TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

**Aramaa, Aarne**

# Rasterkujul kaartidelt sümbolite tuvastamine

**Magistritöö (30 EAP)**

Juhendaja: Vambola Leping

Tartu 2016

## Rasterkujul kaartidelt sümbolite tuvastamine

**Lühikokkuvõte:**

Antud töös uuritakse masinnägemise erinevaid meetodeid ja tehnikaid ning rakendatakse neid raster- ehk pildikujul esinevate kaartide (nii ajalooliste, kui kaasaegsete) töötlemisel, et tuvastada aluskaartidelt erinevaid sümboleid. Töö käigus valmib rasterkujul kaartidelt sümboleid eraldava ja neid grupeeriva programmi prototüüp. Töös kirjeldatakse kuidas kaartide digiteerimise protsessi on võimalik lihtsustada.

**Võtmesõnad:**

Kaardid, sümbolituvastus, masinnägemine

**CERCS:** P175 (Informaatika, süsteemiteooria)

## Detection of symbols from raster maps

**Abstract:**

In this paper we describe research into using machine vision techniques to detect and group similar symbols from maps that are in raster form. As a result of this thesis a prototype of a program that is capable of discovering and grouping clusters of detected symbols from a raster map is made. This paper also describes how the map digitization process could be simplified.

**Keywords:**

Maps, symbol detection, machine vision

**CERCS:** P175 (Informatics, systems theory)

# Sisukord

# Sissejuhatus

Kaarte esineb mitmel erineval kujul, nii digitaalselt kui ka paberil. Digitaalselt on kaardid enamasti rasterkujul ehk piltidena. Kaarte vaadatakse tavaliselt mingis kindlas kaardirakenduses. Nimetatud rakendustes on aluskaardid rasterkujul, ning sümbolid ja muud märked on nende kohal eraldi kihtidena, kas vektor- või rasterkujul, see lubab nende filtreerimist (sisse- ja väljalülitamist) ning muud töötlemist, nagu geograafilise analüüsi teostamist. Kui paberkujul kaarte (nagu näiteks ajaloolisi kaarte) skaneeritakse arvutisse, jääb aluskaart koos kõikide kaardil olevate sümbolite ja märgetega ühekihiliseks rasterkujul kaardiks, ilma võimaluseta sümboleid filtreerida, otsida ega muudmoodi geoinfosüsteemidele (GIS) [1] omaselt töödelda.

Et paberkujul kaarte ja nendel olevat informatsiooni kasutada, koos teiste geoinfosüsteemidele omaste protsessidega, tuleb nad digiteerida. Digiteerimine on protsess, mille käigus tõlgendatakse kaartidel olev geograafiline informatsioon töödeldavale kujule. Digiteerimine on üheks tähtsaks tööks, millega GIS-spetsialistid igapäevaselt tegelema peavad. On olemas mõningaid tööriistasid, mis aitavad digiteerimise protsessi, kuid kogu tegevus sarnaneb siiski liialt kaardi uuesti joonistamisele ja vähemalt osad sammud sellest protsessist võiksid olla automatiseeritud.

Antud töös kirjeldame lühidalt seni kasutusel olevaid digiteerimise tööriistasid ning uurime erinevaid masinnägemise tehnikaid, mis aitavad rasterkujul kaartidelt märgistusi taustast eraldada, sümboleid üksteisest eristada ning sarnasuse alusel grupeerida, et neid paigutada GIS rakenduste standarditele vastavatele kaartidele. See lubaks nendega sooritada põhilisi operatsioone, mida geograafilise infoga tehakse, nagu näiteks sümbolite kihtide filtreerimine kuvamisel, asukohtade leidmine märgistuse alusel või geograafilise informatsiooni analüüs ja töötlemine.

Antud töö eesmärgiks on implementeerida programmi prototüüp, mis on võimeline rasterkujul kaardilt eraldada ja sarnasuse alusel gruppidesse jagada kaardil olevad sümbolid. Töö käigus uurime erinevaid meetodeid, mis võivad olla abiks rasterkujul kaartide digiteerimisel, ehk kuidas eraldada ja töödelda kaartidelt märgistusi, nagu näiteks sümbolid, teed ja erinevad mõjualad, mis esinevad geomeetriliste kujundite ja joontena. Lisaks implementeeritakse töö käigus mõningaid mainitud meetodeid, et nende tõhusust antud teemal testida.

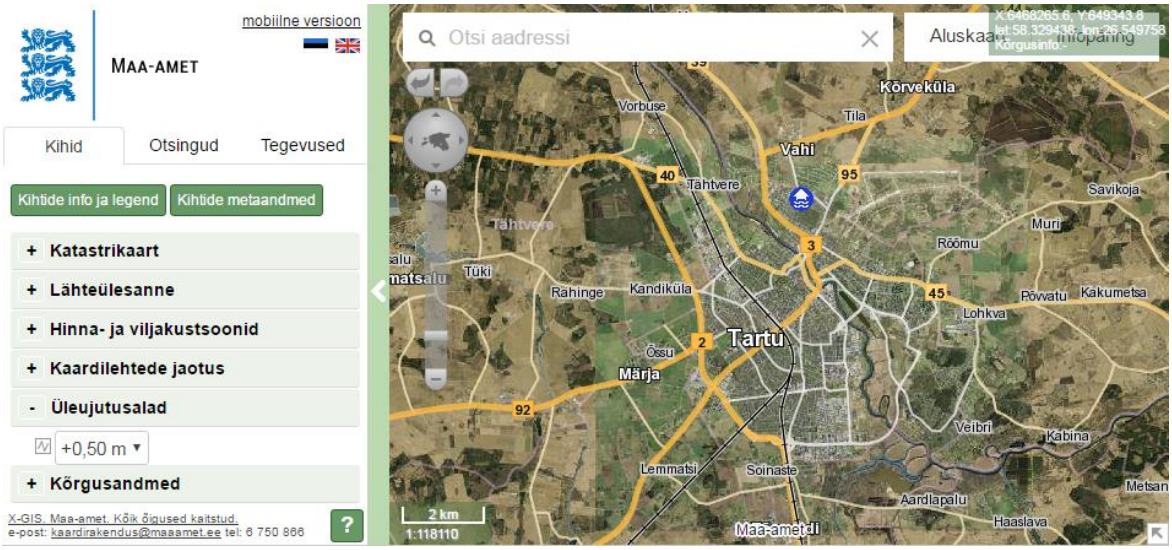
Käesolev magistritöö on jagatud 5 peatükki:

- Esimeses peatükis kirjeldatakse tööga kaasnevat tausta, kirjeldatakse seni kasutusel olevaid tööriistasid ja selgitatakse motivatsiooni antud teema uurimiseks.
- Teises peatükis kirjeldatakse mitmeid masinnägemise meetodeid Nendest meetoditest tuuakse kindlaid näiteid, millest osasid on töö käigus implementeeritud, et nende tõhusust testida ja nende tööd näidetega selgitada.
- Kolmandas peatükis kirjeldatakse geoinfo süsteeme lähemalt, kirjeldatakse millised on neile seatud standardid ja seeläbi, millega tuleks arvestada luues kaartide automaatse digiteerimise rakendust.
- Neljandas peatükis kirjeldatakse implementatsiooni ja milliste ülesannetega ja kui tõhusalt see hakkama saab. Kirjeldatakse millised on implementatsiooni teadaolevad probleemid ja kuidas neid parandada.
- Viimases peatükis arutletakse töö käigus ilmnenud võimalikest edasiarendustest, ehk millised on järgmised sammud käesoleva rakenduse täiustamisel ja kuidas võiks käesoleva töö tulemused tulevikus kasulikud olla.

# 1. Taust

Käesolev peatükk annab ülevaate tööga seotud taustast ja toob näiteid, milleks selline uurimus kasulik on. Kirjeldame, kuidas GIS rakendused reeglina töötavad ning miks ei ole soovitatav sellisel juhul kasutada vaid ühe kihina esitatavat rasterkujul kaarti.

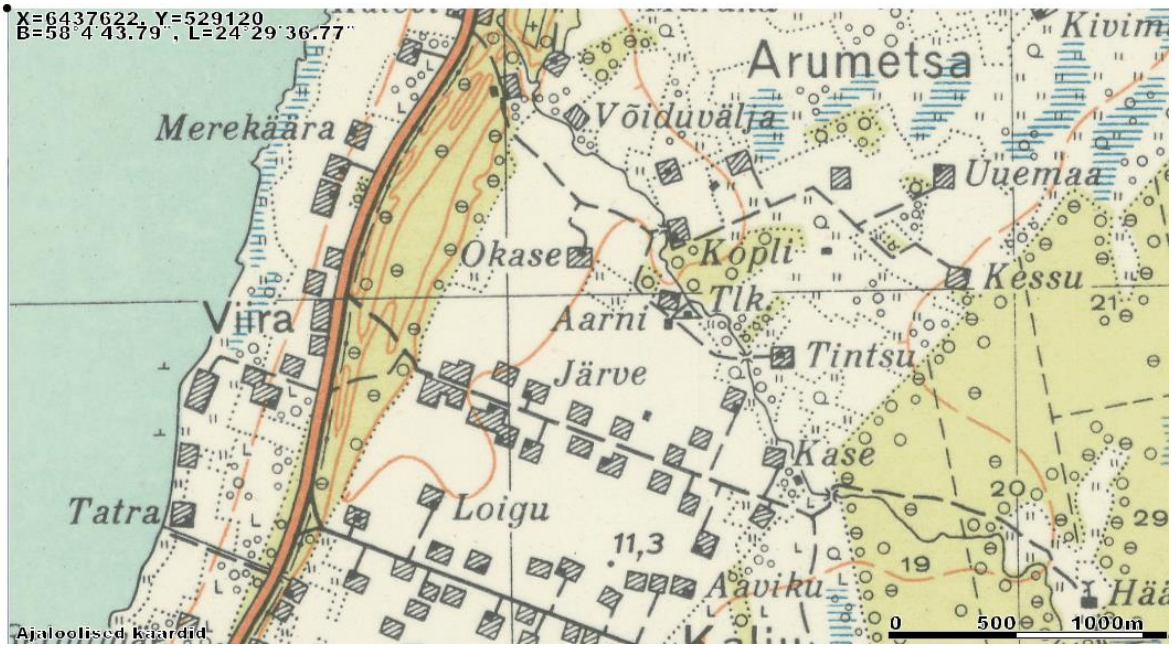
Üheks näiteks võib tuua Maaameti kaardirakenduse XGIS[1].




Joonis 1 Ekraanitõmmis Maaameti XGIS rakendusest.

Joonisel 1 on näha aluskaardina ortofotot Tartu piirkonnast ja eraldi kihtidena selle kohal põhi-, tugi- ja kõrvalmaanteesid, linna, alevite ja alevike nimesid, mis on salvestatud eraldi, mitte kõik ühe rastrina. Selline kihtide eraldi hoidmine on kasulik andmete paindlikumaks kuvamiseks, kui ka andmete muutmiseks, näiteks kui kaardilt tuleb eemaldada mingi teelõik, tuleb muuta vaid seda kihti, millel see on kujutatud, mitte tervet aluskaardi rastrit. On olemas juhtusid, kui kaardid on salvestatud ühe kihina, mis ei soodusta eelnevalt näitena välja toodud paindlikkust.

[1] http://xgis.maaamet.ee/maps/XGis




Joonis 2 Lõik ajaloolisest kaardist Maaameti XGIS rakendusest

Joonisel 2 on näha näidet 1939 aasta kaardist, mis on sisse skaneeritud ja nüüd Maaameti XGIS[2] rakenduses kuvatud ühekihilise rasterkaardina. Sellisel kaardil ei ole võimalik välja filtreerida näiteks kohanimetusi, hooneid ega mingi kindla sümboli järgi asukohtasid leida. Et kaardil kujutatud info oleks lihtsamini töödeldav, oleks vaja see kaart digiteerida. Digiteerimise käigus märgitaks kõik pildil olevad teed, hooned, puud, samakõrgusjooned, ning salvestataks eraldi kihtidena. Nii lubatakse neid eraldi kuvada, nende kuju muuta, kui ka nende välimust või stiili muuta. Mõneks näiteks sellise digiteeritud kaardi kasulikkusest oleksid:

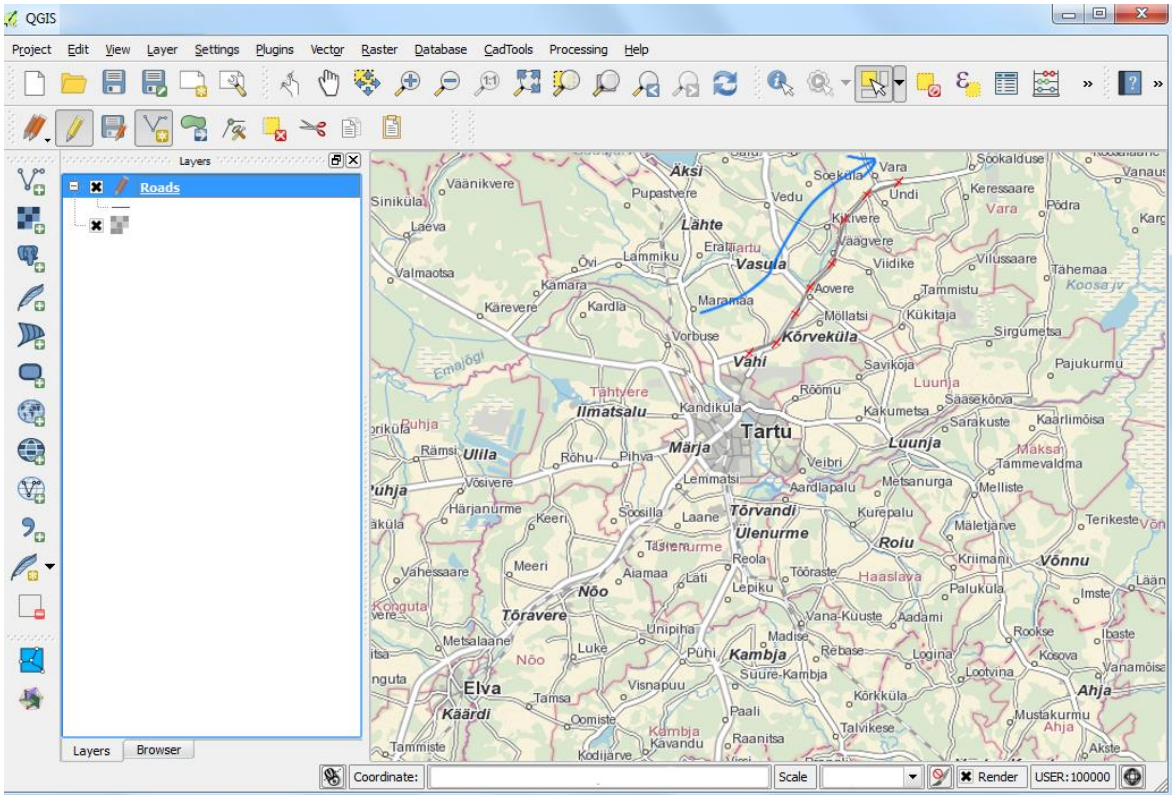
- teede salvestamine vektorandmetena (võib olla tarvilik ka juba eraldi rasterkihina salvestatud teede digiteerimisel vektoriteks) lubaks teostada geograafilist analüüsi, nagu näiteks linnade vaheliste teede pikkuse arvutamist või kahe punkti vahelise lühima tee leidmist
- erinevate sümbolite salvestamine eraldi kihtidele lubaks analüüsida piirkonniti nendele sümbolitele vastavate objektide arvukust.

Kaartide digiteerimist tuleb ka ette juhtudel, kus olemasolevaid alasid, ehk salvestatud polügonide andmeid on vaja muuta ja vajalikud muutused on ette antud näiteks PDF- või pildifailina. Olemasolevate tööriistade abil saab need rasterkujul kaardid ja uued alad uuesti joonistada, kuid see on väga aeganõudev tegevus ja oleks kasulik, kui saaks seda protsessi automatiseerida.

## 1.1 Tööriistade kirjeldus

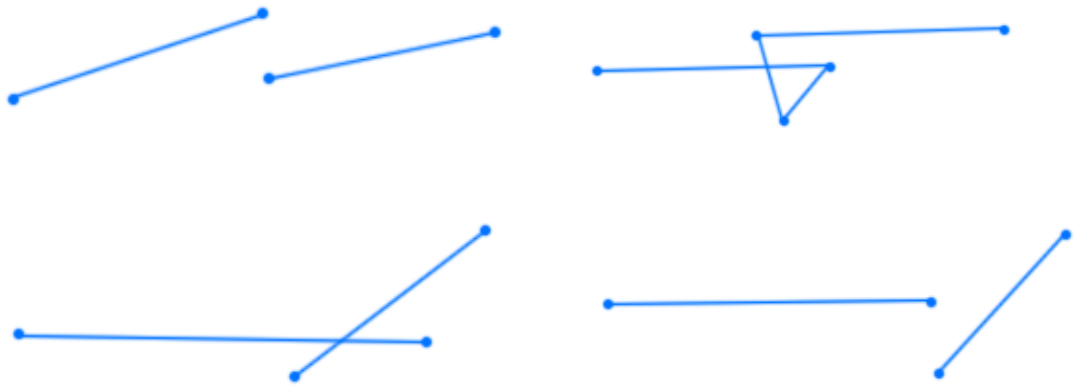
Kaartide digiteerimiseks on olemas tööriistasid ja funktsionaalsusi mõningates GIS arendus programmides. Mõneks näiteks nendest on QGIS[3] ja ArcMap[4] töölauarakendused.

[2] http://xgis.maaamet.ee/xGIS/XGis
[3] http://www.qgis.org/en/site/
[4] http://desktop.arcgis.com/en/arcmap/




Joonis 3 QGIS rakenduse põhivaade

Joonisel 3 on näha QGIS rakenduse põhivaadet ning sinise joonega on näidatud suunda, millel teede digiteerimise protsessi on alustatud. Antud juhul peab digiteerimistööga tegelev kasutaja märkima punkt-haaval olemasoleva tee ning salvestama erinevat liiki märked käsitsi eraldi kihtidele. Sümbolite märkimiseks kaardile tuleks kasutajal märkida kaardile punktid ning kihi seadete alt määrata selle kihi punktidele vastav graafika, ehk välimus. QGIS rakendusel on sisseehitatud funktsionaalsused, mis aitavad kasutajal digiteerimise töid teha. Üheks näiteks abistavatest funktsionaalsustest on *snapping* funktsionaalsus. *Snapping* funktsionaalsus juhib kasutaja kursori hüppeliselt lähima juba märgitud objektini, lihtsustades nii digiteerija tööd ja lubades luua veavabad andmekihid. Küll aga pole olemas erilist tuge sümbolite digiteerimisel juhul, kui töödeldakse näiteks ajaloolist kaarti, see töö kuulub täielikult käsitsi sisestamisele.



Joonis 4 käsitsi vektorandmete joonistamisel esinevad põhivead

Joonisel 4 on kujutatud neli erinevat tüüpviga[5], mis võivad kaardile käsitsi vektorandmete lisamisel, ehk digiteerimisel esineda. Kaardile polügonide ja vektorite joonistamisel peavad neid kirjeldavad piirjooned olema pidevad. Käsitsi ja ilma mingi abivahendita (nagu nimetatud *Snapping* funktsionaalsus) on vead väga lihtsad tekkima, eriti võttes arvesse, et vektoritel puudub laius ja väiksema mõõtkavaga vaadates veatuna paistvad andmed ei pruugi seda suurema mõõtkavaga vaates olla.

Joonisel 3 nähtaval kaardilõigul on näha palju teid, nende kõikide ära märkimine ülalkirjeldatud meetodil võtaks ühel kasutajal tunde, see aeg suureneks mitmekordselt, kui arvestada, et tegu on vaid väikse lõiguga Eesti kaardist, ning eksponentsiaalselt, kui arvestada ka Eesti kaarti suurematel mõõtkavadel.

Sarnaselt ülevalpool välja toodud näitega on võimalik kasutada ArcMap tarkvara. ArcMap on kõrgemal tasemel funktsionaalsuste koha pealt võrdne QGIS tarkvaraga, nimelt peab kasutaja defineerima eraldi kihid, määrama neile stiilid, ning hakkama käsitsi märkeid üle käima. ArcMapis on samuti olemas suunavad funktsionaalsused, nagu ülalpool nimetatud *snapping*.

## 1.2 Motivatsioon

Kuigi eelnevalt kirjeldasime peamiselt olemasolevate kaartide (näiteks paberilt sisse skaneeritud) digiteerimist, siis on suur osa ka ortofotode ja satelliidifotode töötlemisel. [1] Ortofotosid töödeldakse näiteks, et lisada neile vektorandmeid (näiteks ära märkida fotol olevad teed ja hooned) ja kohanimesid, või teha neist kaardid. Sellisteks ülesanneteks, ja ka eelnevalt kirjeldatud ülesanneteks, on olemas ettevõtted, mille teenuseks on kaartide digiteerimine ja töötajateks spetsialistid, kelle tööülesandeks ongi ülalpool kirjeldatud aeganõudev kaartide joonistamine. Suurt osa nende tööst saab kindlate meetoditega automatiseerida, või vähemalt abistavalt suunata, vähendades nii võimalikke vigasid ja kiirendades kogu protsessi.

Kuna tänapäeval on lihtsasti kättesaadavad erinevad kaardid ja ortofotod, on väga tõenäoline, et nende töötlemist vajab inimene, kellel ei pruugi olla otstarbekas palgata vastavaid spetsialiste. Näiteks on erinevatel kodulehtedel minimalistlike kaardirakenduste kujutamine populaarne trend. Inimene, kes vajab ühekordset minimaalset kaardiandmete töötlust, täpsemini digiteerimist, saaks kasu sellistest tööriistadest, mis nõuavad temalt võimalikult minimaalset sekkumist. Antud juhul keskendume peamiselt juhtudele, millest oleks kasu pigem GIS-teadlikule inimesele, kes oskab standardsete geoinfo-andmetega ringi käia, nagu näiteks seadistada vähemalt algset GIS veebirakendust, mis kuvab aluskaarti koos erinevate andmekihtidega.

[5] https://www.gislounge.com/digitizing-errors-in-gis/

# 2. Masinnägemine

Oluliseks osaks kaartidelt sümbolite tuvastamiseks on arvuti-nägemine või masinnägemine. Masinnägemine on valdkond, mis sisaldab paljusid erinevaid meetodeid ja meetodite gruppe, mis tegelevad piltidelt andmete kogumisega, nende töötlemisega, eesmärgil saada automaatselt piltidelt informatsiooni ja teha saadud informatsiooni alusel teatud otsuseid.

Käesolevas peatükis kirjeldame mõningaid meetodeid lähemalt, et anda ülevaadet ka nende alamsammudest, kuna mitmeid neist ei kasuta me nende peamisel otstarbel, vaid kasutame neid mingi teise algoritmi alamosana, või kasutame nende endi alamosasid.

Kasutades erinevaid masinnägemise tehnikaid on võimalik eristada piltidel kujutatud objekte. Masinnägemine on oluline osa ka näiteks robotite poolt nende ümbruse nägemisel, vahemaade hindamisel ja ümbruskonnas navigeerimisel, näoavastuse ja tuvastamise tarkvaradel, tootmisprotsessil toodete automatiseeritud ülevaatusel ja ka meditsiinis skaneeringute tulemusena saadud piltidel erinevate kehaosade, näiteks luude, esiletoomisel ja ka üldsusele tuntud piltide sarnasuse hindamisel (Näiteks *Google image search*[6]).

Masinnägemine koosneb erinevatest tehnikatest, millel on mitmeid erinevaid lähenemisi. Üheks tähtsaimaks operatsiooniks masinnägemise juures on servatuvastus. Servatuvastus aitab, kõige lihtsamal tasemel, tuvastada etteantud pildilt alad, kus mingi pildi omadus nagu hallid toonid muutuvad mingis suunas. Tuvastades piisavalt täpselt, kus asuvad kujutisel esinevate objektide servad, on võimalik eristada objektid pildil, ning kasutada seda informatsiooni edaspidiseks töötluseks.

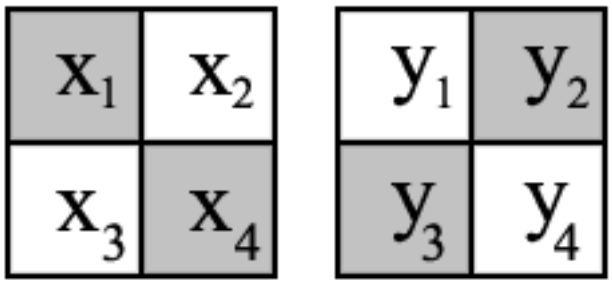
## 2.1 Servatuvastus

Vaatleme mõningaid servatuvastuse liike [3], uurime neid nende positiivsete ja negatiivsete külgede alusel ja arutleme selle üle, kuidas nad sobiksid kaartide töötlemiseks.

### Robert's cross servatuvastus

*Robert's cross* [3]servatuvastus on üks lihtsamaid servatuvastusalgoritme, mis uurib pilti 2x2 piksli suuruste ruutude haaval.

Teoorias vaadeldakse igat 2x2 ruutu kaks korda




Joonis 5 *Robert's cross* servatuvastuse võrdluse tabelid

[6] https://images.google.com/

Joonisel 5 on näha kahte tabelit, mis iseloomustavad piksleid, mida võrreldakse iga väljund pildi piksli väärtuse leidmiseks.

Iga 2x2 ruudu kohta arvutatakse kaks väärtust Gx ja Gy, mis tähistavad vastavalt gradiendi tugevust (määrates sisendpildi ülemisele, alumisele, paremale ja vasakule suunale vastavalt põhja, lõuna, ida ja lääne suunad) kagu-loode sihil ja kirde-edela sihil. Kasutades joonisel toodud väärtusi on Gx ja Gy väärtused leitavad järgnevalt:

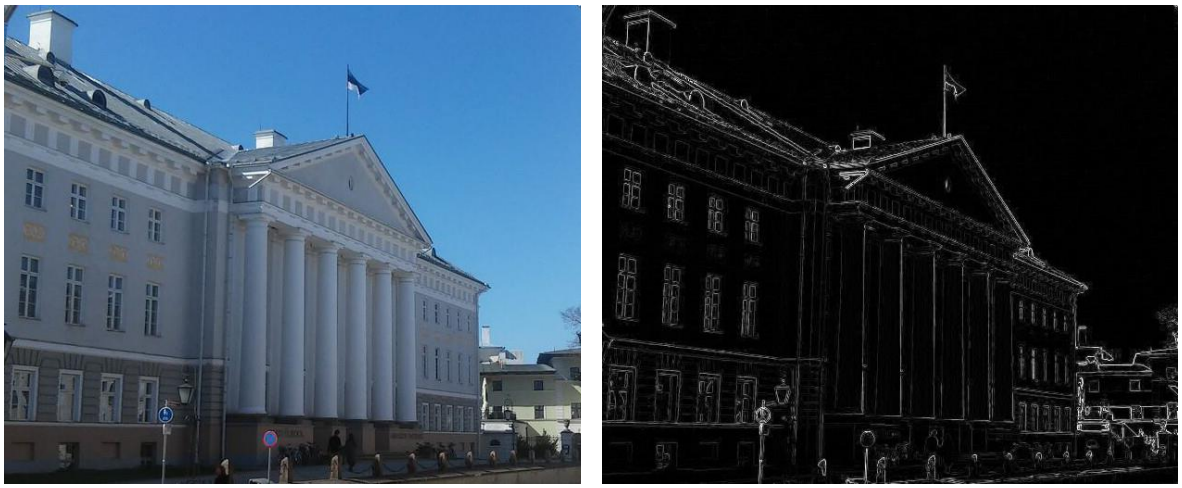
- $G_x = 1 * x_1 + 0 * x_2 + 0 * x_3 + (-1) * x_4 = x_1 - x_2$
- $G_y = 0 * y_1 + 1 * y_2 + (-1) * y_3 + 0 * y_4 = y_2 - y_3$

Gradiendi tugevuse väärtus igas punktis on võrdne nende kahe gradiendi geomeetrilise keskmisega $|G| = \sqrt{{G_x}^2 + {G_y}^2}$. Ligikaudne väärtus, kiiremaks arvutamiseks on leitav $|G| = |G_x| + |G_y|$.

Selgelt on antud meetod kõige efektiivsem 45-kraadiste nurkade tuvastamisel. Eraldi mõlema gradiendi väärtused lubavad ka leida tuvastatud serva nurga valemiga: $\theta = \arctan\frac{G_y}{G_x} - \frac{3\pi}{4}$

Tuvastatud serva nurga väärtus võib olla kasulik näiteks sirgjoonte tuvastamisel ja katkendlike joonte parandamisel, näiteks proovides ühendada katkendlikuks jäänud jooni, kindla kauguseni, järgmise tuvastatud servani.

Arvestades, et sisendiks on N = x*y suurune mustvalgele pildile vastav järjend, kus x on pildi laius ja y pildi kõrgus pikslites, on kirjeldatud meetodi algoritmil lineaarne keerukus, ehk O(N). See tähendab, et iga tulemuseks oleva pildi piksli kohta teostatakse üks nelja lähimat pikslit arvestav operatsioon.



Joonis 6 Tartu Ülikooli peahoone foto enne ja pärast *Robert's cross servatuvastust*

Joonisel 6 on näha näidet, kuidas servatuvastus töötab foto peal. Tulemuseks olevat fotot saab kasutada erinevatel pilditöötluse ja masinnägemisega seonduvatel protsessidel.

Rasterkaartidelt objektide eraldamisel servatuvastusmeetoditega ilmneb mitmeid probleeme. Kuna sümbolid kaartidel on enamasti väga väikesed ja muud märgistused, nagu polügonide piirjooned, on kitsad, siis jäävad tuvastatud objektid moondunud või katkendlikud.


Joonis 7 Kaardilõik enne ja pärast *Robert's cross* servatuvastust

Joonisel 7 on näha kuidas kasutades *Robert's Cross* servatuvastust, moodustavad töödeldud sümbolist enamuse tuvastatud servad, jättes sümbolist endast alles katkendlikud üksikud osad.

Joonis 8 Kaardilõik joonandmetega enne ja pärast *Robert's cross* servatuvastust

Joonisel 8 on näha kuidas servatuvastuse tulemusena on kaardil algselt olnud pidev joon jäänud katkendlik, ja läbi selle tuvastatav, kui mitu väikest sümbolit.

Kuna ülevalpool kujutatud näidetest on näha, kuidas *Robert's cross* servatuvastus ei ole väikeste objektide piiramisel väga edukas, ei ole ta iseseisvalt sobiv meetod sümbolite ja märgete eraldamiseks taustakaardilt.

### Sobel'i servatuvastus

Sobeli servatuvastus [3] on oma olemuselt väga sarnane *Robert's cross* servatuvastusega, kuid 2x2 ruutude asemel on vaatluse all pilt 3x3 piksli suuruste ruutudena.

| | | |
|---|---|---|
| $x_1$ | $x_2$ | $x_3$ |
| $x_4$ | $x_5$ | $x_6$ |
| $x_7$ | $x_8$ | $x_9$ |

| | | |
|---|---|---|
| $y_1$ | $y_2$ | $y_3$ |
| $y_4$ | $y_5$ | $y_6$ |
| $y_7$ | $y_8$ | $y_9$ |

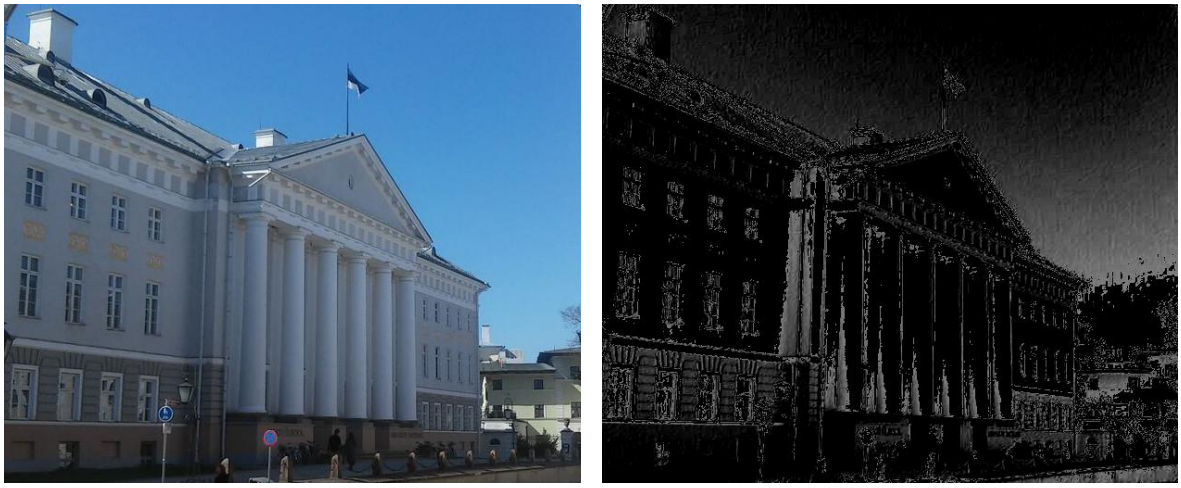
Joonis 9 Sobel'i servatuvastuse võrdluse tabelid

Kasutades joonisel 9 kasutatud tähistusi, leitakse väljundpildile serva gradiendi tugevused järgmiselt:

- $G_x = (-1) * x_1 + 0 * x_2 + 1 * x_3 + (-2) * x_4 + 0 * x_5 + 2 * x_6 + (-1) * x_7 + 0 * x_8 + 1 * x_9 = x_3 + 2x_6 + x_9 - x_1 - 2x_4 - x_7$
- $G_y = (-1) * y_1 + (-2) * y_2 + (-1) * y_3 + 0 * y_4 + 0 * y_5 + 0 * y_6 + 1 * y_7 + 2 * y_8 + 1 * y_9 = y_7 + 2y_8 + y_9 - y_1 - 2y_2 - y_3$

Valemid gradientide tugevuste hindamiseks on identsed *Robert's cross* servatuvastuse valemitega:

- Punkti gradient: $|G| = \sqrt{{G_x}^2 + {G_y}^2}$, ligikaudne väärtus: $|G| = |G_x| + |G_y|$
- Serva nurga valem: $\theta = \arctan\frac{G_y}{G_x} - \frac{3\pi}{4}$



Joonis 10 Tartu Ülikooli peahoone foto enne ja pärast Sobel'i servatuvastust

Joonisel 10 paistab Sobel'i servatuvastus olevat tundlik taustamürale, mis tähendab, et ilma eeltöötluseta pole fotodelt servatuvastus väga tõhus. Erinevalt *Robert's cross* servatuvastusest on Sobel'i servatuvastus peamiselt tundlik horisontaalselt ja vertikaalselt orienteeritud üleminekutele ehk servadele.

### Canny servatuvastus

Canny servatuvastus [4] koosneb mitmest sammust: esmalt töödeldakse sisendpilt *Gaussian* filtriga, mis sisuliselt hägustab pilti, et vähendada müra ning see jätta seeläbi tuvastamata servadena, teisena töödeldakse hägune pilt näiteks, kas *Robert's cross* või Sobel'i servatuvastusega, et hinnata iga piksli servaks olemise tõenäosust ja serva orientatsiooni. Serva orientatsioonid jagatakse nelja sihti (vertikaalne, horisontaalne ja mõlemad diagonaalid) ja selle alusel leitakse serva kõige intensiivsem osa, ehk kohalik maksimum, et jääks alles vaid ühe piksli laiusega serv. Viimase sammuna filtreeritakse alles jäänud servasid kahe lävendi järgi: T1 ja T2. Nende lävendite järgi hindamisel jäetakse alles servapunktid, mille intensiivsus ületab lävendit T1, või servapunkt, mis ületab lävendit T2 ja on ühenduses punktiga, mis ületab lävendit T1, ülejäänud servapunktid arvestatakse välja.

Kui arvestada, et algoritmi sisendiks on N = x*y pikkune mustvalget pilti iseloomustav järjend, kus x tähistab pildi laiust ja y pildi kõrgust pikslites, siis on Canny servatuvastuse algoritmi keerukuseks O(N log N). Kuna Canny servatuvastus sisaldab mitmeid samme, millest esimese, ehk *Gaussian* filtriga töötluse, keerukus on O(N), esialgse servatuvastuse algoritmi keerukus on O(N), servade orientatsiooni arvutamise keerukus on O(N), kohaliku maksimaalse leidmine tuvastatud servadel on O(N log N) keerukusega ja viimane samm on O(N) keerukusega.

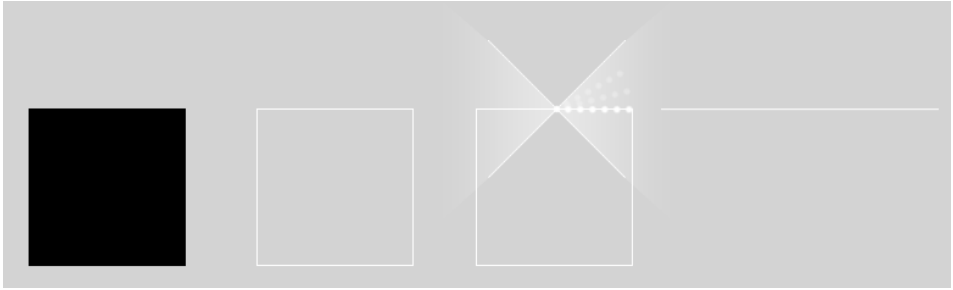
### Hough transform

*Hough transform*-i [5] protsessi käigus hinnatakse iga piksli tõenäosust olla osa mingist põhikujundist, nagu kindla sihiga sirged või kaared. Töötluse lõpus saab koostada kõige suurema tõenäosusega esinevatest põhikujunditest suuremad kujundid, võttes arvesse servad, mis sisendpildil reaalselt esinevad.

Selle protsessi esimeseks sammuks on teostada sisendpildil servatuvastus, kasutades näiteks ühte ülevalpool nimetatud meetodit, ning seejärel töödelda servasid nii, et jääks alles vaid kindlast lävendist kõrgema gradiendiväärtusega, moodustades nii binaarkujutis, nagu seda on tehtud Joonisel 8.

Järgnevalt uuritakse servadena tuvastatud punkte, ning kõiki punkte, mis jäävad mingile parameetrilisele põhikujundile, nagu näiteks sirgele, mis läbib seda punkti. Selliste punktide arv tõstab sellise põhikujundi esinemise tõenäosust, ja kui see ületab mingi piiri, loetakse see *Hough transform*-i väljundisse, ning neid saab arvesse võtta järgmises, ühtlasi viimases sammus.

Viimase sammuna võetakse *Hough transform*-i käigus tuvastatud põhikujundid, ning hinnatakse nende kujundite alamosade servaks-olemise tõenäosust, võrreldes neid sisendiks olnud servapildiga, eemaldades neilt need osad, mis ei paikne esialgselt tuvastatud servalõikude lähedal.



Joonis 11 *Hough transformi* sammude illustratsioon

Joonisel 11 on kujutatud *Hough transform* samme, kus töödeldakse sisendina üksikut ruutu: tuvastatakse ta servad, hinnatakse servapunktiga ühel joonel olevate punktide arv, ning tagastatakse tõenäoliseim põhikujund, ehk sirge, mis katab ruudu ülemist serva.

*Hough transform*-i üheks põhiliseks kasuks antud juhul on see, et see aitab parandada ja seeläbi tuvastada katkendlikuks jäänud tuvastatud servasid (lihtsad kujundid, nagu sirged jooned, kaared jne.)

Selle meetodi peamisteks negatiivseteks külgedeks on tema implementatsiooni keerukus, juhul kui soovitakse arvetada rohkem, kui ainult sirgeid jooni (nagu kaared ja ringid) ja operatiivne keerukus, kuna tegu ei ole lokaalse ega laisa algoritmiga, mille käigus tehakse jooksvalt otstuseid arvestades vaid ainsat pildi osa.

## 2.2 Pildi segmenteerimine

Sarnaselt servatuvastuse meetoditele uurime pildi segmenteerimist

Pildi segmenteerimine on eesmärgilt väga sarnane servatuvastusele, kuid servatuvastust võib pidada pildi segmenteerimise üheks osaks [6]. Segmenteerimise eesmärk on töödelda pilt arusaadavamateks osadeks, tuvastades pildil olevad objektid, klassifitseerides pildi igat pikslit mingi suurema objekti osaks.

Kõige lihtsamal tasemel võib pildi segmenteerimist kujutada ette kui klasterdamist[7] [7], kus iga pildi pikslit jagatakse tema omaduste (nagu näiteks asukoha või värvi väärtuse) järgi gruppidesse, mis tähistavad pildilt tuvastatud objekte. Käesoleval juhul ei sobi klasterdamismeetodid, millele peab ette andma tulemuseks olevate klastrite või võrreldavate objektide arvu (nagu *K-means* või *k-NN* klasterdamised), kuna on vähetõenäoline, et kasutajal on teada, mitu sümbolit on sisendiks oleval kaardil.

Pildi segmenteerimismeetodid, mis sarnanevad klasterdamisele võivad pakkuda lahendusi probleemidele, mis kerkivad servatuvastust kasutades liigselt väikestel objektidel, kuna .nende puhul ei pea määrama ära objekti kujutist ümbritsevaid piksleid ja seega ei teki eelnevalt kirjeldatud probleeme, nagu servatuvastuse korral.

Kuigi antud juhul ei ole see soovituslik, on üheks näiteks pildi segmenteerimiseks kasutada mingit servatuvastusmeetodit (paremini sobivad *Canny* ja *Hough* meetodid), jätta alles vaid servad, mis piiravad sisse mingi ala (ehk on ühenduses endaga), ning seejärel siduda eraldi objektideks sellesse alasse jäävad pikslid kasutades ühendatud komponentide sildistamise meetodeid.

[7] https://en.wikipedia.org/wiki/Cluster_analysis

Joonis 12 Ülikooli 11 hoone pilt enne ja pärast pildi segmenteerimist

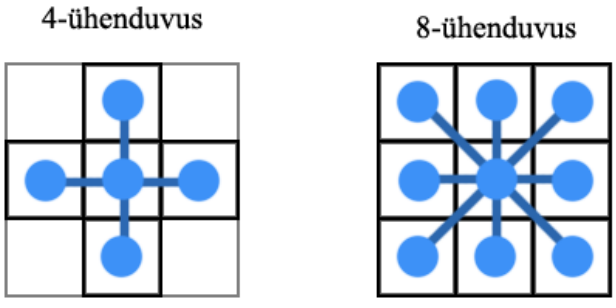
Joonisel 12 on kujutatud näide kirjeldatud segmenteerimismeetodi tulemusest, kus iga eraldi pildi segment on kujutatud erinevat värvi.

### Ühendatud komponentide sildistamine

Ühendatud komponentide sildistamise peaeesmärk on määrata ära, millised objekti osad, või alam-objektid, on omavahel ühenduses, ehk millised pikslid kuuluvad samale objektile [8]. Komponentide sildistamine aitab käsitleda sümboleid tervikutena. Kui sümbolite avastamiseks kasutatakse klasterdamisel põhinevaid meetodeid, on see samm segmenteerimisse integreerunud. Kui kasutatakse meetodeid, mis tuginevad pildi töötlusele (näiteks servatuvastusele või värvikontrastide suurendamisele), tuleb esiletoodud komponendid sidusateks objektideks märkida.

Et leida, millised komponendid on omavahel ühenduses, tuleb eelnevalt määrata ära naabriksolemise reeglid. Lihtsaimateks reegliteks võib lugeda neli-ühenduvuse ja kaheksa-ühenduvuse. Neli-ühenduvus tähendab seda, et naabrikteks loetakse elemente, mis on ühendatud vähemalt ühel suunal neljast: põhja, lõuna, ida või lääs.

Kaheksa-ühenduvuse põhjal loetakse naabriteks lisaks eelpoolnimetatud neljale suunale ka kirde, kagu, edela ja loode suundasid

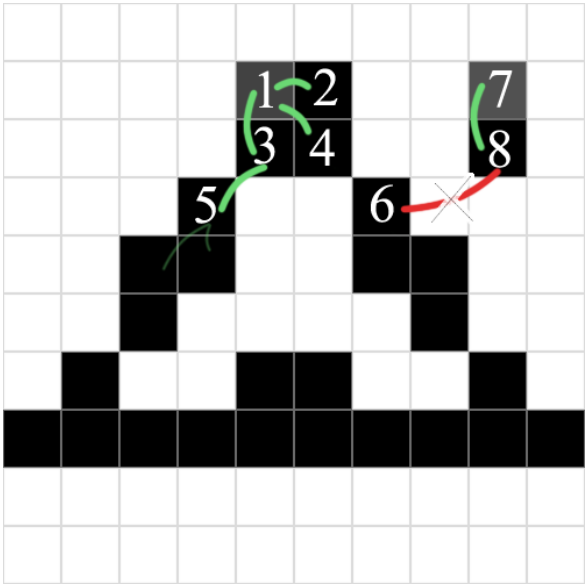



Joonis 13 Naabriksolemise reeglite illustratsioon

Kui on äramääratud naabriks olemise reeglid, nagu näiteks need, mida on kujutatud joonisel 13, saab piksleid sildistada kui mingile objektile kuuluvateks. Kui kasutada 4-ühenduvust või 8-ühenduvust töötlemata kujutisel, tuleb rakendada iga piksli hindamisel

meetodeid, mis aitaksid otsustada, kas piksel kuulub hetkel uuritavale objektile või mitte. Kui aga kasutada ühenduvuse reegleid juba töödeldud, nagu näiteks binaarkujutisel (kus pikslite väärtused tähistavad kas mingi objekti osaks olemist või mitte ühelegi kuulumist) tuleb lihtsalt tuvastada, millised pikslid on millistega sidusad.

Omavahel sidusate pikslite tuvastamine binaarkujutiselt taandub graafiteooria ülesandeks, kus kõik pikslid, mis on märgitud kui sümboli osad, on tipud ja iga kahe tipu vahel, mis on valitud ühenduvuse reegli alusel naabrid, on servad. Koheldes kogu binaarpilti, kui graafi, on üheks lihtsaimaks meetodiks siduvuse kontrollimiseks näiteks süviti graafi läbimine, ehk *depth-first search* (DFS).




Joonis 14 DFS-otsingu illustratsioon

Joonisel 14 on kujutatud 8-ühenduvuse reeglit kasutava DFS algoritmi samme pildil oleva sidusa komponendi sildistamisel. Binaarpildilt esimese sümbolile kuuluva piksli, tähistatud numbriga 1, leidmisel salvestatakse see uue sümboli üheks piksliks ja hakatakse uurima selle naabreid, antud juhul 8-ühenduvuse reegli alusel, kui uuritav piksel on positiivse väärtusega, ehk kuuluv mingile sümbolile, lisatakse see avastatud pikslite pinusse, kust neid pärast võetakse ja töödeldakse, nagu esimest pikslitki, lisades need samasse sümbolisse kuuluvaks, kuni pinu on tühi. Joonisel kujutatud juhul töödeldakse pikslid järjestuses 1,2,3,5,…,6,4 ja eraldi kõrval olevaid piksleid 7 ja 8, salvestades need kui teise sümbolisse kuuluvad.

## 2.3 Pildi skeletoniseerimine

Kirjeldame protsessi, mille tulemusena saame pildil tuvastatud objektidele vastavad minimaalsed graafstruktuurid, ehk objektide skeletoniseerimist [9].

Joonis 15 Kujundid ja neile vastavad skeletid

Joonisel 15 on näha kahte näidet sümbolitest (kujutatud mustana) ja nende skeletoniseerimise tulemustest ehk skelettidest, mis on kujutatud valgete joontena.

### Objekti peenestamine

Üheks skeletoniseerimise meetodiks on objekti iteratiivne peenestamine, mille käigus "kooritakse" objekti kihi haaval, kuni jääb alles minimaalne objekt, ehk objekti skelett. Peenestamismeetodeid võib jagada kahte gruppi: topoloogiat säilitavateks, ning topoloogiat mittesäilitavateks. Peenestamist loetakse topoloogiat säilitavaks, kui kõik esialgsed objektid jäävad alles, ning kõik sidusad objektid jäävad sidusaks, vastasel juhul on tegu topoloogiat mittesäilitava peenestamismeetodiga.

Kuigi objektide skelettide tagastamisel on peamiselt kasu topoloogiat säilitavast peenestamisest, on kasulik ka topoloogiat mittesäilitavast peenestamisest. Nimelt saab seda meetodit kasutada, et tuvastada puutuvate või kattuvate märgistuste kokkupuutepaigad, et neid nendelt kohtadelt eraldada ja kohelda kui eraldi märgistused või objektid.

Topoloogiat mittesäilitava peenestamise kasutamisel, ekslikult üheks loetud sümbolite eraldamisel, on ilmseid probleeme, nagu korrektselt sidusaks loetud objektide eraldamine teineteisest selle kitsamal osal. Nimetatud puuduse tõttu, ei ole peenestamist võimalik kasutada täielikult automatiseeritud protsessis, kuid võib olla abiks, kui kasutada seda soovitusena inimkasutajale.

## 2.4 Sümbolite klasterdamine

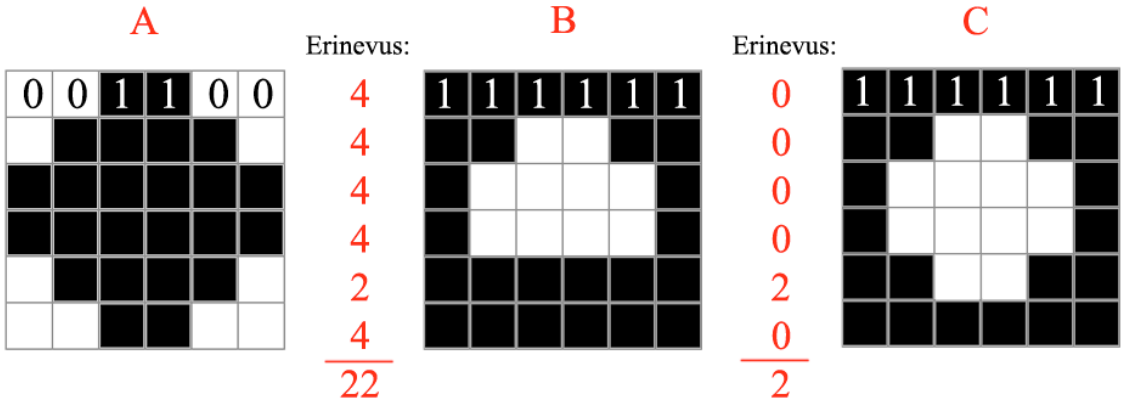
Kirjeldame võimalusi kuidas sümboleid saaks omavahel võrrelda, et neid jagada sarnasuse alusel gruppidesse.

Kuna suure tõenäosusega soovitakse identseid sümboleid samamoodi töödelda (näiteks kõik haiglaid tähistavad sümbolid paigutatakse uue veebikaardi ühele ja samale kihile) tuleb kasuks, kui pärast sümbolite tuvastamist jaotatakse nad sarnasuse alusel gruppidesse, ehk klastritesse.

### Sümbolite vahelise sarnasuse hindamine

Et sümboleid omavahel sarnasuse alusel klasterdada, peab olema võimalik nende omavahelist sarnasust hinnata arvulise väärtusega. Üheks otsekoheseimaks võimaluseks kahe sümboli omavahelise sarnasuse hindamiseks on teha seda sarnase meetodiga Levenshteini kauguse hindamisele [10]. Levenshteini kaugus on suurus, mis iseloomustab kahe jada, nagu näiteks sõne, omavahelist erinevust, võttes arvesse elementaaroperatsioonid selle jada muutmiseks, nagu: elemendi lisamine, elemendi kustutamine ja üksiku elemendi asendamine teisega. Levenshteini kauguse hindamise algoritmi kasutatake peamiselt

sõnede sarnasuse hindamiseks, näiteks otsingu algoritmides, kuid seda saab teisendada kahe binaarsümboli vahelise erinevuse hindamiseks, kujutades ette et sümbolid on mitmerealised sõned, mis koosnevad ainult kahest erinevast tähemärgist, nagu näiteks ”1” ja ”0”.




Joonis 16 Sümbolite vaheliste erinevuste arvutamine

Joonisel 16 on näha kolme sümboli A, B, C võrdlust sümbolitele kohandatud Levenshteini kauguse arvutamise algoritmiga.

Märgistame kõik sümboli A ruudud $a_1$, $a_2$, $a_3$,... $a_{36}$, sümboli B ruudud $b_1$, $b_2$, $b_3$,... $b_{36}$ ja sümboli C ruudud $c_1$, $c_2$, $c_3$,... $c_{36}$. Kahe ruudu, ehk piksli, a ja b vahelise kauguse $d(a;b)$ arvutame järgnevalt: $d(a;b) = \begin{cases} 1, kui\ a \neq b \\ 0, kui\ a = b \end{cases}$, seega iga kahe rea vahelise kauguse arvutame $\sum_{i=1}^{n} d(a_i;b_i)$, kus n on reas olevate ruutude arv. Sümbolite vahelise kauguse arvutamiseks leitakse $\sum_{1}^{r}\sum_{i=1}^{n} d(a_i;b_i)$, kus r on sümbolite ridade arv.

Kuna kaardilt avastatud sümbolid võivad olla erinevate suurustega, tuleb kahe sümboli omavahelisel võrdlemisel seda arvesse võtta. Kahe võrreldava sümboli suuruse erinevust saab lugeda kui suuremale sümbolile lisatud pikslite arvu, või juhul, kui lubatakse eri-suuruses samasuguseid sümboleid, võib väiksemat sümbolit skaleerida suuremaga ühisele suurusele, ning võrrelda pärast seda.

Suuremaid sümboleid ja muid objekte, nagu polügonide servasid, või terveid polügone saaks tõhusalt võrrelda omavahel kasutades osade tehnoloogiate nagu Oracle Spatial and Graph[8] või PostGIS[9] sisseehitatud funktsionaalsusi geomeetriliste kujundite võrdlemiseks. Oracle Spatial and Graph andmebaasi komponendi üheks sisseehitatud funktsiooniks on SDO_RELATE[10], mis võtab parameetriteks kaks SDO_GEOMETRY tüüpi geomeetrilist objekti ja VARCHAR2[11] tüüpi parameetri, mis määrab ära millist geomeetriate vahelist suhet kontrollitakse. SDO_RELATE funktsiooniga saab parameetriks määrata:

- TOUCH – juhul kui tahetakse kontrollida kahe geomeetria piirjoonte omavahelist kokkupuutumist ja sisemuste mittekokkupuutumist
- OVERLAPBDYDISJOINT – kui kontrollitakse kahe objekti piirjoonte ja sisemuste osalist ristumist

[8] https://www.oracle.com/database/spatial/index.html
[9] http://postgis.net/
[10] https://docs.oracle.com/cd/B19306_01/appdev.102/b14255/sdo_operat.htm#i78531
[11] https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/b_char.htm

- OVERLAPBDYINTERSECT – kui kontrollitakse piirjoonte ja sisemuste omavahelist ristumist
- EQUAL – kui tahetakse kontrollida geomeetriate 1:1 võrduvust
- INSIDE / CONTAINS – kui tahetakse kontrollida kas üks geomeetria paikneb teise sees, piirjooned võivad kattuda.
- COVEREDBY / COVERS – kui tahetakse kontrollida, kas üks geomeetria katab täielikult teise, sisemise geomeetria piirjoon täielikult teise geomeetria sees.
- ANYINTERACT – kui kontrollitakse, et objektid ei oleks ilma mingi suhteta
- ON – kui kontrollitakse, et ühe geomeetria piirjoon ja sisu paikneb teise objekti piirjoonel

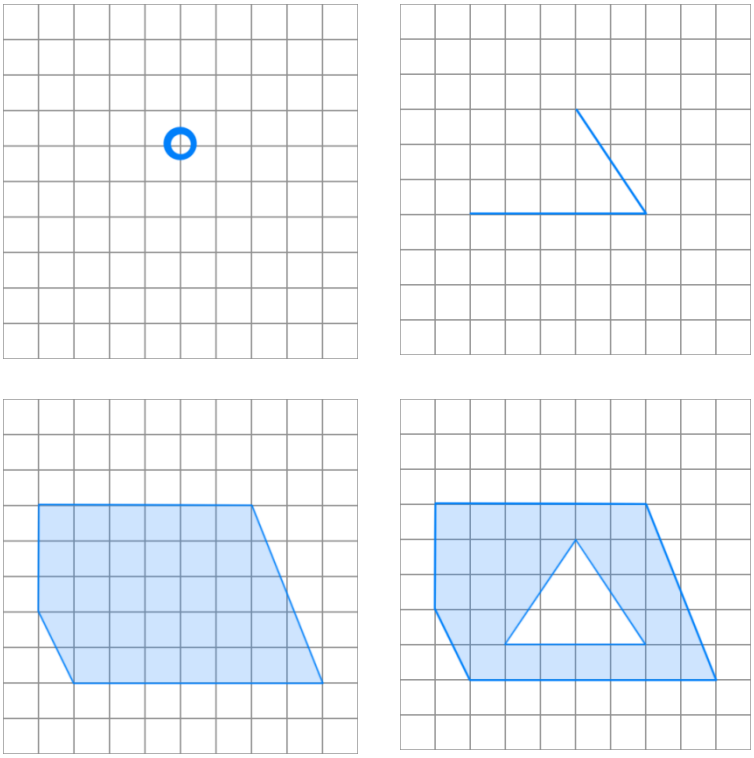
PostGIS-i kasutades saab kasutada kahe geomeetria võrdlemisel järgmisi kahte geomeetrilist objekti parameetriteks võtvaid funktsioone[12]:

- ST_Touches – (analoogne SDO_RELATE parameetriga TOUCH) kui kontrollitakse kahe geomeetria piirjoonte omavahelist kokkupuutumist ja sisemuste mittekokkupuutumist
- ST_Equals – (analoogne SDO_RELATE parameetriga EQUAL) kui kontrollitakse, kas kaks geomeetriat tähistavad sama geomeetriat
- ST_Within / ST_Contains – (analoogsed SDO_RELATE parameetritega INSIDE ja CONTAINS)
- ST_ContainsProperly – kui kontrollitakse, et üks geomeetria on teise sees, kuid ei kattu tema piirjoonega
- ST_CoveredBy / ST_Covers – (analoogsed SDO_RELATE parameetritega COVEREDBY ja COVERS)
- ST_Crosses – kui kontrollitakse, et geomeetriatel on mõned ühised punktid, kuid mitte kõik
- ST_Intersects – (analoogne SDO_RELATE parameetriga ANYINTERACT)

Selleks, et leitud sümboleid või kujundeid omavahel eelpoolnimetatud funktsionaalsustega võrrelda, tuleb need eelnevalt teisendada mingile kujule, mida need tehnoloogiad käsitleda oskavad, nagu näiteks *Well Known Text* (WKT)[13] formaat. WKT koosneb geomeetriliste primitiivide, ehk teisisõnu põhikujundite nimedest ja koordinaatide kogumikust, näiteks:

- POINT(50 60)
- LINESTRING(20 40, 70 40, 50 70)
- POLYGON(20 20, 90 20, 70 70, 10 70, 10 40, 20 20)
- POLYGON((20 20, 90 20, 70 70, 10 70, 10 40, 20 20), (30 30, 70 30, 50 60, 30 30))

[12] http://postgis.net/docs/reference.html#Spatial_Relationships_Measurements
[13] http://www.opengeospatial.org/standards/sfa



Joonis 17 WKT standardiga kirjeldatud põhikujundid

Joonisel 17 on näha nimekirjas väljatoodud WKT kujul objektidele vastavad kujundid

Lisaks on olemas geomeetrilised primitiivid, MultiPoint, MultiLineString ja MultiPolygon, mis on mitut joonisel 17 kujutatud kujundit sisaldavad kujundid.

Kui koordinaatidena kasutada avastatud sümbolite, ja muude märgistuste, koordinaate kaardil, on eelpoolnimetatud funktsioonide tulemused suure tõenäosusega alati negatiivsed, ning sarnasusi ei tuvastata, seega tuleks kõikide sümbolite koordinaadid reprojekteerida sama nullpunkti ümber, et nad, piltlikult öeldes, asuksid üksteise peal.

WKT formaati ei töödelda nimetatud Oracle ja PostGIS funktsioonide poolt otse, vaid neid tuleb eelnevalt parsida. WKT formaadist saab, Oracle Spatial and Graph kasutavates andmebaasides, SDO_UTIL[14] paki funktsionaalsusega FROM_WKTGEOMETRY parsida kujule, mida eelpoolnimetatud Oracle funktsioonid töödelda saaksid. PostGIS-i kasutavates PostgreSQL andmebaasides saab WKT parsimiseks kasutada funktsiooni ST_GeomFromText.

Kuna antud juhul on sisendiks kaardid, millel esinevad sümbolid on üldiselt väga väikesed, ei ole tõenäoline, et neid kirjeldatud andmebaasi funktsionaalsustega edukalt võrrelda saab. Samas ei ole ka tõenäoline et kahe polügoni vahelise erinevuse hindamine mingit kasu toob, kuna polügonidega tähistatakse kaartidel mingit mõju- või leviala ja nende alade kuju ei ole korduv omadus. Küll aga on kasu kaartidelt tuvastatud polügonide teisendamisest mingile tuntud kujule nagu GeoJSON, GML või WKT (nimetatud kujudest lähemalt peatükis 3.1), et neid sellel kujul salvestada ja vajadusel töödelda. Sarnaselt tuleks ka salvestada tuvastatud sümbolite rasterpiltidega kaasa ka selle asukohaandmed.

[14] https://docs.oracle.com/cd/B28359_01/appdev.111/b28400/sdo_util.htm

### Sümbolite jagamine gruppidesse

Kui on olemas võimalus hinnata iga kahe avastatud sümboli vahelist erinevust, on võimalik neid sümboleid ka klasterdada, näiteks K-nearest neighbour (k-NN) klasterdamisega [7], kus iga element, antud juhul sümbol, klasterdatakse etteantud arvu k lähima, antud juhul endaga sarnaseima, sümboliga klastrisse. Kuna üldjuhul ei ole teada, mitu samasugust sümbolit kaardil leiduda võib ja erinevaid sümbolite liike on erinev arv, on eelistatavad klasterdamismeetodid, mis ei nõua sisendina klastrite arvu, ega klastri suurust. Üheks võimaluseks on jätta klastrisse kuulumise nõudeks maksimaalse suuruse, mida üks sümbol võib teisest klastrisse kuuluvast erineda. Maksimaalset erinevust saab anda kaasa parameetrina, või lasta algoritmil endal see dünaamiliselt ära määrata. Üheks elementide vahelist kaugust hindavateks algoritmiks on DBSCAN [11]. DBSCAN võtab arvesse objektide vahelist kaugust ja objektide paiknevuse tihedust.

Lihtsaimal juhul on vaja arvesse võtta vaid objektide vahelist kaugust. Näiteks ühisesse klastrisse võidakse määrata kõik need avastatud sümbolid, mis ei erine teineteisest, ülevalpool kirjeldatud kohandatud Levenshteini meetodi järgi, rohkem kui 10 piksli võrra. Selline lihtne klasterdamise algoritm töötaks kõikide sümbolite kaugusmaatriksi põhjal. Kaugusmaatriks oleks $n * n$ suurune tabel, kus n on avastatud sümbolite arv, milles ruudul (x;y) olev väärtus iseloomustab sümbolite x ja y vahelist erinevust. Sellise klasterdamise teostamiseks läbitakse järgmisi samme:

- Hakatakse läbima avastatud sümbolite kaugusmaatriksit ridamisi $y_1 \dots y_n$, antud juhul tähistab $y_i$ positsioonil $i$ olevat väärtuste rida.
- Võrreldakse iga veeru väärtust $x_1 \dots x_n$ (antud juhul tähistab $x_i$ positsioonil $i$ olevat kaugusväärtust) parameetrina kaasa antud väärtusega, antud juhul 10-ga. Võrdlemata jäetakse väärtused $v(k; j)$ kus k tähistab veeru numbrit ja j tähistab rea numbrit, kui $k \geq j$. Need väärtused tähistavad erinevust iseendast või juba võrreldud väärtusi (kuna maatriks on sümmeetriline). Juhul kui väärtus $v(k; j) \leq 10$ loetakse positsioonil $k$ olev sümbol samasse klastrisse nagu positsioonil $j$ olev sümbol. Enne sama sümboli edasi võrdlemist kontrollitakse rekursiivselt, kas käesolevasse klastrisse lisandub sümboleid.

Kirjeldatud protsessi lõpuks on kõik avastatud objektid jagatud sarnasuse alusel klastritesse, ja see lihtsustab sümbolite töötlemist, kuna sama liiki sümboleid soovitakse tõenäoliselt töödelda ühiselt., näiteks sama liiki sümboleid tahetakse salvestada samale kihile, samasse andmebaasi tabelisse, neile tahetakse määrata samasugune stiil ehk välimus. Iga avastatud sümboli eraldi töötlemine nõuaks kasutajalt isegi rohkem aega kui praegused meetodid, kus iga sümboli kohale tuleb käsitsi joonistada punkt, ning tema kihile määrata mingi kindel stiil.

# 3. Geo-info veebirakendused

GIS tööriistade tootmine on tehtud lihtsaimaks just veebirakenduste kujul, kuna sellisel kujul esineb neid oluliselt rohkem, kui töölauarakendustena, viimased on peamiselt arendustarkvara gruppi kuuluvad tööriistad, mis lubavad täita geograafiliste andmete analüüsiülesandeid.

## 3.1 Open Geospatial Consortium

Kirjeldame mõningaid tähtsamaid standardeid, mis on seatud Open Geospatial Consortiumi (OGC)[15] poolt. Need standardid on loodud, et lubada eraldi arendatud tööriistadel (mis võivad ka erinevates programmeerimiskeeltes kirjutatud olla) paremini koos töötada. Erinevate standardite mõistmine annab ettekujutuse sellest, millistele standarditele peaks loodud tarkvara vastama, et ta oleks kasutatav koos mingi kindla olemasoleva tööriistaga.

Mõned tähtsamad OGC poolt seatud standardid on järgmised[16]:

- GML – *Geographic Markup Language* on sisuliselt XML geograafilise informatsiooni jaoks. GML-i kasutamine lubab kasutajal näiteks salvestada geograafilisi objekte koos selle kuju, koordinaatsüsteemiga, paigutussuhtega teiste objektide suhtes (näiteks kattuvused, sisalduvused ja puutumised), suunad, ajaga seotud andmed ja muud.
- GeoJSON – Sarnaselt GML standardiga on JSON standardil põhinev geograafilise informatsiooni salvestamiseks mõeldud standard.
- WFS – *Web Feature Service*. WFS on liides, mis lubab kasutajal teha päringuid ruumiliste objektide ja tunnuste järgi, nagu näiteks maakonnapiirid aluskaardil kuvamiseks.
- WFS-T – *Transactional Web Feature Service*. WFS-T On WFS edasiarendus, mis lubab lisaks päringute teostamisele ka luua, muuta, kustutada objekte ja tunnuseid.
- WMS – *Web Map Service.* WMS on sarnane WFS-ile, kuid lubab kasutajal teha päringuid kaardipiltide serveerimiseks koos kinnitatud infoga nende geograafilisest asukohast.
- WPS – *Web Processing Service.* WPS on liides, mis lubab kasutada geotöötlus teenuseid, näiteks võttes sisendiks kaks geomeetrilist objekti ehk polügoni ja tagastades nende ühendatud polügon või multipolügon.
- WCS – *Web Coverage Service*. WCS on liides informatsiooni jaoks millel on erinev väärtus kas erinevatel ajahetkedel või erinevates asukohtades, nagu näiteks sademetehulk.
- SLD – *Styled Layer Descriptor*. SLD on XML-il põhinev standard, mis lubab kirjeldada kaartide ja objektide omadusi, nagu näiteks erinevat liiki alad kuvatakse erinevat värvi.
- WMC – *Web Map Context*. WMC on standard, mis lubab salvestada kasutaja hetke vaadet,. Tänu WMC-le ei ole tarvilik mitmete kihtide infot igal käivitamisel uuesti mällu laadida. WMC lubab ka kasutajatel omavahel vaateid jagada.
- FES – *Filter Encoding Specification*. FES on XML-il põhinev standard, mis lubab valida ning filtreerida objekte ja tunnuseid mingi omaduse alusel.

Need standardid aitavad erinevatel tööriistadel koos töötada, näiteks Javas kirjutatud programm võib salvestada geograafilise objekti GML-ina, ning saata seda keeles C++ kir-

[15] http://www.opengeospatial.org/
[16] http://www.opengeospatial.org/docs/is

jutatud programmile, mis saaks sellega edaspidi tööd teha (näiteks lisada sellele 50-meetrine puhverala).

## 3.2 Tööriistade võrdlus

Selles alampeatükis võrdleme mõningaid populaarsemaid serveri- ja veebikliendi tööriistasid omavahel üleval pool kirjeldatud standardite ja nende toetamise seisukohalt.

### Servertööriistade võrdus

Serveritööriistadest valisime võrdluseks tööriistad nagu GeoServer[17], MapServer[18], QGIS Server ja ArcGIS Server.

Tabel 1 Servertööriistade vastamine OGC standarditele

| Nimi | WMS | WFS | WPS | WFS-T | WCS | WMC | SLD | FES |
|---|---|---|---|---|---|---|---|---|
| GeoServer | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| MapServer | ✔ | ✔ | O | O | ✔ | ✔ | ✔ | ✔ |
| QGIS Server | ✔ | ✔ | O | ✔ | ✔ | X | ✔ | X |
| ArcGIS Server | ✔ | ✔ | ✔ | ✔ | ✔ | O | ✔ | ✔ |
| Sümbol | Tähendus | | | | | | | |
| ✔ | Standardversiooni poolt toetatud | | | | | | | |
| O | Toetatud läbi (välise) lisa | | | | | | | |
| X | Pole toetatud | | | | | | | |

Tabelil 1 esitatud võrdluse põhjal paistab, et kõige rohkem nimetatud standarditest toetab üks populaarseimaid vabavaralisi serveritööriistasid GeoServer. Selle võrdluse tulemuse järgi võib pidada vähetõenäoliseks juhtusid, et kasutusel olevad tööriistad ei ole võimelised pakkuma WMS, WFS, WCS teenuseid, ega oska töödelda SLD'sid. Selle võrdluse tulemusena võime GIS programmide implementeerimisel arvestada WMS ja WFS võimekusega.

[17] http://geoserver.org/
[18] http://mapserver.org/

### Veebikliendi tööristade võrdus

Veebikliendi tööriistadest valisime võrdluseks tööriistad nagu OpenLayers[19], Leaflet[20], ArcGIS API[21], CartoDB[22] ja MapBox[23]

Tabel 2 Veebikliendi tööriistade vastamine OGC standarditele

| Nimi | Keel | WMS | WFS | GML | GeoJSON | GeoRSS | KML |
|---|---|---|---|---|---|---|---|
| **OpenLayers** | **JS** | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| **Leaflet** | **JS** | ✔ | **O** | **O** | ✔ | **O** | **O** |
| **ArcGIS API** | **JS** | ✔ | ✔ | **O** | **O** | ✔ | ✔ |
| **CartoDB** | **Ruby/JS** | ✔ | ✔ | ✔ | ✔ | **O** | ✔ |
| **MapBox** | **JS** | ✔ | **X** | **X** | ✔ | **X** | **X** |

| Sümbol | Tähendus |
|---|---|
| ✔ | **Standardversiooni poolt toetatud** |
| **O** | **Toetatud läbi (välise) lisa** |
| **X** | **Pole toetatud** |

Tabelil 2 esinevad eelnevalt mainimata standardid GeoRSS ja KML. GeoRSS on standard, mis iseloomustab RSS voogudesse lisatud geograafilist informatsiooni ja KML on standard Google Earth[24] jaoks väljatöötatud standard geograafilise info tähistamiseks. Tabelil esitatud võrdluse põhjal toetab kõige rohkem uuritavaid standardeid populaarne vabavaraline tööriist OpenLayers. Selle võrdluse tulemuste järgi on näha, et enamus veebikliendi tööriistadest oskavad kuvada WMS päringuga saadud kaarte, ning on vähe selliseid tööriistasid, mis ei saa hakkama WFS päringute töötlemisega.

[19] http://openlayers.org/
[20] http://leafletjs.com/
[21] https://developers.arcgis.com/javascript/
[22] https://cartodb.com/
[23] https://www.mapbox.com/
[24] https://www.google.com/earth/

# 4. Implementatsioon

Selles peatükis kirjeldame käesoleva töö käigus valminud programmi, mille ülesandeks on rasterkujul kaartidelt sümbolite avastamine ja nende sarnasuse alusel klassifitseerimine gruppidesse.

## 4.1 Programmi kirjeldus

Sisendina mõeldud kaartide töötlemiseks on vaja neile ligipääsu. Antud implementatsiooni jaoks on tarvilik ligipääs kaartidele kui tavalistele pildifailidele. Kui sisendina mõeldud kaardid on üles seatud GeoServerile ja sellele on seadistatud WMS päringud koos mingi sobiva failitüübiga nagu näiteks JPEG, PNG või GeoTIFF, saab kaarte pärida WMS päringuga:

```
Localhost:8080/geoserver/wms?
VERSION=1.1.1&
REQUEST=GetMap&
SERVICE=WMS&
LAYERS=massgis:GISDATA.HISTORIC_MAP &
BBOX=232325.38526025353,898705.3447384972,238934.49648710093,903749.1401484597&
SRS=EPSG:26986&
WIDTH=570&HEIGHT=435&
FORMAT=image/png&
```

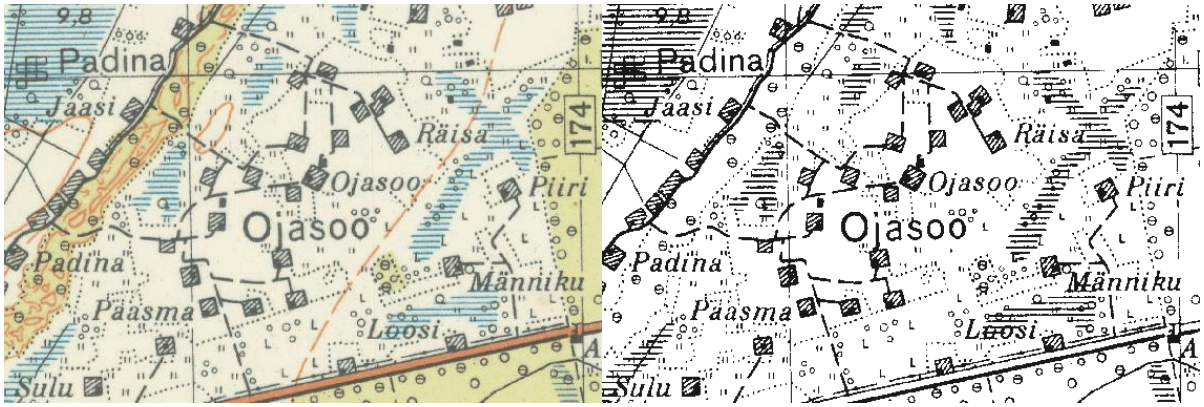
Toodud näide WMS päringu URL-ist, mis tagastab kaardilõigu PNG formaadis pildifailina, et näidata, mis parameetreid selline päring nõuab. Päringule on kaasa antud versioon, päringu tüüp, teenuse tüüp, kihtide nimed, millise ala sisest kaarti nõutakse (BBOX), projektsiooni informatsioon, pildi suurus ja laius pikslites ning pildi formaat.

WMS päringute tulemusena on pildifail, mida meie implementatsioon suudab töödelda, kuid käesoleva töö implementatsioon on seadistatud töötlema kaarte pildifailidena, mis asuvad arendusprojekti kaustas.

Digiteerimise programmi töö on jagatud mitmesse faasi ja iga faas töötab kui iseseisev programmi osa, ehk igale faasile antakse sisendina pildifail ning ta annab väljundiks töödeldud pildifaili, seda selleks, et iga osa oleks eraldi arendatav. Hoides programmi faasid eraldi, lubame lihtsamat vigade parandust, edasi arendust, programmi faaside järjekorra muutust.ja programmilõikude korduvkasutust. Eelnevad võimalused on tarvilikud just arendusprotsessiks, valmisolevate ja reaalseks tööks mõeldud rakenduste töövoog tuleks optimeerida nii, et ei läbitaks korduvalt põhjuseta samu andmeid, nagu seda tehakse väljundpildi kirjutamisel ja sisselugemisel iga kahe faasi vahel.

### Sisendpildi eeltöötlus

Esimeseks sammuks on sisendpildilt müra eemaldamine ja sümbolite taustakaardilt esiletoomine. Nimetatud protsesside tulemuseks selle faasi lõpuks on binaarpilt, ehk pilt kus taustaks loetud aluskaart on ühte värvi (valge), ning märgeteks ja sümboliteks loetud osad teist värvi (must).

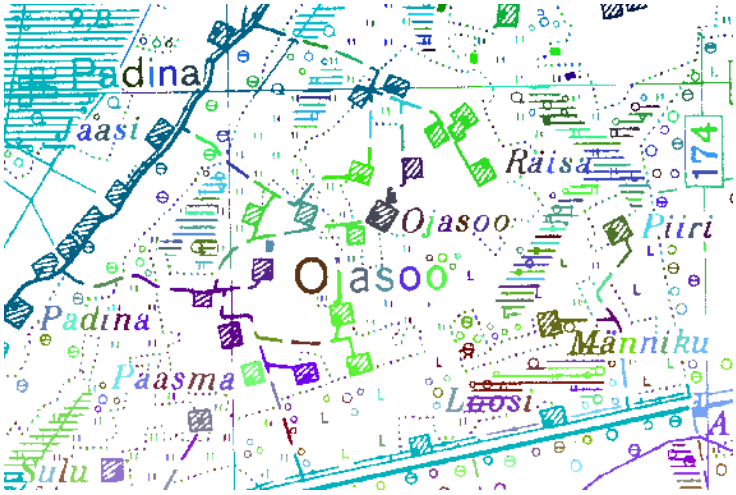



Joonis 18 Sisendiks olev kaardilõik ja sellele vastav binaarpilt

Joonisel 18 esitame näitena sisendiks oleva Maaameti kaardirakenduselt XGIS saadud kaardilõigu ja sellele vastava segmenteerimise tulemusena saadud binaarpildi.

### Sümbolite sildistamine

Kuna antud juhul on teostatud segmenteerimine eraldi pilditöötluse ja komponentide sildistamise faasidena, tuleb taustast eraldatud märked siduda ühtseteks komponentideks. Antud juhul on see lahendatud peatükis 2.2, komponentide sildistamise alampeatükis kirjeldatud meetodil, kasutades DFS otsingu meetodit ja kaheksa-ühenduvuse reeglit naabrite määramisel.

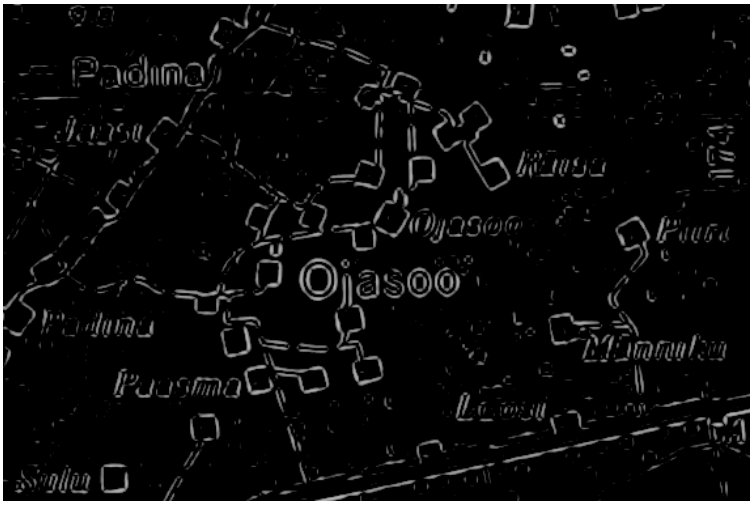



Joonis 19 Kaardilõik peale sümbolite sildistamist

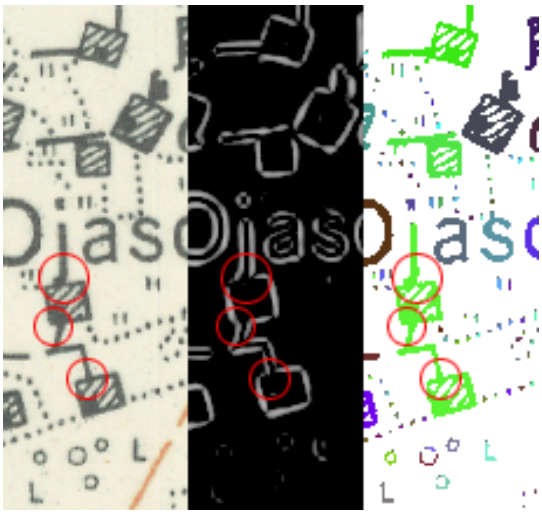
Joonisel 19 on kujutatud ülevalpool mainitud kaardilõik pärast esialgset eraldatud sümbolitele vastavate pikslite sildistamist ühtseteks objektideks.

### Komponentide järeltöötlus

Kuna eelmise sammu tulemusena loeti osad lähestikku paiknevad märked kokku üheks on vajalik lisasamm, mis eraldaks need eksikombel ühendatud märked teineteisest. Antud juhul on selleks kasutatud sisendpildi servaanalüüsi. Sisendpildi servatuvastusega töötlemisel saame eemaldada tuvastatud sümbolitelt ja märgetelt need servapikslid, mis muudavad nad sidusaks mingi teise sümboli või märkega.




Joonis 20 Kaardilõik pärast servatuvastust



Joonis 21 Järeltöötlusel avastatud vead

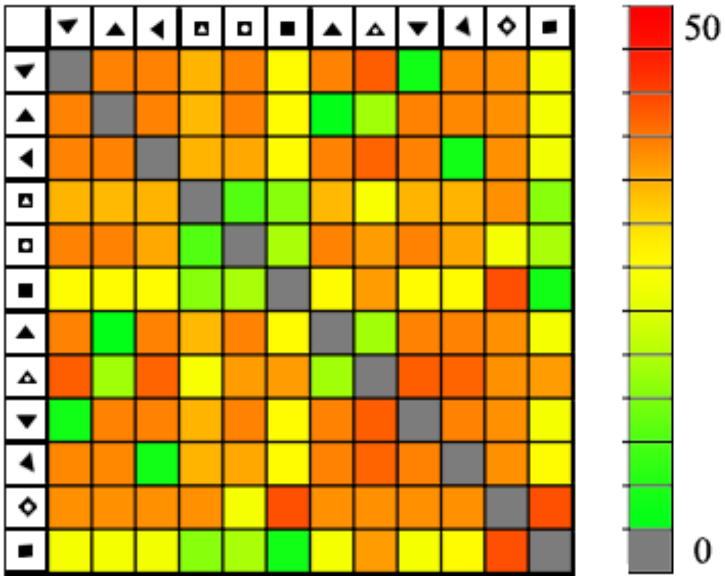
Joonisel 20 on kujutatud eelnevalt kirjeldatud kaardilõigule vastav servatuvastuse tulemus. Joonisel 21 on näha et sisendkaardil samat värvi olevad eriliiki märgistused jäävad eristamata, kuna *Robert's cross* servatuvastus nende vahel serva ei tuvasta. Antud juhul aitaks tõenäolislet *Hough transform* kasutamine.

### Sümbolite sarnasusmaatriksi moodustamine

Et sümboleid jaotada gruppideks, mis iseloomustaksid eraldi kihte, tuleb neid sarnasuse alusel klasterdada, aga et klasterdamine oleks võimalik, tuleb määrata ära iga objektipaari vaheline kaugus.


Joonis 22 Klasterdamise sisendiks olev kaardilõik

Selle protsessi näiteks kasutame lihtsamat, 12 sümboliga, kaardilõiku, mis on kujutatud joonisel 22, et oleks lihtsam jälgida, milliseks programm hindab nendevahelisi erinevusi, ning millised klastrid programm järgmises sammus moodustab.



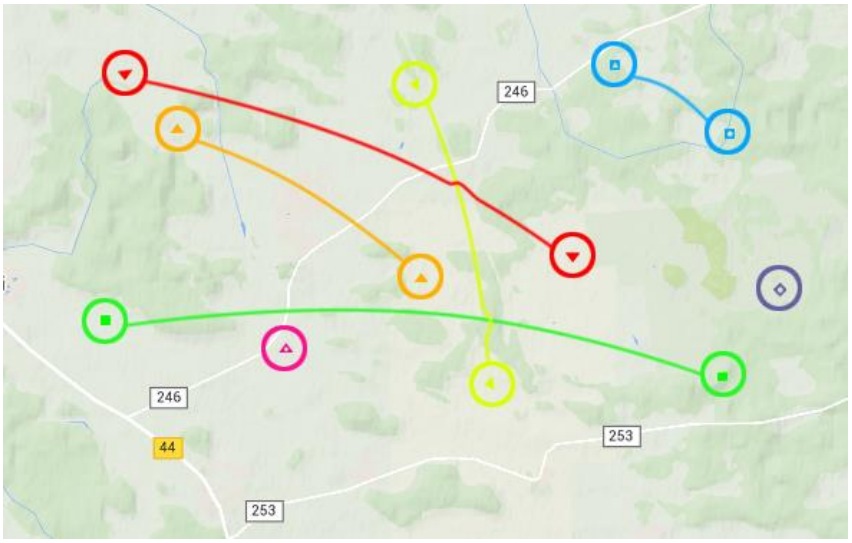

Joonis 23 Avastatud sümbolitevaheliste erinevuste soojuskaart

Joonisel 23 on kujutatud soojuskaarti, kus päises on kujutatud avastatud sümbolid, ning tabelis on kujutatud vastavate sümbolite vahelist erinevust iseloomustav värv. Sarnased sümbolid on tähistatud, vastavalt legendile, rohelise värviga, erinevad punasega ja erinevust iseendaga on kujutatud halli värviga.

### Sümbolite klasterdamine

Kuna nüüdseks on teada iga kahe sümboli vaheline erinevus, ehk kaugus, saame neid sarnasuse alusel klasterdada. Antud juhul on see teostatud kaugusepõhise klasterdamisega, kus läbitakse sümbolite loendit, lisades nendega ühisesse klastrisse eelnevalt määratud maksimaalsesse kaugusesse mahtuvad sümbolid ja samuti omakorda nende sobivad naabrid. Kirjeldatud klasterdamise algoritm võtab parameetriks maksimaalse kauguse kahe

ühisesse gruppi kuuluvate sümbolite vahel, nimetatud kaugus lubab müra tõttu moondunud sümboleid kokku liigitada. Antud juhul on see parameeter soovitatav määrata pigem väikseks, kuna suuremaid probleeme tekib juhtumist, kui erinevad sümbolid loetakse samadeks, kuna seda on hiljem raskem parandada, kui juhtu kus sama tüüpi sümbolid jaotatakse gruppide, ja hiljem kihtide, vahel laiali, kuna kihtide liitmine on üsna otsekohene tegevus, kuid ühest kihist mitmeks lahutamine nõuaks sarnast protsessi kogu kirjeldatud programmi tööga.



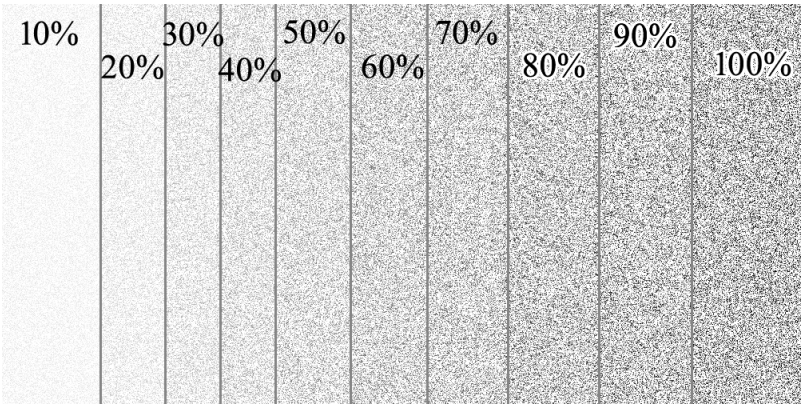

Joonis 24 Klasterdatud sümbolitega kaardilõik

Joonisel 24 toodud näites, kus on kujutatud programmi poolt tuvastatud klastrid, on näha ühte valepositiivset klastrit (märgitud helesinisega). Ühte klastrisse on loetud kaks ruutu, millest ühest on välja lõigatud ringikujuline, teise kolmnurgakujuline auk.

## 4.2 Automaattestid

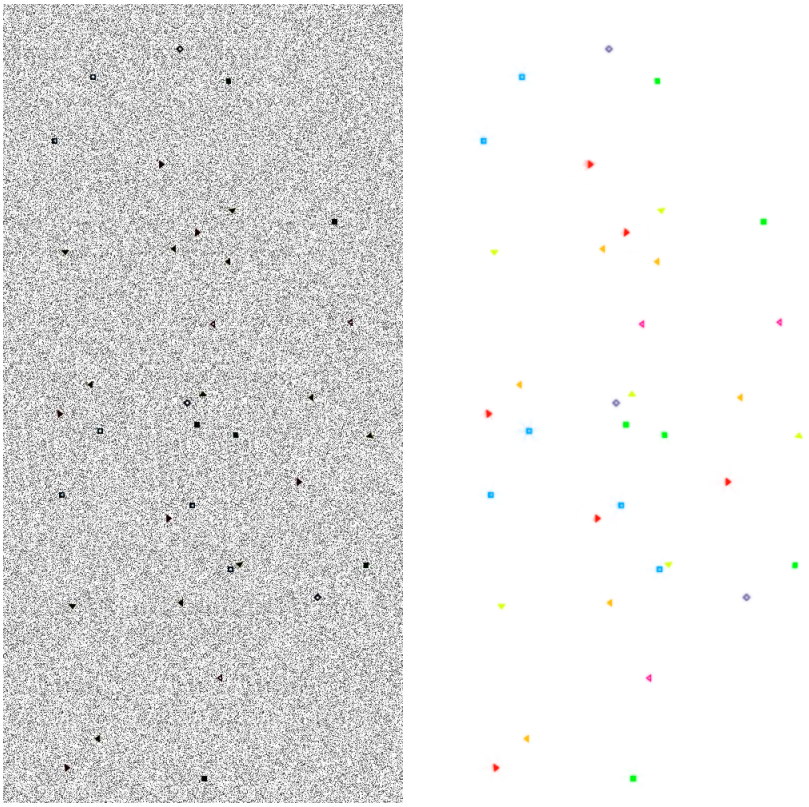
Kuna automatiseeritud tõhususe hindamiseks on vaja teada, mitu sümbolit või objekti sisendilt leida võimalik on, peab sisendkaardid ise genereerima. Päriseluliste kaartide töötlemisel ei ole mõistlikult tulemuse hindamine võimalik.

Käesoleva töö implementatsiooni testimiseks loodi programm, mis genereerib etteantud parameetrite alusel erineva tugevusega taustamüraga pildifailid ning paigutab neile etteantud sümboleid ettemääratud kordades. Genereeritud pildifaile töödeldi seejärel ülevalpool kirjeldatud digiteerimise programmiga ja tulemuseks olevate klastrite arvu ja suuruseid võrreldi genereeriva programmi sisenditega, et hinnata programmi täpsust.




Joonis 25 Näited genereeritava taustamüra tugevusest

Joonisel 25 on näha 10% kuni 100% tugevusega taustamüraga lõike pildifailidest, mida kasutatakse testides aluskaartidena.



Joonis 26 Testimiseks genereeritud kaart enne ja pärast töötlust

Joonisel 26 on näha ühele mürafailile paigutatud sümbolid, ning need samad sümbolid klasterdatult (klastrid on kujutatud eri-värvides).

Tabel 3 Automaattestide tulemused

| Taustamüra | Sisestatud sümbolite arv | Avastatud sümbolite arv | Sisestatud sümbolite gruppide arv | Tuvastatud sümbolite klastrite arv | Täpsus | |
|---|---|---|---|---|---|---|
| **10%** | **120** | **120** | **7** | **6** | **100%** | **85,7%** |
| **20%** | | **120** | | **6** | **100%** | **85,7%** |
| **30%** | | **120** | | **6** | **100%** | **85,7%** |
| **40%** | | **120** | | **10** | **100%** | **42,9%** |
| **50%** | | **135** | | **97** | **87,5%** | **0%** |
| **60%** | | **224** | | **224** | **13,3%** | **0%** |
| **70%** | | **201** | | **201** | **32,5%** | **0%** |
| **80%** | | **196** | | **196** | **36,7%** | **0%** |
| **90%** | | **175** | | **175** | **54,2%** | **0%** |
| **100%** | | **133** | | **133** | **89,2%** | **0%** |

Tabelil 3 on kujutatud iga erineva taustamüra taseme kohta sellelt tuvastatud sümbolite ja klastrite arv, iga müarataseme kohta on sisendite arvud samad, viimases veerus on kujutatud kõrvuti sümbolite avastamise täpsus ja nende klasterdamise täpsus. Programm tuvastab kuni 40-protsendilise taustamüraga testpiltidelt kõik sümbolid ja jaotab nad relatiivse eduga klastritesse. Alates 50-protsendilise tugevusega taustamüraga testpiltidest hakkab programm tuvastama taustamüra sümbolitena, kuna mürased pikslid seotakse liialt suurteks objektideks et neid välja filtreerida. Tabelis on näha 50-protsendilise ja suurema müraga kaartidel suuremat täpsust, aga see kirjeldab vaid olukorda, kus sümboliteks loetud taustamüra komponendid kasvavad suuremaks ja neid leitakse vähem. 50-protsendilise ja suurema müraga kaartidel kuvatakse klasterdamise täpsus nullina, kuna avastatud sümbolid ei iseloomusta enam sisendiks olnud sümboleid.

Tabeli andmete ja eelnevates peatükkides kirjeldatud tulemuste põhjal võime järeldada, et käesoleva töö käigus valminud prototüübi täpsus ei ole piisav iseseisvaks automaatseks rasterkujul kaartidelt sümbolite avastamiseks ning pole võimeline neid, ilma kasutaja sekkumiseta, korrektselt gruppidesse jagama.

Küll aga on valminud prototüüp sobiv alustalaks võimalikele edasiarendustele, mis abistaksid kasutajat kaardi digiteerimise protsessis, tehes avastatud sümbolite ja geomeetriliste märgete kohta soovitusi, mida kasutaja saaks kiirelt järgida või tühistada.

# 5. Edaspidine töö

Antud peatükis kirjeldame, kuhu suunas seni valmisoleva programmiga saab arenduses liikuda ja toome näiteid võimalikest rakendustest, mis aitaksid rasterkujul kaarte digiteerida.

Kuna tervete kaartide digiteerimine on väga erinevatest tegevustest koosnev tegevus, on seda tervenisti raske automatiseerida, küll aga saab automaatseks muuta osasid digiteerimist teostava spetsialisti tööst, näiteks tuvastades ja gruppeerides aluskaardilt kõik sümbolid, saab nende ükshaaval uuesti märkimise asemel nad eraldi kihina salvestada. Kuna kaartidel leidub ka joonandmeid ja geomeetriliste kujunditena märgitud alad, oleks kasulik teostada rakendus, mis lubaks nende kujundite järele joonistamise asemel neid valida ja määrata mingile kihile

Järgnevas kahes alampeatükis kirjeldame kahte rakendust, millest esimene oleks kasutusel sümbolite digiteerimiseks ning teine geomeetriate digiteerimiseks. Kirjeldatud rakendused saaksid olla realiseeritud ka ühe rakendusena juhul, kui implementeerida funktsionaalsus, mis oskaks eristada punktandmeid, ehk sümboleid ja geomeetria-andmeid, ehk jooni ja polügone. Mõlemad rakendused vajavad edukaks tööks spetsialiseeritud *Hough transform* implementatsiooni. *Hough transformi* oleks vaja nii sümbolite digiteerimisel, nende eraldamiseks teineteisest ja geomeetriate digiteerimisel kujundite tuvastamiseks.

## 5.1 Sümbolite digiteerimine

Üheks rakenduseks võiks olla GeoServerilt aluskaarte päriv veebirakendus, mis kuvab neid kaarte OpenLayers rakendusena. See rakendus võiks lubada navigeerida salvestatud kaarte asukohale, mida soovitakse digiteerida. Töötluse tulemusena tuleks kasutajale pakkuda avastatud sümbolid võimalike kihtide kaupa, ning lubada kasutajal igat sümbolit kihtidelt eemaldada, lisada ning vajadusel pooleks lõigata.

Digiteeritud andmete salvestamisel tehtakse kirjed, näiteks Oracle andmebaasi, mis kasutab *Oracle Spatial and Graph* andmebaasi lisa, kus väljadena oleks vähemalt sümboli identifikaator, kihi identifikaator ja tema asukoht SDO_GEOMETRY punkti kujul ja projektsiooni info (asukoht ja projektsiooniinfo oleks kättesaadav aluskaardiga seotud andmetest eeldusel, et kaart on geodeetiliste viidetega).

Rakendus salvestaks sümboli graafika sisendkaardilt pildifailina, ning genereeriks vastava SLD, mis määraks ära, kuidas antud kihi punktid kaardil kuvatud peavad olema.

```
<FeatureTypeStyle>
     <Rule>
       <PointSymbolizer>
         <Graphic>
           <ExternalGraphic>
             <OnlineResource
               xlink:type="simple"
               xlink:href="symbol1.png" />
             <Format>image/png</Format>
           </ExternalGraphic>
           <Size>10</Size>
         </Graphic>
       </PointSymbolizer>
     </Rule>
   </FeatureTypeStyle>
```

Näide SLD’st mis määrab kihi välimuseks symbol.1.png failinimega pildifaili.

## 5.2 Geomeetriate digiteerimine

Esialgu eraldi rakendus võiks olla mõeldud erinevate geomeetriatena kujutatud märgete, nagu polügonid ja jooned, digiteerimiseks. Üldjoontes töötaks rakendus sarnaselt nagu eelpool kirjeldatud sümbolite digiteerimise rakendus, kasutades andmete salvestamiseks andmebaasi nagu näiteks Oracle, aluskaartide serveerimiseks, WMS päringutena, GeoServerit ja nende kuvamiseks OpenLayers-it.

Kui praegu digiteeritakse selliseid andmeid neid järele joonistades, eraldaks rakendus neid taustast iseseisvalt määrates nende piiridele minimaalselt vajaliku arvu punkte. Rakendus lubaks kasutajal tuvastatud geomeetriaid kihtidele määrata, ning neid salvestada. Tuvastatud geomeetriad salvestataks andmebaasi koos oma kuju iseloomustavate punktidega, projektsiooni informatsiooniga ja nii geomeetria, kui ka kihi identifikaatoriga.

Sarnasel loogikal põhinevat rakendust saaks ka kasutada ortofotode digiteerimisel kaartideks, kuid selline rakendus nõuaks palju keerulisemaid algoritme, mis suudaks sisendfotodelt tuvastada erinevaid alasid ja objekte.

## 5.3 Teksti tundmine

Kirjeldame ka lühidalt sümbolitena tuvastatud tähemärkide töötlemist tekstiks.

Kuna kaartidel on mitmed märked, peamiselt kohanimed, märgitud tekstina, on mõistlik püüda käsitleda neid tekstina. Kuna erineva fondi, suuruse ja orientatsiooniga tähemärkidest koosnevaid märkeid on kõige lihtsam võrrelda ja seeläbi klasterdada just tekstina.

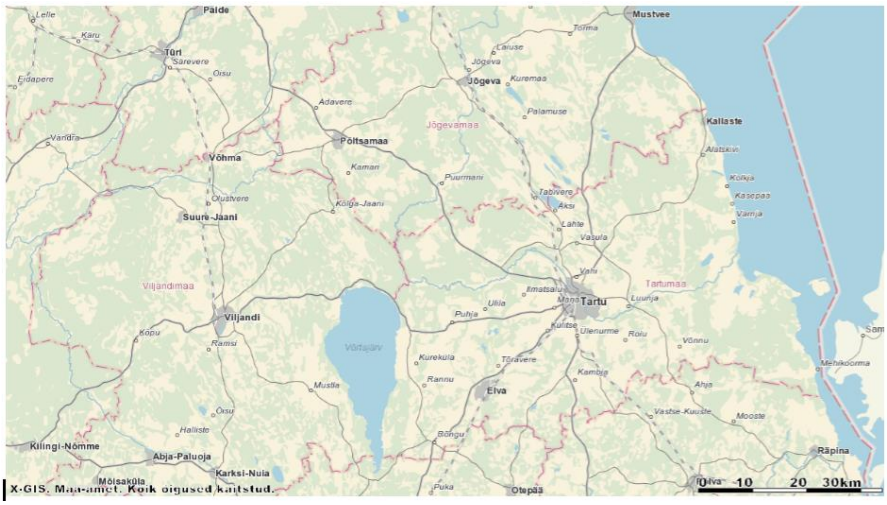
### Optiline trükimärgi tundmine

Optiline trükimärgi tundmine on meetod pildilt leitud tähemärkide muundamiseks tekstiks. Seda meetodit kasutatakse peamiselt skaneeritud tekstidokumentide muundamiseks töödeldavateks dokumentideks. Mõningaks näiteks tööriistadest, mis just nimetatud eesmärgil loodud on või sellist funktsionaalsust omavad, on järgnevad:

- Google Docs[25],
- ABBYY[26],
- Adobe Acrobat[27],
- ScanSnap[28]

[25] https://docs.google.com/
[26] https://www.abbyy.com/
[27] https://acrobat.adobe.com/us/en/
[28] http://scansnap.fujitsu.com/

Tabel 4 Google Docs poolt töödeldud kaart




**Paide & - - --- . Torona y_- / --> Türi o o ơ \ - ärevere Jēgëva Gisu o - \ \, o - ᔕ Jõgeva „Kuremaa \ -ᔕ Adavere \ o o s Jõgevamaa s \ o ᔑ s ᖘ g - \ o o Põltsamaa - ᔑ a Võhma - \ o --- „Kamari \ - - setSivere Kölga-Jaani w - - Surjan o o o - - o o Viljandimaa o Viljandi - Vā - Võrtsjärv „Võnnu - ox * o --- ᔕ. - | of- Ahja -**

**— Kisingi-Nõmme o ■ --- - o o Mõisaküla o**

*** - X-GIS. Maa-ārïěř. Köik öigused kāïtstud.--**

Tabelil 4 on toodud näide Google Docs võimekusest leida rasterkaardilt tekstikujul olevad marked

Kuna Google Docs pildi parsija on mõeldud peamiselt trükitud dokumentide töötlemiseks, jääb ta hätta kaartide ja ka näiteks mitme erineva orientatsiooniga tekstilõikude töötlemisega, küll aga on ta võimekas töötlema tekstilõike, nagu linnade nimed, kui nad on ette antud ükshaaval. Eelnev näitab, et trükimärgi tundmise meetodid on piisavalt võimekad, et neid kasutada rasterkujul kaartide automatiseeritud digiteerimisel.

Põhiline meetod optilise trükimärgi tundmise taga on osaliselt sarnane sümbolite tuvastamisega, kus alguses eraldatakse sisendpildilt tähemärkidele vastavad pikslid, neist moodustatakse ühtsed sümbolid, ning neid võrreldakse musternäidistega. Pärast esialgset mustritega võrdlust on iga sümboli jaoks olemas iga tähemärgiga seonduvad tõenäosused (ehk kui sarnane ta mingile tähemärgile on). Sellele järgneb enamasti sõnaraamatupõhine analüüs, kus tuvastatud sümbolid viiakse kooskõlla inimkeeles esinevate sõnadega. Kaartidelt tuvastatud teksti järeltöötlusel oleks sõnaramaatu sisuks näiteks tuntud kohanimed.

## Kokkuvõte

Käesoleva töö eesmargiks oli implementeerida programm, mis on võimeline avastama rasterkujul kaartidel kujutatud sümboleid, ning neid jagama sarnasuse alusel gruppidesse. Töö käigus uuriti ning tutvustati erinevaid masinnägemise meetodeid, millest mõningad implementeeriti ja testiti rasterkujul kaartidel. Töö käigus valmis esialgne versioon programmist, mis on võimeline taustakaardilt eraldama sellele tehtud märgistusi ja sümboleid grupeerima sarnasuse alusel. Tööle püstitatud eesmärgid täideti. Valminud programm on alustalaks võimalikele rakendustele, mis aitavad GIS spetsialistidel digiteerida rasterkujul kaarte automatiseerides selle mõningaid alamülesandeid.

Magistritöö esimeses osas kirjeldati käsitletava valdkonna tausta, toodi lühike ülevaade sellest, kuidas tavapäraselt geoinfosüsteemide rakendused töötavad, millised probleemid võivad nende rakenduste tavapärast tööd takistada ja kuidas neid senimaani lahendatud on.

Teises osas kirjeldati mõningaid masinnägemise meetodeid, mis on tähtsaks osaks antud ülesande lahendamisel. Kirjeldati servatuvastust, pildi segmenteerimist, pildi skeletoniseerimist ning avastatud sümbolite klasterdamist. Testimisest selgus, et väikeste sümbolite avastamiseks on soovituslik kasutada klasterdamismeetoditel põhinevaid pildi segmenteerimise meetodeid, ning et nii sümbolite kui ka geomeetriate tuvastamisel on tähtsaks osaks *Hough transform* meetodit kaasav servatuvastus.

Magistritöö kolmandas osas kirjeldasime geoinfosüsteeme, täpsemalt geoinfo veebirakendusi ja neile kehtestatud standardeid, mida on heaks tavaks järgida, et oleks võimalik, koostöös loodud tarkvaraga, kasutada erinevaid olemasolevaid tarkvarasid, mis geograafilise informatsiooni töötlemiseks loodud on. Võrdlesime erinevaid serveri ja veebirakenduse tööriistasid kirjeldatud standarditele vastavuse alusel.

Neljandas osas kirjeldasime programmeerimiskeeles Java kirjutatud programmi, mis võtab sisendiks rasterkujul kaardifaili, ning avastab sellelt märked, eraldab need aluskaardilt, ning jagab sarnasuse alusel gruppidesse. Kirjeldasime kuidas antud programmi testisime, ning esitasime ka testide tulemused. Testide tulemustest järeldasime, et implementeeritud programm ei ole praegusel kujul võimeline piisavalt usaldusväärselt iseseisvalt rasterkujul kaarte töötlema, kuid on sobilik alustalaks programmile, mis on mõeldud inimkasutaja abistamiseks kaartide digiteerimisel. Kirjeldasime avastatud puudusi, ning pakkusime võimalikke parandusi.

Kirjeldasime ka võimalikke edasiarendusi ja tõime näiteks kaks võimalikku rakendust, millest üks oleks sobilik abistama geoinfosüsteemide spetsialiste kaartidelt sümbolite digiteerimisega ning teine abistama geomeetriliste andmete digiteerimisel. Kirjeldasime võimalust tuvastada avastatud sümbolite hulgast tähemärke, kirjeldasime lühidalt olemasolevaid tööriistu ning seletasime, kuidas sellest digiteerimisel kasu oleks.

Üldiselt leidsime, et uuritud meetodid on kaartide digiteerimise valdkonnas väga paljulubavad ning võiksid aidata senist kaartide digiteerimise töövoogu, selle mõningaid alamsamme automatiseerides, kiirendada.

## Kasutatud materjalid


[1] D. J. Maguire, „ "An overview and definition of GIS.",“ *Geographical information systems: Principles and applications,* nr 1, pp. 9-20, 1991.

[2] R. L. J. E. M. a. W. J. R. Lawrence, „"An automated method for digitizing color thematic maps.",“ *Photogrammetric engineering and remote sensing,* nr 62, pp. 1245-1248., 1996.

[3] R. H. A. Maini, „"Study and comparison of various image edge detection techniques.",“ *International journal of image processing,* nr 3.1, pp. 1-11, 2009.

[4] J. Canny, „ "A computational approach to edge detection.",“ *Pattern Analysis and Machine Intelligence,* pp. 679-698, 1986.

[5] D. H. Ballard, „"Generalizing the Hough transform to detect arbitrary shapes.",“ *Pattern recognition,* nr 13.2, pp. 111-122, 1981.

[6] J. J. M. Shi, „"Normalized cuts and image segmentation.",“ *Pattern Analysis and Machine Intelligence,* nr 22.8, pp. 888-905, 2000.

[7] R. I. I. D. W. Xu, „"Survey of Clustering Algorithms.",“ *IEEE TRANSACTIONS ON NEURAL NETWORKS,* nr 16.3, p. 645, 2005.

[8] H. M. T. Samet, „"Efficient component labeling of images of arbitrary dimension represented by linear bintrees.",“ *Pattern Analysis and Machine Intelligence,* nr 10.4, pp. 579-586, 1988.

[9] R. e. a. Kimmel, „ "Skeletonization via distance maps and level sets.",“ *Computer vision and image understanding,* nr 62.3, pp. 382-391, 1995.

[10] P. E. e. Black, „ "Levenshtein distance",“ *Dictionary of Algorithms and Data Structures,* 2008.

[11] M. Ester, H.-P. Kriegel, J. Sander ja X. Xu, „A density-based algorithm for discovering clusters in large spatial databases with noise,“ *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,* nr (KDD-96), p. 226–231, 1996.

# Lisad

## I. Terminid

| | |
|---|---|
| Rasterandmed – andmed, mis on kujutatavad maatriksistruktuurina, mille elemendid on pikslid | Raster data – data that is represented as a matrix structure, the elements of which are pixels |
| Vektorandmed – andmed, mis on kujutatud kujunditena, mida kirjeldatakse servapunktidena | Vector data – data that is represented as shapes, that are determined by it's edge points |
| Digiteerimine – rasterkujul kaardi või ortofoto töötlemine geograafiliseks informatsiooniks | Digitization – processing of an raster map or orthophoto into geographic information |
| GIS(Geoinfosüsteem) – geograafilisi andmeid töötlev infosüsteem | GIS(Geographic information system) – information system that processes geographic data |
| Ortofoto – on õhust tehtud maapinna pilt, mis on parandatud vastavalt geograafilistele andmetele | Orthophoto – an aerial image of the ground that is processed to represent geographic areas |
| Masinnägemine – teadusharu, mis tegeleb piltide töötlemisega arvutile arusaadavateks andmeteks | Machine vision – a field of science that deals in processing images into data, that can be understood by computers |
| Skelett – Kujundi minimaalne graafstruktuur, mis on saadud skeletoniseerimise käigus | Skeleton – a minimal graph structure which is the result of the skeletonization process |

## II. Autori GIT repositoorium

Kirjeldatud programmi prototüübi arendus on hoiul git repositooriumis aadressil: https://bitbucket.org/AarneaA/mapparser.

Repositoorium sisaldab kirjeldatud programmi Java koodi, mitmeid sisend- ja väljund-pilte ja mõningaid alternatiivseid proovitud lähenemisi.

## III. Litsents

**Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks**

Mina, **Aramaa, Aarne**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
**Rasterkujul kaartidelt sümbolite tuvastamine**,
(*lõputöö pealkiri*)

mille juhendaja on Vambola Leping,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **20.05.2016**