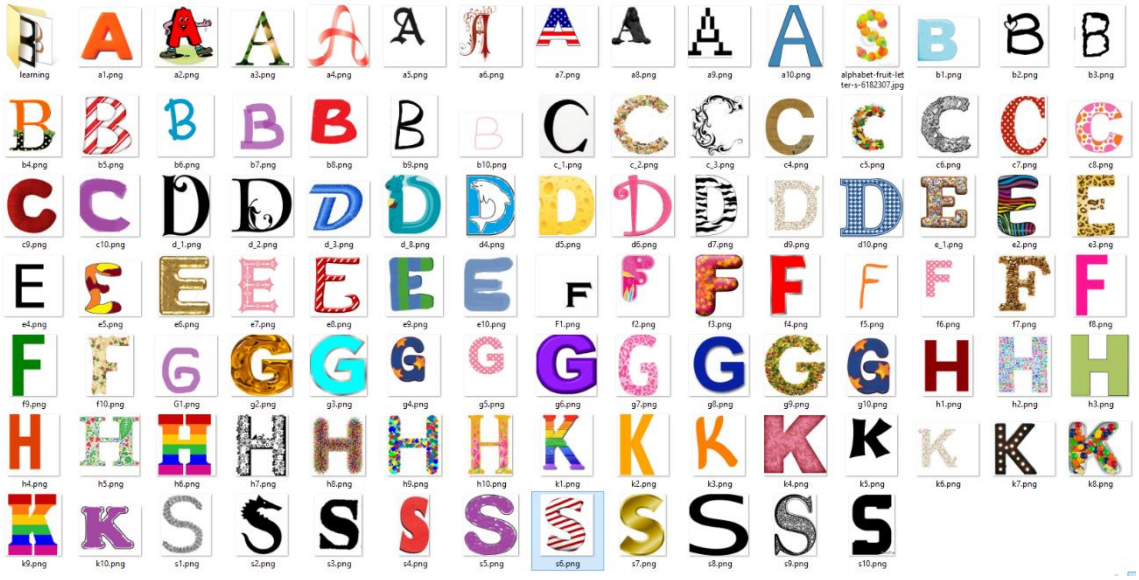
Kaust „OCR“



Kaust „KNN/learning“



Kaust „KNN“



# Lisa 2 – Programmikood MATLAB-is

**„runColor.m“**

```
source   = 'color';
images = dir(fullfile(source, '*'));
ocr=@OCR;
folder = fullfile(source);
queryFolder = fullfile('..\');
fontSize = 12;

%Tell user to run program.
message = sprintf('Kaustas %s olevatele piltidele leitakse
põhitoonid.\n Palun vali "OK", et jätkata ning oota kuni pilte
töödeldakse. :\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','K');

allResults = cell(numel(images)-3,2);
if strcmpi(reply, 'OK')
    for k = 4:numel(images)
        img = imread(fullfile(source, images(k).name));

        colorsInImg=findAvgColor(img);

        allResults{k-3,2}=colorsInImg;
        allResults{k-3,1}=images(k).name;
    end
else
    return;
end

%Tell user to run select a query image.
message = sprintf('Kaustas %s olevatele piltidele on põhitoonid
määratud.\n Palun vali otsingu aluseks olev pilt.\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%Characteristics are found, user browse for the query image
cd(folder);
[baseFileName, folder] = uigetfile('*.*', 'Palun vali sobiv fail.');
fullImageFileName = fullfile(folder, baseFileName);
cd(queryFolder);

%try to read input file
try
    queryImage=imread(fullImageFileName);
catch e
    message = sprintf('Valitud fail peab olema pildi formaadis :\n%s',
fullImageFileName);
        WarnUser(message);
        return;
end


queryColors=findAvgColor(queryImage);
```

```
%check for all results
[results_rows, results_columns]=size(allResults);
compareResults = cell(results_rows,2);

%Tell user that program is starting to compare images
message = sprintf('Valitud pildile %s\n on põhitoonid määratud.\n
Palun oota, kuni kuvatakse  sobivad vasted.\n',...
    fullImageFileName);
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%copmpare images characteristics with the query image characteristics
for k = 1:results_rows

    imgColors=allResults{k,2};
    imgName=allResults{k,1};

    img=imread(fullfile(source, imgName));

    colorMatch=findColorMatch(imgColors,queryColors);

    compareResults{k,2}=imgName;
    compareResults{k,1}=colorMatch;

end

j=7;
compareResults=reArrangeColor(compareResults,30);
resultsSize=size(compareResults);
subplot(5, 6, 3);
imshow(queryImage, []);
title('Päringu aluseks olev reproduktsioon.', 'FontSize', fontSize);

 for i=1:resultsSize(1)
     fprintf('Pildi nimi: %s\n',compareResults{i,2});

    % Display the result images.
    img = imread(fullfile(source, compareResults{i,2}));
    j=j+1;
    subplot(5, 7, j);
    imshow(img, []);
    title(i, 'FontSize', fontSize);
 end
clear all;
```

**„runOCR.m“**

```
maxResultNr=16;
source   = 'OCR';
images = dir(fullfile(source, '*'));
ocr=@OCR;
folder = fullfile(source);
queryFolder = fullfile('..\');
fontSize = 12;

%Tell user to run program.
```

```
message = sprintf('Kaustas %s olevatelt piltidelt leitakse tekstiline
osa mustrite sobitamise metoodika alusel.\n Palun vali "OK", et
jätkata ning oota kuni pilte töödeldakse. :\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','K');

allResults = cell(numel(images)-3,2);
if strcmpi(reply, 'OK')
    for k = 4:numel(images)
        img = imread(fullfile(source, images(k).name));

        [imgPrepared,imgIsOK]=prepareForFindLabels(img);

        %find text in image
        oc=ocr(imgPrepared);
        oc=oc.findWord();

        text_img=oc.Text;

        allResults{k-3,2}=text_img;
        allResults{k-3,1}=images(k).name;
    end
else
    return;
end

%Tell user to run select a query image.
message = sprintf('Kaustas %s olevatele piltidele on tekstid leitud.\n
Palun vali otsingu aluseks olev pilt.\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%Characteristics are found, user browse for the query image
cd(folder);
[baseFileName, folder] = uigetfile('*.*', 'Palun vali sobiv fail.');
fullImageFileName = fullfile(folder, baseFileName);
cd(queryFolder);

%try to read input file
try
    queryImage=imread(fullImageFileName);
catch e
    message = sprintf('Valitud fail peab olema pildi formaadis :\n%s',
fullImageFileName);
        WarnUser(message);
        return;
end

[queryImagePrepared,imgIsOK]=prepareForFindLabels(queryImage);

ocr_queryImage=ocr(queryImagePrepared);
ocr_queryImage=ocr_queryImage.findWord();
text_queryImage=ocr_queryImage.Text;

%check for all results
[results_rows, results_columns]=size(allResults);
compareResults = cell(results_rows,2);

%Tell user that program is starting to compare images
```

```
message = sprintf('Valitud pildile %s\n on tekst leitud.\n Palun oota,
kuni kuvatakse %s sobivamat vastet.\n',...
    fullImageFileName,num2str(maxResultNr-1));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%copmpare images characteristics with the query image characteristics
for k = 1:results_rows

    img_text = allResults{k,2};
    imgName=allResults{k,1};

    img=imread(fullfile(source, imgName));

    %compare text
    coef=findCoef(text_queryImage,img_text);

    compareResults{k,2}=imgName;
    compareResults{k,1}=num2str(coef);

end

j=5;
compareResults=reArrange(compareResults,maxResultNr);
resultsSize=size(compareResults);
subplot(3, 5, 3);
imshow(queryImage, []);
title('Päringu aluseks olev reproduktsioon.', 'FontSize', fontSize);

 for i=2:resultsSize(1)
     fprintf('Lõpptulemus: %s,Pildi nimi: %s\n',...
         compareResults{i,1},compareResults{i,2});

    % Display the result images.
    img = imread(fullfile(source, compareResults{i,2}));
    j=j+1;
    subplot(4, 5, j);
    imshow(img, []);
    title(i-1, 'FontSize', fontSize);
 end
clear all;
```

**„runKNN.m“**

```
source  = 'KNN/learning';
letters = dir(fullfile(source, '*'));
queryFolder = fullfile('KNN/');
homeFoler= fullfile('..\');

%find features
[values,features]=findFeaturesForKNN(letters,source);

%learning
X=features;
Y=values;
Mdl = fitcknn(X,Y,'NumNeighbors',10);

%----------------search
```

```
%Tell user to run select a query image.
message = sprintf('Kaustas %s olevatele piltidele on karakteristikud
määratud.\n Palun vali otsingu aluseks olev pilt.\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%Characteristics are found, user browse for the query image
cd(queryFolder);
[baseFileName, queryFolder] = uigetfile('*.*', 'Palun vali sobiv
fail.');
fullImageFileName = fullfile(queryFolder, baseFileName);
cd(homeFoler);

%try to read input file
try
    queryImage=imread(fullImageFileName);
catch e
    message = sprintf('Valitud fail peab olema pildi formaadis :\n%s',
fullImageFileName);
    WarnUser(message);
    return;
end

%find features without values
queryImage=prepareForFindLabels(queryImage);
queryImage=bwmorph(queryImage,'skel',1);
queryFeatures=findImgFeatures(queryImage);

%Tell user to select K.
prompt={'Mis on K väärtus?'};
% Create all your text fields with the questions specified by the
variable prompt.
title='K väärtus';
% The main title of your input dialog interface.
k=inputdlg(prompt,title);

if not(isempty(k{1}))
    searchModel = createns(X); %prepare for searching
    IdxES = knnsearch(searchModel,queryFeatures,'K',round(numel(k)));
%search
    letterId=mode(IdxES);
    letter=values(letterId);
    letter=letter{1,1};
    message = sprintf('Lähim tähemärk on %s.\n',letter);
    reply = questdlg(message, 'KNN', 'OK','OK');
else
    WarnUser('K väärtus peab olema valitud.');
    return;
end
clear all;
```

**„runLocal.m“**

```
maxResultNr=16;
source   = 'local';
images = dir(fullfile(source, '*'));
ocr=@OCR;
folder = fullfile(source);
queryFolder = fullfile('..\');
fontSize = 12;
```

```
%Tell user to run program.
message = sprintf('Kaustas %s olevatele piltidele leitakse lokaalsed
karakteristikud.\n Palun vali "OK", et jätkata ning oota kuni pilte
töödeldakse. :\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','K');

allResults = cell(numel(images)-3,2);
if strcmpi(reply, 'OK')
    for k = 4:numel(images)
        img = imread(fullfile(source, images(k).name));

        surfFeatures=findSurfFeatures(img);

        allResults{k-3,2}=surfFeatures;
        allResults{k-3,1}=images(k).name;
    end
else
    return;
end

%Tell user to run select a query image.
message = sprintf('Kaustas %s olevatele piltidele on lokaalsed
karakteristikud määratud.\n Palun vali otsingu aluseks olev
pilt.\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%Characteristics are found, user browse for the query image
cd(folder);
[baseFileName, folder] = uigetfile('*.*', 'Palun vali sobiv fail.');
fullImageFileName = fullfile(folder, baseFileName);
cd(queryFolder);

%try to read input file
try
    queryImage=imread(fullImageFileName);
catch e
    message = sprintf('Valitud fail peab olema pildi formaadis :\n%s',
fullImageFileName);
        WarnUser(message);
        return;
end

query_surfFeatures=findSurfFeatures(queryImage);

%check for all results
[results_rows, results_columns]=size(allResults);
compareResults = cell(results_rows,2);

%Tell user that program is starting to compare images
message = sprintf('Valitud pildile %s\n on lokaalsed karakteristikud
määratud.\n Palun oota, kuni kuvatakse %s sobivamat vastet.\n',...
    fullImageFileName,num2str(maxResultNr-1));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%copmpare images characteristics with the query image characteristics
for k = 1:results_rows
```

```
    img_surfFeatures=allResults{k,2};
    imgName=allResults{k,1};

    img=imread(fullfile(source, imgName));

    %compare surf features

surfResult=findSurfMatch(query_surfFeatures,img_surfFeatures,queryImag
e,img);

    compareResults{k,2}=imgName;
    compareResults{k,1}=num2str(surfResult);

end

j=5;
compareResults=reArrange(compareResults,maxResultNr);
resultsSize=size(compareResults);
subplot(3, 5, 3);
imshow(queryImage, []);
title('Päringu aluseks olev reproduktsioon.', 'FontSize', fontSize);

 for i=2:resultsSize(1)
     fprintf('Lõpptulemus: %s,Pildi nimi: %s\n',...
         compareResults{i,1},compareResults{i,2});

    % Display the result images.
    img = imread(fullfile(source, compareResults{i,2}));
    j=j+1;
    subplot(4, 5, j);
    imshow(img, []);
    title(i-1, 'FontSize', fontSize);
 end
clear all;
```

**„runGlobal.m“**

```
maxResultNr=16;
source   = 'global';
images = dir(fullfile(source, '*'));
ocr=@OCR;
folder = fullfile(source);
queryFolder = fullfile('..\');
fontSize = 12;

%Tell user to run program.
message = sprintf('Kaustas %s olevatele piltidele leitakse globaalsed
karakteristikud.\n Palun vali "OK", et jätkata ning oota kuni pilte
töödeldakse. :\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','K');

allResults = cell(numel(images)-3,2);
if strcmpi(reply, 'OK')
    for k = 4:numel(images)
        img = imread(fullfile(source, images(k).name));
```

```
        zernikeFeatures=findZernikeFeatures(img);
        allResults{k-3,2}=zernikeFeatures;
        allResults{k-3,1}=images(k).name;
    end
else
    return;
end

%Tell user to run select a query image.
message = sprintf('Kaustas %s olevatele piltidele on globaalsed
karakteristikud määratud.\n Palun vali otsingu aluseks olev
pilt.\n',...
    fullfile(source, '*'));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%Characteristics are found, user browse for the query image
cd(folder);
[baseFileName, folder] = uigetfile('*.*', 'Palun vali sobiv fail.');
fullImageFileName = fullfile(folder, baseFileName);
cd(queryFolder);

%try to read input file
try
    queryImage=imread(fullImageFileName);
catch e
    message = sprintf('Valitud fail peab olema pildi formaadis %s:\n',
fullImageFileName);
        WarnUser(message);
        return;
end

zernikeFeaturesQuery=findZernikeFeatures(queryImage);

%check for all results
[results_rows, results_columns]=size(allResults);
compareResults = cell(results_rows,2);

%Tell user that program is starting to compare images
message = sprintf('Valitud pildile %s\n on globaalsed karakteristikud
määratud.\n Palun oota, kuni kuvatakse %s sobivamat vastet.\n',...
    fullImageFileName,num2str(maxResultNr-1));
reply = questdlg(message, 'Programmi käivitamine.', 'OK','OK');

%copmpare images characteristics with the query image characteristics
for k = 1:results_rows

    imgName=allResults{k,1};
    zernikeFeaturesImg=allResults{k,2};

    img=imread(fullfile(source, imgName));


zernikeResult=findZernikeMatch(zernikeFeaturesQuery,zernikeFeaturesImg
);

    compareResults{k,2}=imgName;
    compareResults{k,1}=num2str(zernikeResult);

end
```

```
j=5;
compareResults=reArrange(compareResults,maxResultNr);
resultsSize=size(compareResults);
subplot(3, 5, 3);
imshow(queryImage, []);
title('Päringu aluseks olev reproduktsioon.', 'FontSize', fontSize);

 for i=2:resultsSize(1)
     fprintf('Lõpptulemus: %s,Pildi nimi: %s\n',...
         compareResults{i,1},compareResults{i,2});

    % Display the result images.
    img = imread(fullfile(source, compareResults{i,2}));
    j=j+1;
    subplot(4, 5, j);
    imshow(img, []);
    title(i-1, 'FontSize', fontSize);
 end
clear all;
```

**„create_templates.m“**

```
%CREATE TEMPLATES
%Letter
clc;
close all;
A=imread('letters_numbers\A.bmp');B=imread('letters_numbers\B.bmp');
C=imread('letters_numbers\C.bmp');D=imread('letters_numbers\D.bmp');
E=imread('letters_numbers\E.bmp');F=imread('letters_numbers\F.bmp');
G=imread('letters_numbers\G.bmp');H=imread('letters_numbers\H.bmp');
I=imread('letters_numbers\I.bmp');J=imread('letters_numbers\J.bmp');
K=imread('letters_numbers\K.bmp');L=imread('letters_numbers\L.bmp');
M=imread('letters_numbers\M.bmp');N=imread('letters_numbers\N.bmp');
O=imread('letters_numbers\O.bmp');P=imread('letters_numbers\P.bmp');
Q=imread('letters_numbers\Q.bmp');R=imread('letters_numbers\R.bmp');
S=imread('letters_numbers\S.bmp');T=imread('letters_numbers\T.bmp');
U=imread('letters_numbers\U.bmp');V=imread('letters_numbers\V.bmp');
W=imread('letters_numbers\W.bmp');X=imread('letters_numbers\X.bmp');
Y=imread('letters_numbers\Y.bmp');Z=imread('letters_numbers\Z.bmp');
%lower case letters
a=imread('letters_numbers\a.png');b=imread('letters_numbers\b.png');
c=imread('letters_numbers\c.png');d=imread('letters_numbers\d.png');
e=imread('letters_numbers\e.png');f=imread('letters_numbers\f.png');
g=imread('letters_numbers\g.png');h=imread('letters_numbers\h.png');
i=imread('letters_numbers\i.png');j=imread('letters_numbers\j.png');
k=imread('letters_numbers\k.png');l=imread('letters_numbers\l.png');
m=imread('letters_numbers\m.png');n=imread('letters_numbers\n.png');
o=imread('letters_numbers\o.png');p=imread('letters_numbers\p.png');
q=imread('letters_numbers\q.png');r=imread('letters_numbers\r.png');
s=imread('letters_numbers\s.png');t=imread('letters_numbers\t.png');
u=imread('letters_numbers\u.png');v=imread('letters_numbers\v.png');
w=imread('letters_numbers\w.png');x=imread('letters_numbers\x.png');
y=imread('letters_numbers\y.png');z=imread('letters_numbers\z.png');


%Number
one=imread('letters_numbers\1.bmp');
two=imread('letters_numbers\2.bmp');
```

```
three=imread('letters_numbers\3.bmp');four=imread('letters_numbers\4.b
mp');
five=imread('letters_numbers\5.bmp');
six=imread('letters_numbers\6.bmp');
seven=imread('letters_numbers\7.bmp');eight=imread('letters_numbers\8.
bmp');
nine=imread('letters_numbers\9.bmp');
zero=imread('letters_numbers\0.bmp');
%-*-*-*-*-*-*-*-*-*-*-*-
letter=[A B C D E F G H I J K L M...
    N O P Q R S T U V W X Y Z];
number=[one two three four five...
    six seven eight nine zero];

lowercase = [a b c d e f g h i j k ...
     l m n o p q r s t u v w x y z];
character=[letter number lowercase];
templates=mat2cell(character,42,[24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 24 ...
    24 24]);
save ('templates','templates')
clear all
```

**„clip.m“**

```
function [img_out ,f ,c]=clip(img_in)
[f ,c]=find(img_in);
img_out=img_in(min(f):max(f),min(c):max(c));
```

**„create_similarity_table.m“**

```
function st=create_similarity_table()
symbol={'0';'6';'7';'O';'l';'v';'f'};
meaning = {'g';'s';'t';'g';'i';'u';'t'};

st=table(symbol,meaning);
```

**„findAvgColor.m“**

```
function mainColorsInImg=findMainColors(img)
    mainColorsInImg=[];
    colors=[];
    percents=[];
    myColorsMap = [0 0 0;0.1 0.1 0.1;0.2 0.2 0.2; 0 0 1;0 0 0.9;0 0
0.8;0 0 0.7;...
    0 0 0.6;0 0 0.5;0 0 0.4;0 0 0.3;0.3 0.3 0.7;0 1 0;0 0.9 0;0 0.8
0;0 0.7 0;...
    0 0.6 0;0 0.5 0;0 0.4 0;0 0.3 0;0.3 1 0.3;0 1 1;0 0.9 0.9;0 0.8
0.8;0 0.7 0.7;...
    0.3 0.8 0.8;1 0 0;0.9 0 0;0.8 0 0;0.7 0 0;0.6 0 0;0.5 0 0;0.4 0
0;0.3 0 0;...
    1 0 1;0.9 0 0.9;0.8 0 0.8;0.7 0 0.7;0.6 0 0.6;1 1 0;0.9 0.9 0;0.8
0.8 0;0.3 0.3 0.3;...
```

```
    0.4 0.4 0.4;0.5 0.5 0.5;0.6 0.6 0.6;0.7 0.7 0.7;0.8 0.8 0.8;1 1
1];

colorsInMap=[1,1,1,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,2,2,2,2,2,4,4,4
,4,4,4,4,4,...
        5,5,5,5,5,6,6,6,8,8,8,8,8,8,8];
    img=rgb2ind(img,myColorsMap,'nodither');
    [len width]=size(img);
    whitePixels=0;

    for i=1:len*width
        colors(i)=colorsInMap(img(i)+1);
        if colors(i)==8
         whitePixels=whitePixels+1;
        end
    end

    [frequencies,colors]=hist(colors,unique(colors));
    finalLen=1;
    areaWithoutWhite=(len*width)-whitePixels;
    c=1;
    for i=1:length(colors)
        if colors(i)==1
            limit=0.15;
        else
            limit=0.005;
        end
        percent=round(frequencies(i)/areaWithoutWhite,4);
        if percent>limit&& not(colors(i)==8)
            mainColorsInImg(finalLen)=colors(i);
            finalLen=finalLen+1;
            c=c+1;
        end
    end
end
```

**„findCoef.m“**

```
function coef=findCoef(text1,text2)
    coef=0;
    if strcmp(text1,text2)==0
        weight=findWordCoef(text1,text2);
        coef=round(weight/length(text1),3); %sõnade kokkulangevuse
protsent (0..1)
    else
        if not(isempty(text1)) && not(isempty(text1))
           coef=1;
        else
           coef=0;
        end
    end
    if coef <0.1
        coef = 0;
    end
end
```

**„findWordCoef.m“**

```
function weight=findWordCoef(word1,word2)
    st=create_similarity_table();
```

```
    if length(word1)>=length(word2)
        max=word1;
        min=word2;
    else
        max = word2;
        min= word1;
    end
    for j=1:length(st.meaning)
        minReplaced=doReplacement(min,j,st);
        maxReplaced=doReplacement(max,j,st);
    end
    weight=0;
    weight=findBestWeight(min,max,weight);
    weight=findBestWeight(minReplaced,max,weight);
    weight=findBestWeight(minReplaced,maxReplaced,weight);
    weight=findBestWeight(min,maxReplaced,weight);
end

function weight=findBestWeight(text1,text2,weight)
    w=findWeight(text1,text2);
    if w>weight
        weight=w;
    end
end

function weight=findWeight(min,max)
    letterMatch=0;
    matched=[];
    if isequal(min,max)
        weight=length(min);
    else
        w=0;
        weight = 0;
        for i=1:length(min)-2
            w=0;
            for j=i+2:length(min)
                if regexpi(max,min(i:j))>=1
                    if j-i+1> w
                        w=j-i+1;
                    end
                end
            end
            matched(i)=w;
        end
        for i=1:length(matched)
            if i>1
                if matched(i-1)==0 && not(matched(i)==0)
                    weight= weight+matched(i);
                end
            elseif matched(i)>weight
                 weight = matched(i);
            end
        end
    end
end

function text=doReplacement(text,j,st)
    for i=1:length(text)
        if text(i)==st.symbol{j}
            if i>1
```

```
                text=[text(1:i-1) st.meaning{j}
text(i+1:length(text))];
            else
                text=[st.meaning{i} text(i+1:length(text))];
            end
        end
    end
end
```

**„findColorMatch.m“**

```
function match=findColorMatch(imgColors,queryColors)

lenQuery=length(queryColors);
lenImg=length(imgColors);

match=0;
m=0;

if lenQuery==lenImg
    for i=1:lenQuery
        for j=1:lenImg
            if queryColors(i)==imgColors(j)
                m=m+1;
            end
        end
    end
    if m==lenQuery
        match=1;
    end
end
end
```

**„findColorMatch.m“**

```
function [values,allFeatures]=findFeaturesForKNN(images,source)
values = cell(numel(images)-3,1);
allFeatures=[];
 for k = 4:numel(images)
    img = imread(fullfile(source, images(k).name));
    img=prepareForFindLabels(img);
    img=bwmorph(img,'skel',1);
    features=findImgFeatures(img);
    allFeatures=[allFeatures;features];

    if (k-3)<=25
        values{k-3}='A';
    elseif (k-3)>=26 && (k-3)<=50
        values{k-3}='B';
    elseif (k-3)>=51 && (k-3)<=75
        values{k-3}='C';
    elseif (k-3)>=76 && (k-3)<=100
        values{k-3}='D';
    elseif (k-3)>=101 && (k-3)<=125
        values{k-3}='E';
    elseif (k-3)>=126 && (k-3)<=150
        values{k-3}='F';
    elseif (k-3)>=151 && (k-3)<=175
        values{k-3}='G';
```

```
    elseif (k-3)>=176 && (k-3)<=200
        values{k-3}='H';
    elseif (k-3)>=201 && (k-3)<=225
        values{k-3}='K';
    else
        values{k-3}='S';
    end
  end
end
```

**„findImgFeatures.m“**

```
function features=findImgFeatures(img)
    img=imresize(img,[256 256]);
    [len width]=size(img);
    l8=len/8;
    w4=width/4;
    l=1;
    w=1;
    features=[];
    counter=1;
    for j=1:4
        for i=1:8
            pic=img(l:l+l8-1,w:w+w4-1);
            features(counter)=mean(pic(:));
            counter=counter+1;
            l=l+l8;
        end
        l=1;
        w=w+w4;
    end
end
```

**„findLines.m“**

```
function [fl re x x_re rowLen]=findLines(img_remained)
    fl=[];
    re=[];
    x=[];
    x_re=[];
    rowLen=0;

    [img f]=clip(img_remained);
    x1=min(f);
    num_filas=size(img,1);

    for s=1:num_filas
        if sum(img(s,:))==0
            nm=img(1:s-1, :);
            rm=img(s:end, :);
            [fl] = clip(nm);
            x=x1+(s-1);
            [re f]=clip(rm);
             x_re=min(f);
            rowLen=x-x1;
            break
        else
            fl=img;
            x_re=0;
            x=0;
```

```
            re=[ ];
            rowLen=0;
        end
    end
end
```

**„findRows.m“**

```
classdef findRows

properties
    Image
    Rows
end

methods
    function obj = findRows(image)
            % class constructor
            if (nargin > 0)
                obj.Image = image;
            end
    end
    function obj=find(obj)
        % Convert to gray scale
        if size(obj.Image,3)==3 %RGB image
            obj.Image=rgb2gray(obj.Image);
        end
        % Convert to BW
        threshold = graythresh(obj.Image);
        obj.Image =~im2bw(obj.Image,threshold);
        % Remove all object containing fewer than 30 pixels
        obj.Image = bwareaopen(obj.Image,30);
        re=obj.Image;
        while 1
            %Fcn 'lines' separate lines in text
            [fl,re]=lines(re);

            if isempty(re)%See variable 're' in Fcn 'lines'
                break;
            end
        end
    end
end
end
```

**„makeSurfFeatures.m“**

```
function [imgFeatures,imgPoints]=makeSurffeatures(img)
    if size(img,3)==3
        img=rgb2gray(img);
    end
    surfPoints = detectSURFFeatures(img);
    [imgFeatures,imgPoints] = extractFeatures(img,surfPoints);
end
```

**„findSurffeatures.m“**

```
function surfFeatures=findSurfFeatures(img)

    surfFeatures = cell(1,2);
```

```
  % Convert to gray scale
    if size(img,3)==3 %RGB image
        img=rgb2gray(img);
    end

    surfFeatures=makeSurfFeatures(img,surfFeatures);
end

function surfFeatures=makeSurfFeatures(img,surfFeatures)
    [imgFeatures,imgPoints]=makeSurffeatures(img);
    [len width]=size(surfFeatures);
    surfFeatures{len+1,1}=imgFeatures;
    surfFeatures{len+1,2}=imgPoints;
end

function [imgFeatures,imgPoints]=makeSurffeatures(img)
    if size(img,3)==3
        img=rgb2gray(img);
    end
    surfPoints = detectSURFFeatures(img,'MetricThreshold',1);
    [imgFeatures,imgPoints] = extractFeatures(img,surfPoints);
end
```

**„findSurfMatch.m“**

```
function result=findSurfMatch(queryFeatures,imgFeatures,query,img)
    result=0;
    [lenQ,widthQ]=size(queryFeatures);
    [lenI,widthI]=size(imgFeatures);
    for i=1:lenQ
        for j=1:lenI

r=findSurf(imgFeatures{j,1},queryFeatures{i,1},imgFeatures{j,2},queryF
eatures{i,2},query,img);
            if r>result
                result=r;
            end
        end
    end

end

function
result=findSurf(surfFeatures_img,surfFeatures_original,imgPoints,origi
nal_surfPoints,query,img)
    result=0;
     % Convert to gray scale
    if size(img,3)==3 %RGB image
        img=rgb2gray(img);
    end
    if size(query,3)==3 %RGB image
        query=rgb2gray(query);
    end



    if not(isempty(surfFeatures_img)) &&
not(isempty(surfFeatures_original))
```

```
        %Retrieve the locations of matched points.
        [indexPairs,matchmetric] =
matchFeatures(surfFeatures_original,surfFeatures_img,'unique',true,'Me
tric','SSD');

        if length(matchmetric)>0
            matchedPointsQuery =
original_surfPoints(indexPairs(:,1),:);
            matchedPointsImg = imgPoints(indexPairs(:,2),:);
            try
             [tform, inlierQueryPoints, inlierImgPoints] = ...
                 estimateGeometricTransform(matchedPointsQuery,
matchedPointsImg, 'similarity');
                 result=length(inlierQueryPoints);
            catch e
                result=0;
            end
        end
    else
        result=0;
    end
end
```

**„findZernikeFeatures.m“**

```
function zernikeFeatures=findZernikeFeatures(img)
    if size(img,3)==3
        img=rgb2gray(img);
    end
    threshold=graythresh(img);
    img =~im2bw(img,0.95);

    img = bwareaopen(img,50);

    img=clip(img);
    img = logical(not(img));
    img=imresize(img,[200 200]);

    zernikeFeatures=[];

    n=0;
    m=0;
    new=false;
    for i=1:36
        if new==true
            m=m+1;
            n=m;
        end
        magnitude = Zernikmoment(img,n,m);
        zernikeFeatures(i)=magnitude;
         new=false;
         if i==6 || i==11 || i==16 || i==20 || i==24 || ...
                 i==27 || i==30 || i==32 || i==34 || i==35 || i==36
             new=true;
         end
         n=n+2;
    end

end
```

**„findZernikeMatch.m“**

```
function zernikeCoef=findZernikeMatch(featuresQ,featuresImg)
    [indexPairs,matchmetric] =
matchFeatures(featuresQ,featuresImg,'MaxRatio',1,'Metric','SSD');
    value=100;
    for i=1:length(matchmetric)
        if matchmetric(i)<value
            value=matchmetric(i);
        end
    end
    if not(value==100)
        zernikeCoef=1-value;
    else
        zernikeCoef=0;
end
```

**„labelling.m“**

```
 function [L,Ne]=labelling(Image)
    Image = im2uint8(Image);
    L = bwlabel(Image);
    s = regionprops(L, 'BoundingBox', 'Extrema',
'Centroid','PixelIdxList');
    extrema = cat(1, s.Extrema);
    left_most_bottom = extrema(6:8:end, :);
    left = left_most_bottom(:, 1);
    bottom = left_most_bottom(:, 2);
    % quantize the bottom coordinate
    bottom = 6 * round(bottom / 6);
    [sorted, sort_order] = sortrows([bottom left]);
    s2 = s(sort_order);
    Ne=length(sorted);
    for k = 1:numel(s2)
       kth_object_idx_list = s2(k).PixelIdxList;
       L(kth_object_idx_list) = k;
    end
end
```

**„lines.m“**

```
function [xArray,rowLengths]=lines(img)
    re=img;
    count=0;
    x_re=0;
    rowLengths = [];
    xArray=[];

    while 1
        count=count+1;
        re_x=x_re;
        size_re= size(re);

        [fl re x x_re rowLen]=findLines(re);

            xArray(count)=x + re_x;
            rowLengths(count)=rowLen;
            if isempty(re)
                if xArray==0
```

```
                    [img_out f]= clip(img);
                    xArray(1)=max(f);
                    rowLengths(1)=max(f)-min(f);
                end
                break;
            end
    end
end
```

**„msgboxw.m“**

```
function msgboxw(message)
    uiwait(msgbox(message));
    return; % from msgboxw()
end
```

**„WarnUser.m“**

```
function WarnUser(warningMessage)
    uiwait(warndlg(warningMessage));
    return; % from WarnUser()
end
```

**„OCR.m“**

```
classdef OCR

properties
    Image
    Text
    templates
end

methods
    function obj = OCR(image)
            if (nargin > 0)
                obj.Image = image;
            end
    end
    function obj=findWord(obj)
        word=[ ];
        re=obj.Image;
        load templates
        obj.templates = templates;
        num_letras=size(obj.templates,2);

        while 1
            [fl re x x_re rowLen]=findLines(re);
            imgn=fl;
            if not(isempty(imgn))
                [L,Ne] = labelling(imgn);
                for n=1:Ne
                    [r,c] = find(L==n);
                    if not(isempty(r)) || not(isempty(c))
                        n1=imgn(min(r):max(r),min(c):max(c));
                        img_r=imresize(n1,[42 24]);
                        letter=read_letter(img_r,num_letras);
                        word=[word letter];
                        obj.Text=word;
```

```
                    end
                end
            end
            if isempty(re)
                break
            end
        end
    end
  end
end
```

**„preparedForFindLabels.m“**

```
function [img,imgIsOK]=prepareForFindLabels(img)
if size(img,3)==3
    img=rgb2gray(img);
end
threshold=graythresh(img);
img =~im2bw(img,threshold);
img = bwareaopen(img,50);

blackPixels=0;
DarkImg=false;
[len width] = size(img);
if sum(img(1,:))/width>0.9
    DarkImg = true;
end
img=clip(img);
[len width] = size(img);
if DarkImg == true
    for s=1:len
        for s2=1:width
            if img(s,s2)==0
                img(s,s2)=1;
                blackPixels = blackPixels+1;

            else
                img(s,s2)=0;
            end
        end
    end
end
if blackPixels == len*width
    imgIsOK = false;
else
    imgIsOK = true;
end
```

**„read_letter.m“**

```
function letter=read_letter(imagn,num_letras)
    load templates;
    comp=[ ];
     for n=1:num_letras
        sem=corr2(templates{1,n},imagn);
        comp=[comp sem];
     end
    if max(comp)>0.45
        vd=find(comp==max(comp));
    else
```

```
    vd=0;
end

if vd==1
    letter='A';
elseif vd==2
    letter='B';
elseif vd==3
    letter='C';
elseif vd==4
    letter='D';
elseif vd==5
    letter='E';
elseif vd==6
    letter='F';
elseif vd==7
    letter='G';
elseif vd==8
    letter='H';
elseif vd==9
    letter='I';
elseif vd==10
    letter='J';
elseif vd==11
    letter='K';
elseif vd==12
    letter='L';
elseif vd==13
    letter='M';
elseif vd==14
    letter='N';
elseif vd==15
    letter='O';
elseif vd==16
    letter='P';
elseif vd==17
    letter='Q';
elseif vd==18
    letter='R';
elseif vd==19
    letter='S';
elseif vd==20
    letter='T';
elseif vd==21
    letter='U';
elseif vd==22
    letter='V';
elseif vd==23
    letter='W';
elseif vd==24
    letter='X';
elseif vd==25
    letter='Y';
elseif vd==26
    letter='Z';
    %*-*-*-*-*
elseif vd==27
    letter='1';
elseif vd==28
    letter='2';
elseif vd==29
```

```
    letter='3';
elseif vd==30
    letter='4';
elseif vd==31
    letter='5';
elseif vd==32
    letter='6';
elseif vd==33
    letter='7';
elseif vd==34
    letter='8';
elseif vd==35
    letter='9';
elseif vd==36
    letter='0';
    %********
elseif vd==37
    letter='a';
elseif vd==38
    letter='b';
elseif vd==39
    letter='c';
elseif vd==40
    letter='d';
elseif vd==41
    letter='e';
elseif vd==42
    letter='f';
elseif vd==43
    letter='g';
elseif vd==44
    letter='h';
elseif vd==45
    letter='i';
elseif vd==46
    letter='j';
elseif vd==47
    letter='k';
elseif vd==48
    letter='l';
elseif vd==49
    letter='m';
elseif vd==50
    letter='n';
elseif vd==51
    letter='o';
elseif vd==52
    letter='p';
elseif vd==53
    letter='q';
elseif vd==54
    letter='r';
elseif vd==55
    letter='s';
elseif vd==56
    letter='t';
elseif vd==57
    letter='u';
elseif vd==58
    letter='v';
elseif vd==59
```

```
        letter='w';
    elseif vd==60
        letter='x';
    elseif vd==61
        letter='y';
    elseif vd==62
        letter='z';
    elseif vd==63
        letter='l';
    elseif vd==0
        letter='';
end
```

**„reArrange.m“**

```
function resultMatrix=reArrange(resultMatrix,maxResultNr)
    emptyCells = cellfun('isempty', resultMatrix);
    resultMatrix(all(emptyCells,2),:) = [];

    [~,sortedMatrix] =
sort(str2double(resultMatrix(:,1)),1,'descend');
    resultMatrix = resultMatrix(sortedMatrix,:);
    sizeResult= size(resultMatrix);
    if maxResultNr<sizeResult(1)
    resultMatrix = resultMatrix(1:maxResultNr,:);
    end
end
```

**„reArrangeColor.m“**

```
function resultMatrix=reArrangeColor(resultMatrix,maxResultNr)
    emptyCells = cellfun('isempty', resultMatrix);
    resultMatrix(all(emptyCells,2),:) = [];

    resultMatrix(cellfun(@(x) ~x(1),resultMatrix(:,1)),:) = [];
    sizeResult= size(resultMatrix);
    if maxResultNr<sizeResult(1)
        resultMatrix = resultMatrix(1:maxResultNr,:);
    end

end
```

**„Zernikemoment.m“**

```
function A = Zernikmoment(p,n,m)

    N = size(p,1);
    x = 1:N; y = x;
    [X,Y] = meshgrid(x,y);
    R = sqrt((2.*X-N-1).^2+(2.*Y-N-1).^2)/N;
    Theta = atan2((N-1-2.*Y),(2.*X-N+1));
    R = (R<=1).*R;
    Rad = radialpoly(R,n,m);    % Rnm

    Product = p(x,y).*Rad.*exp(-1i*m*Theta); %Vnm
    Z = sum(Product(:));

    cnt = nnz(R)+1;             % nr. of non zero el. in img
```

```
    Z = (n+1)*Z/cnt;            % Znm
    A = abs(Z);                 % calculate the amplitude of the
moment

end

function rad = radialpoly(r,n,m)
    rad = zeros(size(r));
    for s = 0:(n-abs(m))/2
      c = (-1)^s*factorial(n-s)/(factorial(s)*factorial((n+abs(m))/2-
s)*...
          factorial((n-abs(m))/2-s));
      rad = rad + c*r.^(n-2*s);
    end
end
```