TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Teadmussüsteemide õppetool

Algoritmi ja ekspertsüsteemi arendus piltide tekstituvastuseks

Magistritöö

Üliõpilane: Rauno Rüga
Üliõpilaskood: 121865
Juhendaja: prof Jaak Tepandi

Tallinn
2014

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.

---------- ----------

(*kuupäev*) (*allkiri*)

Rauno Rüga

Algoritmi ja ekspertsüsteemi arendus piltide tekstituvastuseks

Magistritöö

Annotatsioon

Käesoleva töö eesmärgiks on välja töötada lahendus, mida saaks kasutada rakendustes piltide tekstituvastuseks piiratud tingimustel, kui pildid on struktuurilt sarnased ja esinevad tähemärgid on teada ning need on alati samas formaadis. Lahenduse leidmisel on olulisteks argumentideks kiirus, täpsus, väike maht ning sobivus kasutuseks nutitelefonide rakendustes.

Töös kirjeldatakse põhjalikult probleemi tausta ning tulemuse potentsiaalset kasutust. Võrreldakse erinevaid alternatiivseid lahendusi, mis on nende negatiivsed ja positiivsed küljed, ning miks on siiski vaja arendada uus lahendus. Tuuakse välja töö käigus esinenud probleemid ning kirjeldatakse komponentide arendust, mis lõpuks algsele probleemile lahendust pakuvad.

Magistritöö olulisi tulemusi on kolm. Esiteks uus algoritm, mis võimaldab tähemärkide analüüsimist sisendiks saadud reeglite põhjal. Teiseks on ekspertsüsteem, mille graafilise kasutajaliidese abil saab seadistada pildi malle. Määrates piirkondi, millele tuvastust hiljem soovitakse teostada ning tähemärke, mis tuleks saata analüüsiks. Peale seadistusi on võimalik ekspertsüsteemi veebiteenusele teha päringuid uute piltidega, tagastuseks on tuvastuse tulemus. Kolmandaks oluliseks tulemuseks on kaks valminud prototüüplahendust, mis demonstreerivad lahenduse kasutusvõimalusi. Üks neist on Androidi nutitelefoni rakendus, mille arenduses kasutati ekspertsüsteemi ainult esialgseks analüüsiks. Uue piltide puhul kasutab see otse algoritmi realisatsiooni. Teine on lihtne veebirakendus, mis kasutab ekspertsüsteemi veebiteenust küsimustiku vormi vastuste automaatseks süsteemi lugemiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 81 leheküljel, 9 peatükki, 18 joonist ja 7 tabelit.

Rauno Rüga

Development of an Algorithm and Expert System for Image Text Recognition

Master's thesis

Abstract

The main aim of this master's thesis is to develop a solution that can be used in applications to recognize text characters within images in restricted conditions, in which images have a similar structure, the possible characters are known in advance and the characters are always in the same format. Important aspects are speed, precision, size and suitability for smartphone applications.

The thesis will provide a through description of the background of the problem and how the results can potentially be used. A comparison is provided of alternative solutions, their negative and positive aspects and why the author of the thesis decided to develop his own solution for the problem. Later, a description is given of the problems appearing during the development process and the development of components which eventually provided a solution for the initial problem.

The project has three main outcomes. Firstly, an algorithm was developed for character analysis on the basis of input rules. Secondly, an expert system was developed with a graphical user interface for image template configuration. The user can select areas where OCR should be processed and mark characters that have to be sent for analysis. After configuration, it is possible to send an image as an input for the expert system web service and receive recognized text as a response. The third important outcome is the completion of two prototype applications developed for demonstrative purposes. The first one is an Android smartphone application for which the expert system was used as an analysis tool only. Analysis of new images analysis is conducted through integrated algorithm implementation. The second prototype is a simple web application that uses the expert system web service for automatic detection of questionnaire results.

The thesis is in Estonian and contains 81 pages of text, 9 chapters, 18 figures and 7 tables.

Lühendite ja mõistete sõnastik

OCR ***Optical Character Recognition***
Optiline tekstituvastus [1]

MB ***Megabyte***
Megabait ehk ligikaudu 1 miljon baiti, täpsemalt 2^20 ehk 1 048 576 baiti või 1024 kilobaiti [1]

API ***Application Programming Interface***
Rakendusliides ehk arvuti operatsioonisüsteemiga või rakendus-programmiga määratud reeglistik, mille alusel rakendusprogramm kasutab operatsioonisüsteemi või teise rakendusprogrammi teenuseid [1]

RGB ***Red Green Blue***
Rohe-sini-puna ehk värvusruum videopildi kuvamiseks arvutiekraanil põhivärvuste (roheline, sinine ja punane) kombinatsioonidena. Iga piksel ekraanil moodustatakse tegelikult kolmest eri värvi punktist, mille intensiivsuse reguleerimisega saadaksegi kõik vajalikud värvitoonid. [1]

Malltuvastus ***Template matching***
Meetod kujundite või kuju tuvastamiseks mallidega võrdluse teel. [2]

Tehis-närvivõrk ***Artificial neural network***
Kaalutud sidemete kaudu ühendatud elementaarsetest töötluselementidest koosnev võrk, milles iga element tekitab mingi väärtuse, rakendades oma sisend väärtusele mingit mittelineaarset funktsiooni, ja edastab selle väärtuse teistele elementidele või esitades ta väljundina. [2]

Avatud lähtekood ***Open source***
Üldjuhul nimetatakse avatud lähtekoodiks mistahes programmi, mille lähtekood on tehtud programmeerijatele ja kasutajatele kättesaadavaks nii kasutamiseks kui muutmiseks. [1]

SDK	***Software Development Kit*** Programmipakett, mis võimaldab programmeerijal luua rakendusi konkreetsele platvormile. [1]
Piksel	***Pixel*** Tuletis sõnadest „picture“ ja „element“, seega pildielement. Arvuti ekraanile kuvatav pilt koosneb neljakandilistest elementidest – pikslitest – ja mida suurem on pikslite arv ekraani pinnaühiku kohta, seda teravam paistab silmale pilt. Koosneb kolmest eri värvi punktist – punane, roheline ja sinine. [1]

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Juba mitu aastat on olnud kasvav trend kaasaskantavate nutiseadmete müügis ning see jätkub lähitulevikus. Enamusel neist seadmeist on sisseehitatud kaamera, millega saab teha juba väga kõrge kvaliteediga pilte. Saadud tulemust saab jäädvustada meenutuseks, jagada sõpradega sotsiaalmeedias ning kasutada fotol olevat informatsiooni edasiseks töötlemiseks. Selleks, et oleks andmeid, mida töötlemisel uue tegevuse sisendiks anda, tuleb pildil olevad kujutised tõlgendada ümber masintöödeldavale kujule. See aga pole enam triviaalne tegevus ning vajab tihti konkreetsele situatsioonile omast lähenemist.

1.1 Taust ja probleem

Antud magistritöö annab lahenduse, kuidas tuvastada pildilt teksti kindlalt fikseeritud tingimuste puhul. Eelduseks on see, et pildid on struktuurilt sarnased ehk tekst asub kindlas asukohas, erinevad on ainult esinevad tähemärgid. See loob võimaluse kasutada rakendustes teksti pildituvastust, mis on kohandatav konkreetse juhu jaoks ning seetõttu kiirem ja täpsem üldlevinud lahendustest.

Töö võib pakkuda huvi kõigile, kellel on IT alased algteadmised ja on huvitatud intelligentsete süsteemide toimimisest ja eriti sellest, kuidas masinad suudavad piltidel olevat teksti tuvastada. Töö tulemusi saaksid potentsiaalselt kasutada rakenduste ja tarkvarasüsteemide arendajad, kellel on vaja teostada piltide tekstituvastust, aga olemasolevad lahendused pole andnud soovitud tulemusi.

1.2 Ülesande püstitus

Magistritöö põhieesmärgiks on välja töötada uus algoritm, mis võimaldaks analüüsida pildil olevaid kujutisi ning väljastada seejärel iseloomustavaid parameetreid. Selle informatsiooni põhjal oleks võimalik pakkuda senisest paremat teksti pildituvastust juhtudel, kui pilte on palju ja tekib juurde, kuid iga pilt on seotud mingi alusmalliks oleva struktuuriga. Samuti peab algoritm olema realiseeritud sellises programmeerimiskeeles, mis võimaldaks selle hilisemat kasutust erinevatel platvormidel.

Teiseks eesmärgiks on arendada ekspertsüsteem, mis kasutab eelnevalt välja töötatud algoritmi võimalusi lahendamaks algselt püstitatud probleemi. Selle abil saab kasutaja kui ekspert seadistada personaalse teksti pildituvastuse ning hiljem ekspertsüsteemi liidese abil teha piltidel oleva info saamiseks päringuid.

Kolmas oluline eesmärk on algoritmi ja ekspertsüsteemi toimimist demonstreerivate prototüüp rakenduste arendus. Selle jaoks arendatakse üks Androidi rakendus, mis peale eelnevat seadistamist kasutab otse pilte analüüsivat algoritmi ja veebirakendus, mis seda läbi ekspertsüsteemi teeb.

1.3 Metoodika

Töös seatud eesmärkideni jõudmiseks pannakse algul paika nõuded ja tingimused, mille jaoks see lahendus on mõeldud. See pole mõeldud konkureerima tuntud pildituvastustarkvaradega, vaid olema parim valik juhul, kui on täidetud antud lahenduse eeldused. Küll aga uuritakse olemasolevaid lahendusi, kas kuskilt saaks osaliselt midagi kasutada või võtta inspiratsiooniks.

1.4 Ülevaade tööst

Magistritöö esimene peatükk on töö sissejuhatuseks, teises peatükis minnakse probleemi analüüsimises põhjalikumaks, selgitades pikemalt miks ja kellele töö lahendust vaja on.

Kolmandas peatükis tuuakse välja eesmärgid, mis tuleb töö käigus saavutada ja mis väärtus eesmärke täites luuakse.

Magistritöö neljas peatükk annab ülevaate olemasolevatest sarnastest lahendustest, mida nad võimaldavad. Lisaks mis on nende puuduseks, miks on siiski vaja midagi uut.

Viiendas ja kuues peatükk keskenduvad töö põhitulemuste loomisele, esimeses antakse ülevaade uue algoritmi arendamise protsessist ja teises kirjeldatakse ekspertsüsteemi loomist. Kuna kuues peatükk kasutab otseselt loodud algoritmi, lisades kasutajaliidese ja targa abisüsteemi, siis on need töö osad omavahel tihedalt seotud.

Seitsmendas peatükis arendatakse kaks rakendust, mis demonstreerivad eelnevates peatükkides loodud algoritmi ja ekspertsüsteemi kasutust. Järgnevas kaheksandas peatükis

antakse magistritöö käigus arendatud lahendusele hinnang ning võrreldakse saadud tulemust analüüsitud alternatiividega. Seda nii kiirust, täpsust kui ka mahtu võrreldes.

Viimane peatükk võtab kokku töö tähtsamad punktid, mis oli probleem, kuidas seda kavatseti lahendada ja mis seati eesmärkideks. Seejärel antakse hinnang, kas lõpptulemus vastas algselt seatud eesmärkidele.

2. Probleemi analüüs

Käesolev peatükk annab põhjalikuma ülevaate probleemist, mis tingis vajaduse lahenduse järgi, mis lõputöö eesmärgiks on seatud. Olulise informatsioonina kirjeldatakse need piirangud ja eeldused, mis käesoleva töö pildituvastuse tavalise tekstituvastusest teostamisest erinevaks muudab. Lisaks tuuakse välja alternatiivsed lahendused, mis võiksid täita sama eesmärki, kuid eelduste tõttu ei tööta piisavalt hästi.

2.1 Probleemi teke

Magistritöö sai alguse sellest, et lõputöö autor tegeleb hobi korras nutitelefonidele rakenduste arendamisega. Tekkis soov kasutada programmis tekstituvastuse funktsionaalsust. Kui kasutaja on teinud lotopiletist pilti, siis tuvastatakse pildil olevad numbrid. Selle info põhjal kontrollitakse ja antakse teada, kas tegemist on võidunumbritega.

Üsna varsti sai selgeks, et vaja läheb spetsiaalset pildituvastuse API-liidest, mis rakenduse kõige keerulisema osa ehk pildituvastuse teostaks. Uuritud sai erinevaid pildituvastuse lahendusi ja valitud välja Tesseracti [1], mis on üks enim levinud ja foorumites programmeerijate poolt soovitatud pildituvastuse API-liides. Katsetamisel selgus, et numbrid küll tuvastatakse pildilt, aga lisaks sellele tagastatakse veel muud ebavajalikku informatsiooni, näiteks püstkriipse ja punkte. Lisaks ilmnes, et kui pilt ei olnud väga selge, siis oli probleeme õigete numbrite tuvastamisega.

Peale esile toodud probleemide tuli välja veel see, et tuvastus on aeglane ning tagastatud andmed ei ole väga sobival kujul, et edasiseks töötluseks saata. Näiteks on raske vahet teha, kas number kaks on numbri kaksteist teine komponent või eraldi number. Soovitud tagastus võiks olla ideaaljuhul eelnevalt defineeritud kindla suurusega massiiv, mis täidetud õigete andmetega.

Seega tuli leida midagi muud, mis antud olukorras sobivam oleks. Ideaalse lahenduse puhul tuli arvestada järgnevaid tingimusi, mis piiravad ja konkreetsemalt toovad välja sobiva valiku nõuded:

1) Pilte on palju, aga kõik on sarnase struktuuriga ehk oluline informatsioon paikneb piirkonnas, mis on kindlalt teada. Seega võiks saada pildi mallil selekteerida alad, kust andmeid tuvastada.

2) Kui mingi ala on selekteeritud, siis peaks saama defineerida, mitu andmelõiku seal on ja kas ühes või kahes dimensioonis. Selle peale peaks tarkvara jaotama selekteeritud ala omakorda tükkideks ning tagastama tuvastatud märgid selliselt, et oleks aru saada, kus mingi tähemärk paikneb.

3) Algselt peaks saama selekteerida ja seadistada pildil olevatele kujutistele vastavad tulemused. Näiteks olukorras, kui on pildil ruudustik, kus on ringid ja ristid, siis peab saama määrata, et ring vastab numbrile 0 ja rist numbrile 1, kui tagastatakse tuvastuse tulemus.

2.2 Olemasolevate lahenduste piirangud

Erinevate OCR lahenduste uurimise tulemusena sai selgeks, mis võimalused ja piirangud olemasolevaid lahendusi kasutades on. Järgnevalt on välja toodud leitud piirangud ja negatiivsed aspektid. Need said määravateks argumentideks soovile arendada uus lahendus, selle asemel et kasutada midagi olemasolevatest.

Eelnevalt juba mainitud Tesseracti algoritmi puhul kasutas magistritöö autor selle mugavdatud Androidi projekti versiooni Tess-two [2]. Antud valiku puhul oli esimene ohumärk juba see, et Androidi klassiteegid olid mahult kokku enam kui 35 MB. See on mobiilirakenduse jaoks juba üsna märkimisväärne suurus. Teine puudus ilmnes testimise käigus, kui selgus, et pildi töötlemine võttis mitu sekundit ja tulemus ei olnud alati täpne, vaid oli eksimusi ja sellist informatsiooni, mis on lihtsalt segavaks taustamüraks.

Uurides veel teisi OCR lahendusi, mille tutvustused internetis kättesaadavad on, sai üsna pea selgeks, et suurem osa potentsiaalsetest lahendustest on liiga üldised ja mahukad. See tähendab, et nad on mõeldud töötama suurte dokumentide või pikkade tekstide pildituvastuseks. Samuti puudub võimalus saada tuvastuse vastuseks kindla suurusega numbrite massiivi, selle asemel saab pika tähemärkide jada, mida on keeruline edasi töödelda.

Kuna antud töö algse probleemi lahendamiseks on vaja rakendusest pidevalt pildituvastuse osa välja kutsuda, siis sellest tingituna on konkreetne piirang, et lahendus peab olema

kasutatav kas tavalise API-ina või veebiteenusena. Tavaliste API-ide puhul aga pole sellist lahendust, kus saaks eelnevalt mingi programmi abil teatud alasid selekteerida ja defineerida võimalikke tagastatavaid tähemärke. Seega peaks iga kord kogu alade selekteerimise ja töötluse tegema rakenduse kihis, mis aga iga uue rakenduse puhul tähendaks korduvat sarnast tööd ja ning lahendus poleks seetõttu üldkasutatav.

Alles jääb võimalus kasutada veebiteenuseid, mis pakuvad piltidel oleva info tuvastust. Sellist lahendust, mis rahuldaks kõiki eelmises alampeatükis välja toodud nõudeid aga ei leidunud. Saadaval on palju OCR veebiteenuse lahendusi, kuid nõuete järgi oli vaja alade eeldefineerimist, tagastatavate andmetüüpide struktuuri määramist ning kasutajapoolselt pildi kujutise ja tulemuse vastavusse seadmist.

Täpsem ülevaade uuritud lahendustest on eraldi alternatiivsete lahenduste suure peatüki all, kus on välja toodud ka see, mis leevendavatel asjaoludel siiski saaks kasutada olemasolevaid lahendusi, kuid miks see siiski ei ole hea valik.

2.3 Vajadus paremale tekstituvastusele

Kuna algselt püstitatud probleemile ei leidnud lõputöö autor olemasolevat head lahendust, siis sai magistritöö eesmärgiks võetud sellise süsteemi väljatöötamine, mis täidaks probleemipüstituses esitatud nõudeid.

Üks olulisemaid nõudeid on see, et välja töötatud lahendus oleks kiire. Kui on juba eelnevalt võimalik defineerida, mida ning mis pildi piirkonnast tuvastada tuleb, siis saab töötluse ka ära piirata, et ei tehtaks üleliigset tööd üritades tuvastada seda, mis ei ole antud juhul vajalik. Kiirus on aspekt, mis nutitelefonidel on veel eriti oluline, kuna näiteks võrreldes arvutitega jääb nende keskmine jõudlus veel kõvasti alla. Tingitud on see suuresti sellest, et nende protsessorid on lihtsalt tunduvalt väiksemad. Ja kuna samuti ekraan on enamus juhtudel väiksem, on kasutaja tähelepanu rohkem fokuseeritud, oodates kohest vastust.

Samuti annaks palju lisaväärtust selline võimalus, kui oleks võimalik enne pildianalüüsi lülitada sisse või välja erinevad reeglid, mida analüüsi lõpptulemus sisaldada võiks. Tänu sellele muutuks algoritmi käitumine vastavalt olukorrale. Näiteks kui oleks vaja pildilt tuvastada numbreid null ja üks, siis saaks kasutada lihtsamaid reegleid võrreldes sellega kui võimalik tulemuste ring oleks nullist üheksani. See mõjutaks nii analüüsiva algoritmi kui ka hilisema tähemärgi tuvastuse kiirust ja täpsust.

Oluliseks aspektiks, mis muudaks lõpprakenduse poolset töötlust oluliselt lihtsamaks ja loogilisemaks, oleks võimalus määrata mis kujul tuvastuse tulemust saada soovitakse. Näiteks kas massiivi 5 elemendiga või siis 7x7 maatriksit. Sellisel juhul saab rakenduse kihis teha töötluse konkreetse juhu jaoks. Kontrollida oleks vaja ainult seda, kas mingil konkreetsel positsioonil asub mingi tähemärk või on see tühi.

2.4 Lahenduse huvipooled

Magistritöö tulemuse huvipooled võib jaotada kolmeks. Esiteks tarkvaraarendajad, kes otseselt kasutaks oma süsteemis või rakenduses kas siis algoritmi või ekspertsüsteemi. Teiseks on lõppkasutajad, kelle elu muutuks tänu magistritöö tulemuse rakendustele mugavamaks või siis toimiks see meelelahutusena. Kolmandaks huvipooleks võib jagada need inimesed, kes on huvitatud intelligentsete süsteemide toimimisest ja sooviksid tutvuda praktilise teostuse detailidega.

Tarkvaraarendajad saaksid antud töö lõpptulemust kasutada erineval moel. Esimesena tuleks kasutada ekspertsüsteemi, kus pildi mall või mitu näidist üles laadida ja määrata ära, mis tähemärkidele pildil olevad kujutised vastavad. Seejärel defineerida ära huvipakkuvad pildi piirkonnad ja soovitav tagastuse info andmestruktuur. Edasi saaks teha integratsiooni ekspertsüsteemi veebiteenusega, kuhu sisendiks anda pilt, ning tagastuseks tuleb eelnevalt ise seadistatud andmestruktuur. See võimalus eeldab aga, et lahendus oleks avalikus veebis kättesaadav, mis on aga lõputöö autori hilisemaks eesmärgiks ja edasiarenduseks. Esmane kasu magistritöö tulemusest oleks eelkõige lõputöö autorile endale, kes saaks seda ärilistel eesmärkidel kasutada, realiseerides pildituvastust mobiilirakendustes. Algul kasutades lokaalses masinas ekspertsüsteemi ning hiljem rakenduses otse analüüsivat algoritmi, võrreldes uute piltide analüüsi tulemusi eelnevalt leitud mallpildil andmetega.

Teine välja toodud huvigrupp oli lõppkasutajad. Selle all on mõeldud inimesi, kelle elu võib muutuda tänu välja töötatud lahenduse rakendustele mugavamaks. Näiteks arendades rakenduse, mis peale lotopiletist pilti tegemist tuvastab seal olevad numbrid ja kontrollib kas tegemist on võidunumbritega ning annab kasutajale selle kohta tagasisidet. See võtaks aega vaid paarkümmend sekundit, võrreldes mitme minuti või enamaga kui kasutaja ise kontrolliks neid numbreid ühekaupa või siis viiks pileti kuhugi poodi kontrolli. Samamoodi saaks teha pilti Sudoku väljast ja küsida lahendust või vihjet, kui endal ideed otsa saanud.

Viimaseks huvigrupiks oleks need inimesed, kes õpivad intelligentsete süsteemide toimimist või on lihtsalt antud teemast huvitatud. Vajalik oleks algteadmiste pagas optilise pildituvastuste mõistetest ja olemusest ning samuti infotehnoloogia alaste baasteadmiste olemasolu. Antud magistritöö oleks üks võimalik näide, kuidas arendada ekspertsüsteemi ning seda realiseerivaid rakendusi, mis teostaksid tegevusi [3], mida peetakse omaseks intelligentsetele süsteemidele.

3. Töö eesmärgid

Magistritööl on kolm põhieesmärki, mida saab eraldi käsitleda ja mis on praktilise ning rakendusliku loomuga. Selleks, et jõuda eesmärkide püstitamiseni, tuli põhjalikult analüüsida ja hinnata probleemi. Selle tulemusena selgus, mida vaja oleks ning kuidas seda saavutada võiks. Püstitatud eesmärkidest esimesed kaks on seotud lahenduse väljatöötamisega magistritöös püstitatud probleemi jaoks. Kolmas eesmärk on eelnevalt arendatud komponentide toimimise demonstreerimine praktiliste näidete peal.

3.1 Uus algoritm piltide reeglipõhiseks analüüsiks

Esimene oluline eesmärk on uue algoritmi väljatöötamine ja selle hilisem realiseerimine programmeerimiskeeles, mis oleks kasutatav võimalikult paljudel erinevatel platvormidel. Algoritm on mõeldud pildi analüüsimiseks arvestades lubatud reegleid. Tähemärkide tuvastust sealjuures ei tehta, vaid see ülesanne jäetakse teiste komponentide katta.

Algoritmi tööpõhimõte seisneb selles, et ta saab sisendiks pildi ja palju parameetreid, mis määravad reeglid, mille alusel pilti töödelda tuleb. Sisendparameetriks olev pilt võib olla nii tervikpilt kui ka osa mingist suuremast pildist, mis on tükeldatud analüüsi jaoks. Saadud informatsiooni põhjal peab algoritm suutma töödelda pildil olevaid kujutisi, kaasates määratuid reegleid. Reegel võib näiteks olla see, et leia tumedate pikslite osakaal või siis vastupidi heledate pikslite osakaal. See, mida tähendab tume piksel ehk milline on RGB erinevate värvide jaotus, tuleks samuti otsustada defineeritud reeglite põhjal.

Vajadus kirjeldatud analüüsiva algoritmi jaoks on seetõttu, et siis saab tuvastada mingile konkreetsele tähemärgile iseloomulike parameetreid. Saadud informatsiooni saab hiljem kasutada alampeatükis 3.2 kirjeldatav ekspertsüsteem tähemärkide tuvastamiseks, võrreldes seda uute sisenditega. Oluline on ka loodud algoritm programmiks realiseerimine sellises programmeerimiskeeles, mida saaks kasutada erinevate platvormide peal. See võimaldaks välja töötatud funktsionaalsuse laialdast kasutust, ilma et oleks vaja erinevate keskkondade jaoks dubleerida pildi analüüsi. Samuti tooks kesktaseme keeles realiseeritud algoritm kiirema töötluse võrreldes kõrgtaseme keeltega.

3.2 Uut algoritmi kasutav ekspertsüsteem piltide tekstituvastuseks

Teine suurem eesmärk on ekspertsüsteemi arendus, mis kasutaks eelnevalt välja töötatud algoritmi, et analüüsida erinevaid pildil olevaid kujutisi ning saadud informatsiooni põhjal salvestada andmebaasi tähemärkidele vastavad omadused. Nende andmete põhjal saab hiljem uusi sisendeid võrrelda olemasolevatega ja reeglite põhjal uuelt pildilt tähemärgid tuvastada.

Ekspertsüsteem peaks toimima veebirakendusena, mis on mõeldud kasutamiseks avalikule kasutajaskonnale. Võimalik on registreerida kasutaja, millega süsteemi sisse logida. Registreeritud kasutajale avaneb võimalus laadida süsteemi üles pilte, millel saab selekteerida alasid, millel olevate tähemärkide tuvastamist soovitakse teostada. Lisaks tuleb selekteerida eraldi kõik võimalikud tähemärgi kujutised ning defineerida, mis tähemärgile vastab mingi konkreetne kujutis. Siit tuleb välja ka ekspertsüsteemi õppiv osa ehk kasutaja kui ekspert ise õpetab süsteemile, mida mingi märk tähendab. Seejärel peab olema võimalus määrata, mis reeglite vastu edasine analüüs toimib ehk tähemärk seotakse süsteemis analüüsiva algoritmi tulemustega.

Eelnevas lõigus kirjeldatu on mingi uut tüüpi pildi tuvastuse algne seadistamine, et süsteemi õpetada, mille alusel ta töötama peaks. Et hiljem reaalset tulemust näha ja piltidel olevat infot tuvastada, selleks on vaja kasutada ekspertsüsteemi veebiteenust. Selle eesmärgiks oleks sisendinfo vastuvõtt, milleks on pilt baidikujul ja identifitseerivad parameetrid. Selle info põhjal saab ekspertsüsteem aru, mis pildiga on tegu ja leiab andmebaasist seotud reeglid ja õpetatud tähemärgid. Ekspertsüsteem peaks seejärel teostama uue pildi analüüsi ja õpetuse kohaselt tuvastama tähemärgid ning tagastama tulemused veebiteenuse väljakutsujale. Sealjuures peaks tagastatud andmestruktuur olema selline, nagu varasemalt õpetamise faasis defineeritud.

3.3 Rakendused lahenduse demonstreerimiseks

Selle jaoks, et magistritöö käigus loodud lahendus piltide tekstituvastuseks piiratud tingimustes ei jääks teoreetiliseks ning et tuleks välja selle toimimine reaalses olukorras, on lõputöö autoril eesmärgiks arendada kaks rakendust prototüübi kujul, mis välja töötatud lahendust kasutaks. Esimene näide oleks nutitelefoni rakendus Androidile ja teine ASP.NET veebirakendus, mis tooks välja kaks erinevat lähenemist, kuidas eelnevates alampeatükkides kirjeldatud lahendust rakendada saaks.

Esimene näide, milleks oleks Androidi nutitelefoni rakendus, peaks toimima selliselt, et lõplikus kasutuses oleks ta sõltuv ainult pilte analüüsivast komponendist. Seda peaks olema võimalik saavutada selliselt, et algul kasutatakse ekspertsüsteemi, kus sisestatakse pildi mall, et sellelt selekteerida kujutised, mida tuvastada tuleb ning lasta neid analüüsida. Kui kõik võimalikud esile tulevad tähemärgid on ekspertsüsteemi poolt analüüsitud, siis teha väljavõte statistikana, et milliseid tulemusi saadi mingeid kindlaid reegleid rakendades ja salvestada informatsioon konstantidena rakenduse kihti. Näiteks kui suur oli tumedate bittide osakaal tähemärgil „0“ ja kui palju olid neid ülevalt esimeses veerandis. Selle informatsiooni põhjal oleks võimalik nutitelefoni rakenduses arendada loogika, mis võrdleks saadud andmeid uute piltide analüüsi tulemusega ning otsustaks seejärel mis tähemärkidega on tegu.

Eelneva lõigu lõpus kirjeldatud tegevused on mõeldud algselt ekspertsüsteemi ülesannete hulka, kuid teatud juhtudel on põhjust seda dubleerida ka lõpprakenduses. Tänu sellele on võimalik saavutada rakenduse sõltumatus ekspertsüsteemi veebiliidesest ning samuti ei ole oluline võrguühenduse olemasolu. See on eriti oluline nutitelefoni rakenduste puhul, kuna endiselt on paljudes riikides üle maailma võrguühendus aeglane, kallilt maksustatud või puudub teatud piirkondades üldse.

Teine praktiline näide demonstreerimaks välja töötatud rakendust oleks veebirakendus, mis kasutaks piltidel oleva info tuvastamiseks ekspertsüsteemi veebiteenust. Toimida tuleks sarnaselt mobiilirakendusega, ehk alguses kasutatakse ekspertsüsteemi pildi malli üles laadimiseks ja süsteemi õpetamiseks. Seejärel aga genereeritaks API võti, mille alusel saaks hakata saatma ekspertsüsteemile läbi veebiliidese uusi sarnase struktuuriga pilte. Varasemalt tehtud seadistuste ja määratud reeglite põhjal analüüsitakse uut pilti ning tagastatakse tuvastuse tulemus. Veebirakendus saab saadud tulemuse põhjal informatsiooni edasi töödelda.

4. Pildituvastuse tehnikad ja alternatiivsed lahendused

Selleks, et tuvastada piltidelt tähemärke, on kasutusel palju erinevaid tehnikaid ja metoodikaid. See teema on olnud juba aastakümneid aktuaalne ning avaldatud on palju uurimus- ja teadustöid. Käesolevas peatükis antakse lühike ülevaade erinevatest tehnikatest, mille võimaldavad seda teostada. Ainult teoreetilisest osast aga ei piisa, rakendustes kasutamise jaoks on vaja ka reaalset komponenti, kus valitud tehnika oleks realiseeritud. Seega teine osa käesolevast peatükist keskendub nendele lahendustele, mis on potentsiaalseks alternatiiviks magistritöö käigus loodud lahendusele.

4.1 Optilise tekstituvastuse tehnikad

Optilise piltituvastuse tehnikaid on palju, neist kümmekond levinumat. Järgnevalt antakse ülevaade mõnest populaarsemast [6], kuidas nad toimivad. Tehtud on vaid väike valik kõigist võimalikest ja kirjeldused on rakendamise selles faasis, kus on juba eelnev töötlus piltidele tehtud. Tihti tuleb eelnevalt muuta pildi tumedust, teha kindlaks mitmel real tähemärgid asetsevad ja pöörata kaldu või nurga all olevad tähemärgid sirgeks. Viimasel juhul saab kasutada näitaks teksti keskkoha määramist, ning selle järgi naabertähemärkide nurkade arvutamist ja sobiva nurga alla pööramist [7].

4.1.1 Malltuvastus

Malltuvastus on üks enim kasutatavaid tekstituvastuse tehnikaid, mis on ka toimimise poolt kergemini teostatavam ja arusaadavam kui alternatiivid. Algselt kogutakse süsteemis palju erinevaid tähemärkide prototüüpe ja näidiseid, erinevas suuruses ja stiilis. Hilisemas tuvastamise faasis võrreldakse uut sisendit, milleks on sisend tähemärk pildil, varasemalt kogutud kogumikuga. Leitakse sarnased tähemärgid ning kogutakse informatsiooni nende sarnasuse protsendi ja vea määra kohta. Sarnasus tehakse tihti kindlaks pildil olevate pikslite järgi, võrreldes RGB väärtusi malli ja uue pildi vahel. Lõplik tuvastuse tulemus väljastatakse selle põhjal, mis eelnevalt süsteemis või programmis olevatest tähemärkides osutus kõige sarnasemaks võrreldes uue uuritava sisendiga.

Selle tehnika kasutusvõimalused on väga suured ja rakendust leiab ka teistes valdkondades peale teksti tuvastamise. Üks levinud valdkond on näotuvastus turvalisuse tagamiseks, kus

kasutatakse sarnaseid meetodeid, ainult et võrdluse all olevaks objektiks on inimese näo pilt. Näiteks 2008 aastal toimunud Pekingi olümpiamängudel kasutati näotuvastussüsteemi avamis- ja lõpetamistseremoonial staadioni sissepääsu turvakontrollis [8].

4.1.2 Struktuurne klassifitseerimine

Teine võimalus tähemärkide tuvastamiseks on kujundite struktuuri uurimine ja reeglite järgi märkide klassifitseerimine. Struktuuri all on mõeldud erinevaid jooni, tühimike tähemärgi keskel, nurki, lohke ja muid sarnaseid omadusi. Näiteks kui võtta täht „P“, siis saaks seda kirjeldada kui vertikaalne joon, millel on paremal üleval ääres kaarjas poolring, mis on seest tühi. Kui programmi tuleb sisendina mingi uus tähemärk, siis analüüsitakse seda ja määratakse, mis teda kirjeldab. Seejärel kontrollitakse algselt seadistatud reegleid ja kui leitakse vastavus samade omadustega, on tähemärk süsteemi poolt tuvastatud.

4.1.3 Tehis-närvivõrk

Veel üks võimalus optilist tähemärgituvastust teostada on tehis-närvivõrkude kasutamine. See tehnika sarnaneb osalt klassifitseerimisega, kuid võib ka eraldi käsitleda. Algus toimib samamoodi ehk defineeritakse omadused nagu näiteks kurvid, suletud alad, horisontaalsed ja vertikaalsed jooned, sümmeetria ja muud parameetrid. Mida rohkem selliseid erinevaid aspekte arvesse võetakse, seda täpsem tuvastus on võimalik hiljem saavutada.

Järgnevalt toimub aga närvivõrkudele eriline käitumine ehk selle „treenimine“. Selle käigus võetakse suur kogus näite malle, mille erinevaid omadusi arvutama hakatakse. Programmeerija määrab millised on oodatavad lõpptulemused, ning „treenimise“ tulemusena on eesmärgiks anda erinevatele sisendparameetritele kaalud, kui oluline mingi parameetri väärtus on. Kui närvivõrk on „treenitud“, siis peaks ta uute sisendite puhul arvestama erinevate parameetrite osakaalu ning selle põhjal tagastama õige väljundi.

„Treenitud“ tehis-närvivõrkude puhul on ettevalmistav protsess väga oluline, kuna kui muutub teksti stiil, siis võib tuvastuse täpsus langeda väga kiirelt. On tehtud eksperiment [9], kus „treenimiseks“ on võetud standardsed suured tähed ja numbrid ning hiljem on testimise sisendiks antud uusi stiile. Kui näiteks *Arial* fondi puhul suutis süsteem endiselt 97.20 % täpsusega märke tuvastada, siis *Verdana* puhul 83.33 % ja *Georgia* fondis sisendi korral ainult 33.33 %.

4.1.4 Valik reaalses olukorras

Tegelik olukord, kuidas optilise tekstituvastuse programmid töötavad, on mitmekesisem võrreldes eelmistes alampeatükkides kirjeldatuga. Kasutatakse erinevaid tehnikaid koos ning lisaks veel konkreetsele situatsioonile omaseid kontrolle ja meetodeid. Kombineerides on võimalik saavutada teksti tuvastusel ligi 99% täpsust, kui tegemist on masina poolt välja prinditud tekstiga, mis on kindla stiiliga. Käsikirja puhul kõigub tuvastuse protsent veel palju, olenedes käekirjast ja keelest, milles tekst kirja pandud. Selle parandamiseks kasutatakse tekstituvastust koos teksti õigekirja ja lausete loogilisuse kontrollimisega, mis parandab esialgses tuvastuses tekkinud ilmselged vead.

4.2 Alternatiivsed lahendused

Magistritöö üks eesmärke on kasutada rakenduses pildi tekstituvastust, et tõlgendada pildil olev info masintöödeldavale kujule. Et seda funktsionaalsust programmis kasutada, peab seda olema võimalik kas otseselt koodist välja kutsuda või kasutada vastavat API-it. Järgnevalt on välja toodud 3 võimaliku lahendust, analüüsitud nende puuduseid ja tugevaid külgi. See on vaid väike valik kõigist võimalikest alternatiivsetest lahendustest. Vaatamata sellele aga siiski piisavalt põhjalik ülevaade, kuna ülejäänud lahendused toimivad sarnaselt.

4.2.1 Tesseract

Uurides erinevaid lahendusi jäi esimesena silma avatud lähtekoodiga Tesseract [3] OCR mootor, mida arendati algselt Hewlett Packardi laborites aastatel 1985 kuni 1994. Peale seda jäi projekt venima ning tehti vaid väiksemaid täiendusi, lisaks kirjutati kood ümber C programmeerimiskeele pealt C++ peale. 2005. aastal aga muutus see kommertstootest avatud lähtekoodiga tarkvaraks ning see tõi positiivseid muutusi. Google alustas 2006. aastal selle sponsoreerimist ning sellest ajast alates on käinud pidev töö selle paremaks muutmisele ja kümned tuhanded arendajad on seda oma süsteemiga integreerinud. [10]

Oma olemuselt on Tesseract OCR mootor, seega enne kasutamist tuleks teda muuta, et saaks seda kasutada kui API-it ning käivitada vastavalt oma soovile. Teine võimalus on kasutada juba valmis arendatud komponente, mida saaks kiirelt oma rakendusse integreerida, nagu näiteks Androidile mõeldud tess-two [4] API, mis kasutab Tessaracti OCR mootorit lisades mugandused kasutamaks seda Androidi rakenduste arenduses. Laadides koodi alla, mis eelnevalt mainitud OCR mootorit töös hoiab, ilmnes esimene ohumärk, mis ei soosi selle

kasutamist nutitelefoni rakendustes. Selle kogumaht on 36,5 MB, mis nutitelefonide rakenduse jaoks on märkimisväärne suurus ja võib tekitada komplikatsioone. Uurides lähemalt lähtekoodi tuleb välja ka see, et sisendi töötlus on üsna keeruline ja põhjalik. See on täpsuse huvides küll hea, aga antud magistritöö probleemi lahendamiseks oleks vaja midagi lihtsamat, mis töötaks ainult piiratud juhtudel, kuid siis hästi. Tesseract sobiks keerulisemate tekstituvastuse ülesannete lahendamiseks, kus pole aeg niivõrd oluline, vaid tulemuse täpsus muutuvates keerulistes oludes.

4.2.2 OCRopus

OCRopus [11] on avatud lähtekoodiga OCR raamistik, mis sai alguse 2007. aastal ning selles kasutas algselt Tesseracti OCR mootorit. 2009. aastal asendati see uue sisemise töötlusloogikaga ja sellest ajast alates ei ole need enam omavahel seotud, küll on sarnaselt eelnevas alampeatükis kirjeldatule projekti ja tarkvara arenduse toetajaks Google korporatsioon. Lisaks tähemärkide tuvastusele võimaldab ta eeltöötlemist, pildi segmentidesse jagamist, ridade tuvastust ja uue andmehulga treenimist. [12]

Eeltöötlemise all on mõeldud taustamüra eemaldamist. Tihti on piltidel kas kirju taustavärv või on näiteks mõned sõnad alla joonitud. Selline tuvastuse jaoks mitteoluline informatsioon eemaldatakse, et hilisemas töötluses ei tekitaks see segadust. Segmenteerimise käigus tehakse selgeks, mis tekst kuulub kokku, on sarnases lõigus või piirkonnas. Seda selleks, et lõpptulemuses ei oleks omavahel põimunud need lõigud, mis paiknesid algselt eraldi. Oluline on ka võimalus uut andmehulka treenida. See tuleb kasuks kui tahetakse tuvastada andmeid, mis on sellises keeles, mille kohta mallandmed puuduvad. Näiteks eesti keelele spetsiifilisi tähemärke paljud optilise tekstituvastuse tarkvarad ei tunne, kui pole nende põhist infot eelnevalt sisestatud või andmeid treenitud.

Kaaludes OCRopuse kasutamist magistritöö probleemi lahendamiseks, tulevad esile samad probleemid, mis eelnevalt uuritud lahendusega. Kogu koodiloogika maht on enam kui 20 MB ning algne kood ei ole kohe kasutatav API-ina. See on hetkel kasutatav vaid läbi käsurea ning selle muutmine eeldaks palju tööd ning põhjalikku olemasoleva koodi uurimist, et saada aru mis komponendid omavahel seotud on ja kuidas seda ülemistes kihtides kasutada. Arendaja on kodulehel välja toonud ka soovituslikud nõuded, milleks on Ubuntu 64 bitine operatsioonisüsteem (versioon 12.6 või suurem), 4 gigabaiti mälu ning kiire protsessor [11]. Nende nõuete järgi sobiks see lahendus ainult serverile töötama, mitte aga kasutaja nutiseadmesse.

4.2.3 ABBYY

Uurides võimalusi, kuidas saaks kasutada piltidel oleva teksti tuvastust ilma suuri klassiteeke programmikoodi lisamata nagu eelnevad lahendused oleks nõudnud, jäi silma ABBYY [13]. See võimaldab kasutada OCR võimalusi läbi veebiteenuse. Teenusele tuleb saata pilt ja määratud seaded ning tagastatakse seadistatud kujul tekstituvastuse tulemus. Tegemist on kommertstootega, kuid arenduse ja testimise faasis saab seda tasuta proovida.

Testimaks kui hästi ABBYY töötab viis magistritöö autor läbi katse. Tõmmati alla tootja kodulehelt arvutisse testprogramm, milles muudeti koodi, et töötlusse võetav pilt oleks arvutis tehtud pilt Sudoku väljast, mis on ka antud töö lisa 1 all nähtav. Seejärel käivitati testprogramm, mis tegi ABBYY online teenusele päringu, eesmärgiga tuvastada pildil olevad numbrid. Päringu tagastuse tüübiks määrati tekstidokument ning selle tulemus on näha joonisel 1. Test oli edukas, tuvastada suudeti kõik numbrid. Uurides tulemust tekstiredaktorist ilmneb ka see, et tühjad kohad ruudustikus on tabeldusmärgiga eraldatud, seega saaks seda kasutada ka edasisel töötlusel. Korrates seda katsest erineva pildiga, milleks võeti nutitelefoniga tehtud foto samast arvutiga tehtud pildist (näha töö lisa 2 all), ei suutnud online teenus ühtegi numbrit tuvastada. Tingitud oli see sellest, et taust oli muutunud kirjumaks, ning seetõttu oli tuvastamine häiritud.

```
1     3           1     7
2  6        8              2
3        1     4     5
4     7              2     4
5  2              9              6
6     4     3              1
7        5     3     4
8  1                    6              5
9     2     1              3
```

Joonis 1 ABBYY OCR teenuse tulemus Sudoku pildile

Tehes veel teiste piltide peal katseid, kus tuli tuvastada teksti, oli märgata, et ABBYY OCR teenus töötab keskmiselt kiirelt (keskmiselt saabub tulemus umbes 2 sekundiga). Kui taust on valge, siis ei teki enam probleeme ning tuvastatud saavad ka sõnad hägusemate piltide pealt. Mõeldes arendajate peale, on ABBYY–l väga hea tugi. Eraldi SDK on 8 erinevale programmeerimiskeele ja –keskkonna jaoks, lisaks veel 3 põhilisele mobiiliplatvormile. Vaatamata sellele ei ole otstarbekas seda kasutusele võtta, kuna litsentseeritud versiooni iga

pildi tuvastus maksab 3 kuni 10 eurosenti, mis laiale massile kasutuseks mõeldud nutitelefoni rakenduses ei tasuks ära.

4.3 Alternatiivsete lahenduste puudused

Alternatiivsetest lahendusest käsitleti vaid väikest osa kõigist võimalikest lahendustest. Ülejäänud võimalikud olemasolevad lahendused, mida võiks kasutada antud lõputöö probleemi lahendamiseks, omavad sarnaseid puudusi ja ebasoosivaid omadusi, mida eelnevalt kirjeldatud lahenduste puhul juba välja on toodi. Kuna tähemärkide ning muu informatsiooni tuvastamine on uurimuste teemaks olnud aastakümneid ning selle realisatsioone on sadu, siis kõikidega realisatsioonidega ei jõua detailselt tutvuda, kuid välja joonistuvad lahendustele ühised jooned.

Vabalt kättesaadava avatud lähtekoodiga tarkvara puhul on tihti probleemiks see, et lahendus on üles ehitatud olemaks võimalikult universaalne. See toob kaasa selle, et pildituvastuse toimimise loogika on arendatud üsna keeruliselt ning klassiteegid ja mallandmed on väga mahukad. Võimalik oleks küll isiklikuks otstarbeks vajamineva osa eraldada ja ainult seda kasutada, kuid kuna selliste lahenduste puhul on pildituvastusel kasutatavad tehnikad keerulised ja tihedalt põimunud, siis oleks selle vajaliku osa leidmine ning ümber tegemine isiklikuks otstarbeks keeruline ja aeganõudev.

Teine võimalus oleks kasutada kas juba valmis kompileeritud SDK või online veebiteenuseid, mis tekstituvastuse funktsionaalsust pakuvad, nagu näiteks varasemalt kirjeldatud ABBYY [13]. Suurem enamus sellistest lahendustest on aga kommertstooted, ning hinnaklass on väga kallis. Toodete puhul on aastane litsents keskmiselt sadadest eurodest kuni paari tuhande euroni. Online veebiteenuste puhul on maksustamine tehtavate päringute koguarvu pealt. Kui võttes nutitelefoni rakendusele näiteks 10 000 aktiivset kasutajat kuus, siis tähendaks see juba minimaalset 300 eurost arvet. Seetõttu võib väita, et kommertstooted ei ole hobi korras rakendusi arendavale arendajale hinna poolest vastuvõetavad ning ei sobi antud magistritöös püstitatud probleemi lahendamiseks.

Kui ka leidub selliseid tekstituvastuse online veebiteenuseid või kompileeritud SDK, mis on tasuta ja mida saaks potentsiaalselt kasutusele võtta, siis jääksid segama järgnevad aspektid:

1) Tarkvara tugi puudus – kui midagi saab tasuta, siis on enamus juhtudel selle tarkvarakomponendi arendaja motivatsioon kiirelt reageerida ja parandusi teha väga väike või puudub üldse.

2) Kasutatav toode või teenus on kui must kast – puudub ülevaade, kuidas see sisemiselt realiseeritud on. Probleem tekib siis, kui midagi enam ei tööta. Sellisel juhul võib olla probleemi algallika leidmine väga keeruline või vahel ka peaaegu võimatu.

3) Teenus suletakse või esineb ajutisi tõrkeid – tasuta kasutuses oleva veebiteenusega võib juhtuda, et see suletakse äkiliselt ilma ette teatamata või serveriga juhtub midagi ning siis ei anta informatsiooni, kas see on ajutine või püsiv.

Arvestades magistritöös püstitatud probleemi konkreetsust ning tekstituvastuse funktsionaalsuse vajadust vaid piiratud tingimustes ja alternatiivsete lahenduste puudusi, on magistritöö autor otsustanud arendada oma lahenduse. Kuna piltide tekstituvastusel kasutatavad erinevad tehnikad on teada ja informatsiooni on antud teemal laialdaselt saada, siis on võimalik arendada lahendus, mis on optimeeritud just käesoleva magistritöö probleemi lahendamiseks.

5. Algoritm

Üks magistritöö kolmest peaeesmärgist on uue algoritmi väljatöötamine ja selle realiseerimine programmeerimiskeeles, mis võimaldaks hiljem selle kasutust võimalikult suurel hulgal erinevatel platvormidel. Algoritmi eesmärk on pildi analüüsimine erinevate parameetrite järgi ja informatsiooni kogumine, et selle põhjal otsuseid teha. Käesolev peatükk annabki ülevaate milline on algoritmi tööpõhimõte, kuidas see on realiseeritud ning millisel moel on realisatsiooni lõpptulemus kasutatav.

5.1 Vajalik funktsionaalsus

Esimene oluline tegevus enne algoritmi väljatöötamist oli vajaliku funktsionaalsuse kirjapanek, et mida sellega saavutada loodetakse ja kuidas ta toimima peaks. Nõudeid kirjeldades tuli mõelda juba ka sellest, et milline peaks olema programmikoodiks realiseeritud algoritmi lõppkasutus. Selle informatsiooni põhjal oli võimalik funktsionaalsed nõuded täpsemini kirjeldada ning vähendada võimalust, et realiseeritakse midagi, mida tegelikult ei lähe vaja.

Järgnevalt on välja toodud põhinõuded, mida tuleb võtta aluseks algoritmi väljatöötamisel:

1) Sisendiks on töötlemist vajav pilt ja suunav reeglite kogum. Pilt võib olla erinevas suuruses, ning see tuleb teha kindlaks jooksavalt vastavalt uuele sisendile. Suunav reeglite kogumik määrab mis analüüsi reegleid on vaja käivitada ning millised võib kõrvale jätta.

2) Selleks, et algoritmi tööülesanne oleks võimalikult konkreetne ja tulemus täpne, siis on eelduseks see, et tuvastatakse maksimaalselt kaks tähemärk. Informatsioon kas tuleb arvestada ühe märgiga või võib neid rohkem esineda, antakse sisendina reeglite kaudu edasi. Mõeldud on see just eelkõige numbrite tuvastamiseks, et kahekohalised numbrid analüüsitaks samaaegselt koos. Kui reeglites on valik tehtud, siis tähendab see seda, et on võimalik kahekohalise tähemärgi esinemine, kuid võib ka üks tähemärk olla. Siit järeldub see, et sisendit andev süsteem on ise vastutav selle eest, et sisend vastab nõuetele ja pilt on õiges asendis, mitte nurga all või tagurpidi. Vajadusel

tükeldatakse suurem pilt väiksemateks osadeks, et seda algoritmi realisatsioonile sisendina ette anda.

3) Sisendiks olevad reeglid jagunevad iseloomu poolest kaheks. Ühte gruppi kuuluvad reeglid määravad algoritmi käitumise, näiteks mida tuleb mõelda tumeda piksli all või kas esinema peaks üks või enam tähemärki. Teine grupp reegleid määrab selle, mis arvutusi tuleb käivitada ja mis informatsiooni tagastada tuleb.

4) Algoritm peaks olema kiire ja efektiivne ning abistava iseloomuga. Seetõttu jääb see pildil olevaid kujundeid analüüsivaks ja kirjeldavaks, tähemärkide tuvastamise loogika jääb selle skoobist välja.

5.2 Algoritmi väljatöötamine

Järgmine etapp peale nõuete koostamist oli algoritmi kirjeldamine pseudokoodis, et saada selge ülevaade tegevuste järjekorrast ning seejärel see toimivaks programmiks realiseerida. Arvestatud on juba eelnevalt kirjeldatud nõudeid ja üldistatud kujul magistritöö järgmistest peatükkides kirjeldatud reegleid. Järgnevalt on pseudokoodis kirjeldus pilte analüüsivast algoritmist, kus tuntud väljendid on inglise keeles, et oleks lugejale tuttavam ja paremini hoomatavam:

```
INPUT pilt, reeglid
DECLARE laius, kõrgus, esimeneTumeRide, viimaneTumeRida, tumedaPiksliLoendur,  esimeneTumeVeerg,
viimaneTumeVeerg, onÄär
SET laius TO (CALL LeiaLaius WITH pilt)
SET kõrgus TO (CALL LeiaKõrgus WITH pilt)
FOR EACH rida IN pilt
        FOR EACH veerg IN pilt
                IF (SUM(RGB) IN [rida,veerg]) <= (MaxTumeRGB FROM reeglid) AND NOT (CALL
KasOnÄäris WITH pilt, rida, veerg) THEN
                        INCREMENT tumedaPiksliLoendur
                        IF esimeneTumeRida IS UNDEFINED THEN
                                SET esimeneTumeRida TO rida
                        SET viimaneTumeRida TO rida
                        IF esimeneTumeVeerg > veerg OR esimeneTumeVeerg IS UNDEFINED
                                SET esimeneTumeVeerg TO veerg
                        IF viimaneTumeVeerg < veerg OR viimaneTumeVeerg IS UNDEFINED
                                SET viimaneTumeVeerg TO veerg
```

```
        END FOR
END FOR
DECLARE Tulemused
IF (MaxTähemärkideArv FROM reeglid) > 1 THEN
        DECLARE Tähemärgid
        SET Tähemärgid TO (CALL LeiaPildiltKõikTähemärgid WITH pilt, tumedaPiksliLoendur,
esimeneTumeRida, viimaneTumeRida, esimeneTumeVeerg, viimaneTumeVeerg)
        FOR EACH tähemärk IN Tähemärgid
                FOR EACH aktiivneAnalüüsiReegel IN reeglid
                        ADD(CALL  LeiaAnalüüsiReegliTulemus) TO Tulemused
                END FOR
        END FOR
ELSE
        FOR EACH aktiivneAnalüüsiReegel IN reeglid
                ADD(CALL LaiaAnalüüsiReegliTulemus) TO Tulemused
        END FOR
OUTPUT Tulemused
```

Pseudokood algab sellega, et kirjeldatakse sisendid ja lokaalsed muutujad, mida edasiseks töötluseks vaja läheb. Esmalt on oluline teada saada, millised on sisendiks saadud pildi mõõtmed ehk pikkus ja laius. Selle informatsiooni põhjal saab käivitada sobivas pikkuses iteratsioonid, mis kontrollivad pildi kõiki piksleid. Kui piksli RGB väärtuste summa on suurem kui reeglites defineeritud ja piksel ei ole pildi äärises paiknev eraldaja, siis käsitletakse seda kui tumedat pikslit ja hakatakse täitma käsublokki, mis ainult sellele tingimusele omane. Kui tingimus osutus tõeseks, siis suurendatakse tumedate pikslite loendurit, mille väärtust läheb hiljem arvutustes vaja. Lisaks järgneb loogika, mille ülesanne on selgitada välja koordinaadid, kus piirkonnas kujund pildi peal asub, mis on samuti oluline arvutustes, mida magistritöö järgnevates alampeatükkides põhjalikumalt kirjeldatud.

Järgnev lõik pseudokoodis mida võib ka teise osana käsitleda, on seotud aktiivsete reeglite põhise analüüsiga. Aktiivse all on mõeldud seda, et mingi analüüsi reegel on sisse lülitatud ehk mingi konkreetse parameetri väärtus tuleb arvutada. Enne seda aga kontrollitakse, kas on võimalik, et pildil paikneb rohkem kui üks tähemärk. Kui vastav sisendtingimus on tõene, siis tuleb käivitada eraldi loogika, mis leiab kõik erinevad kujundid ja käivitab iga leitu puhul reeglipõhise analüüsi. Kui tingimus oli väär, siis käivitataks samuti reeglipõhine analüüs, aga ainult ühe tähemärgi põhiselt. Lõpetuseks tagastatakse väljakutsujale tulemus, mis sisaldab analüüsi käigus saadud andmeid.

Põhjalikum kirjeldus reeglitest, nende vajalikkusest ja toimimisest, tuleb käesoleva peatüki järgnevas kahes alampeatükis. Esimeses kirjeldatakse neid reegleid ja tingimusi, mida algoritm peaks alati kontrollima ja vajadusel täitma. Teises tuuakse välja kõik võimalikud reeglid, mis määravad algoritmi väljundi ehk pildil oleva kujundi või kujundite kohta käivate parameetrite väärtused. Arvutatakse ainult need parameetrid, mis reeglites aktiivsetena määratud on.

5.3 Kohustuslikud reeglid

Algoritm eelduseks on, et ta saab sisendiks pildi ning reeglite kogumi, mille alusel toimub edasine tegevus. Esimene grupp reegleid on kohustuslikud reeglid, mis on kindlasti sisendis määratud ning mis kõik on omavahel seotud. Käesolev peatükk annab ülevaate kõigist nendest.

5.3.1 Tumeda piksli definitsioon

Tuleb arvestades võimalusega, et erinevatel piltidel võib olla taustaks nii valge värv kui kirju taust ning pildil olev kujutis ei ole täiesti must vaid lihtsalt tume. Seetõttu on vaja identifikaatorit, mille alusel saaks määrata, kas töötluses olev piksel on tume või hele. Selle saab algoritm sisendist ja võimalikud väärtused on 0 kuni 765-ni. Arvutatakse see selliselt, et liidetakse kokku töötluses oleva piksli punase, rohelise ja sinise värvi väärtus.

Kuna piltidel üldjuhul täiesti tumedat värvi ei eksisteeri (punase, rohelise ja sinise värvuse väärtus kõigil 0), siis reaalne reegli väärtus on kompromiss vastavalt olukorrale ja kindel piir puudub. Täpne sobiv väärtus selgub juba rakenduse arenduse ja testimise käigus, jäädes suure tõenäosusega 100 ja 200 vahele. Järgnevas tabel 1 on välja toodud mõned värvid ja nende koodid illustreerimaks kirjeldatud numbreid ja värvide väärtusi.

Tabel 1 Näitlikustav värvide väärtuste tabel

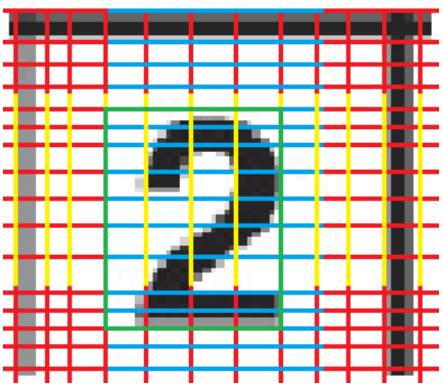
Värv	Punase osakaal	Rohelise osakaal	Sinise osakaal	Väärtus kokku
	0	0	0	0
	32	32	32	96
	48	0	0	48
	104	0	0	104
	64	64	64	192
	104	104	104	312
	160	160	160	480
	255	255	255	765

5.3.2 Pildi äärise olemasolu

Magistritöös kirjeldatud probleemis oli vajadus lahenduse järgi, mis töötaks ka sellisel juhul kui andmed asuvad tabelina paigutatult mitmel real. Sellisel juhul on aga tihti tähemärgid üksteisest millegagi eraldatud, näiteks Sudoku [14] välja puhul. Enamasti on eraldajaks kas tavaline joon või lihtsalt punktiirjoon. Teine kohustuslik reegel näitabki, kas algoritm peab selle võimalusega arvestama või mitte. See on oluline selle jaoks, et algoritm ei arvestaks tumedate pikslitena neid, mis ei ole kujundi osad, vaid on pildi ääres eraldajaks. Need ei tohi arvutustes arvesse minna. Samuti oleks äärises olevad tumedad pikslid segavaks faktoriks otsides pildilt esimest rida ja veergu, kust hakkavad kujundi tumedad pikslid. Võimalik probleeme tekitav olukord on näha jooniselt 2 mis on samas ka näide potentsiaalsest algoritmi sisendpildist.

Joonis 2 Äärisega sisend pildi analüüsil

Probleemi lahendamiseks oli mitu võimalust, millel oma positiivsed ja negatiivsed küljed. Kõige lihtsam oleks eeldada, et soovitav kujund asub pildi keskel ja üldse mitte töödelda näiteks 15% pildi äärtel olevaid alasid. See ei ole aga kõige parem lahendus, kuna ei saa kindel olla, et just see 15% õige eraldaja oleks. Teine võimalus oleks kasutada *Hough Transform* algoritmi [15], mis on levinud viis pildituvastuse tarkvarades joonte ja kurvide tuvastamiseks. Arvestades asjaolusid, et magistritöö eesmärgiks oli välja töötada võimalikult lihtne ja efektiivne lahendus, siis valik osutus uue meetodi väljatöötamisele, mis on väljendatud joonisel 3 ning millele järgneb selle kirjeldus.



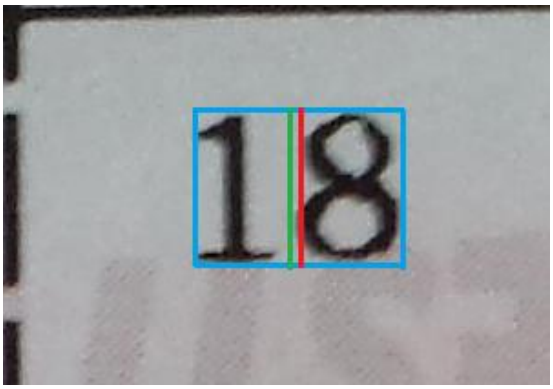
Joonis 3 Äärisega pildilt kujundi leidmine

Uus meetod eeldab, et sisendiks saadaval pildil olev kujund ei paikne täiesti ääres ja äärises oleva võimaliku eraldajaga on selge vahe. Toimib see selliselt, et pilti töödeldakse horisontaalselt ja vertikaalselt, mida näitavad joonise punased jooned. Otsitakse rida ja veergu, mis asuvad pildi keskpunktile võimalikult lähedal ja kus on veel kõik heledad pikslid. Sealjuures kontrollitakse ainult 50% (testimise ja realisatsiooni käigus võib see muutuda) keskmist ala, et sektorisse ei langeks äärises olevad jooned. Seda 50% väljendavad vertikaalselt sinised ja horisontaalselt kollased jooned. Sellise töötlemise tulemusena on võimalik leida lõpuks ristkülik, mille sisse oluline kujund jääb, mida saab edasi analüüsida. Joonisel 3 on see ristkülik rohelisega eraldatud.

5.3.3 Võimalike tähemärkide arv

Kolmas algoritmi toimimisele väga oluline sisend on informatsioon võimalike tähemärkide arvu kohta. Kuna lõpus käivitatav analüüs on ühe tähemärgi põhine, siis tuleb enne seda selgeks teha, mitu tähemärki pildil esineb ning need eraldada, arvutades eraldi ka kõigile tumedate pikslite arvu ja esimesed ning viimased tumedad read pildi pikslimaatriksis.

Esialgses realisatsioonis on plaanis toetada maksimaalselt kahte võimaliku tähemärki ja selle põhjal on valitud ka sobiv meetod tähemärkide eraldamiseks. Enne kui hakatakse tähemärkide võimalikku arvu kontrollima on juba toimunud töötlus, mis on arvestanud võimaliku äärisega ja leidnud pildi keskel oleva ala, kus asuvad tumedad pikslid, see on joonisel 4 sinisega märgitud. Edasi tähemärkide eraldamiseks hakkab algoritm selle ala keskelt (punane joon) liikuma kordamööda mõlema ääre poole kuni leiab veeru, kus on kõik heledad pikslid. Selle informatsiooni põhjal saab kujundid eraldada ja leida neile eraldi tumedate pikslite arvu ja teised parameetrid. Kui selles piirkonnas (joonisel 4 sinine ala) ei leitud eraldavat veergu, siis järelikult on pildil ainult üks tähemärk ja edasine käitumine toimub seda arvestades.



Joonis 4 Pildil olevate tähemärkide eraldamine üksteisest

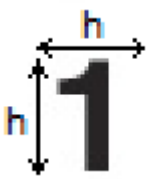
5.4 Analüüsi reeglid

Teine grupp sisendreegleid algoritmile on mõeldud selle jaoks, et otsustada mis analüüsid on vaja läbi viia. Piiramine läbi reeglite on oluline, et ei tehtaks üleliigset tööd, mis viib kiirust alla. Eelnevalt algoritmis leitud – esimene rida, viimane rida, esimene veerg ning viimane veerg tumeda piksliga – olid olulised analüüsi jaoks. Edasine piltide analüüs toimibki ainult selle ristkülikuks oleva ala sees, mis esimesed ja viimased tumedad pikslid moodustavad. Käesolevas alampeatükis kirjeldatavate reeglite põhjal annab algoritm lõpuks väljundi, kus on analüüsi tulemused.

Tulemused on protsentuaalse, kuna sellisel viisil on võimalik saavutada täpsem tulemus. Kui näiteks on vaja analüüsida kahte erinevat pilti, kus on peal sama kujutis, siis võib olla üks tehtud kaameraga, mis teeb foto kus on kolm korda rohkem piksleid kui teisel. Seetõttu ei oleks ainult tumedate pikslite võrdlemine sobiv lahendus ja annaks erinevate seademetega väga erinevaid tulemusi. Protsentides tulemuse arvutamine võimaldab sellised olukorrad üldjoontes võrdsustada, jäävad vaid pildi kvaliteedist ja pikslite jaotamisest tekkivad vead, mis aga on väiksed ning võib lugeda loomuliku eksimusprotsendi alla.

5.4.1 Tumedate pikslite osakaal

Esimene parameeter, mida algoritm suudab leida, on tumedate pikslite osakaal pildil. Selle jaoks on oluline tumedate pikslite loenduri arv ning see, milline on esimene ja viimane rida tumeda piksliga. Viimase info põhjal on võimalik arvutada kujundi kõrgus, mis omakorda määratakse ka pildi laiuseks nagu näha joonisel 5 kui tähis h tähendab nii kõrgust kui laiust. See on võrreldes teiste reeglitega erand, teistel juhtudel võetakse laiuseks kujundi tegelik laius. Antud juhul sai selline valik tehtud, kuna mõeldes potentsiaalsetele sisendpiltidele, siis kõigil neil olevatel kujunditel on kõrgus alati suurem kui laius. Ning et saaks võrrelda erinevaid kujundeid, siis on mõistlik võtta aluseks ruudu kujuline taust. Seega lõplik tumedate pikslite protsentuaalne osakaal leitakse, kui jagada tumedate pikslite arv kõrguse (võrdub numbriga, mitu pikslit mahub selle ala sisse) ruuduga ja korrutada sajaga.

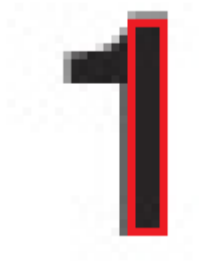


Joonis 5 Tumedate pikslite osakaalu leidmine

5.4.2 Tumedate pikslite osakaal 40% paremas ääres

Teine parameeter, mida algoritm võimaldab mõõta, on tumedate pikslite osakaal 40% pildi kujutise paremas ääres. Võrreldes eelnevalt kirjeldatud kõikide tumedate pikslite osakaaluga on siin käitumine erinev. Pildi laiuseks võetakse ala, mis asub esimesest veerust tumeda piksliga kuni viimase veeruni, millel on tume piksel. Mitte aga enam kujundi kõrguse väärtus nagu oli eelneva reegli puhul.

Eesmärk on saada protsentuaalne väärtus kujundil olevate tumedate pikslite paiknemisest paremas ääres. Joonisel 6 oleval kujundil on selleks punasega eraldatud piirkond. See võimaldab näiteks number ühte teistest tähemärkidest eraldada, sest teistel on tumedate pikslite jaotus ühtlasemalt laiali.



Joonis 6 Tumedad pikslid kujundi 40% paremas ääres

5.4.3 Tumedate pikslite osakaal 40% vasakul ääres

Leidmaks tumedate pikslite osakaalu 40% vasakul ääres toimitakse sarnaselt eelnevale reeglile. Algselt leitakse kujundi laius ning seejärel leitakse kujundi 40% vasakus äärse olevate tumedate pikslite osakaal kõigist tumedatest pikslitest. Joonisel 7 on näitena punasega välja toodud uuritav kujundi piirkond.

Joonis 7 Tumedad pikslid kujundi 40% vasakul ääres

5.4.4 Tumedate pikslite osakaal 40% keskel

Tumedate pikslite osakaal 40% keskel on kolmas mõõt, mis leiab analüüsi tulemuste jaoks tumedate pikslite osakaalu vertikaalselt. Kuna eelnevad reeglid olid vastavalt 40% paremalt ja vasakult, siis võiks eeldada, et tervikpildi saamiseks arvestatakse nüüd ainult allesjäänud osa. Siiski tuleb arvestades seda, et erinevate tähtede ja muude kujundite formaadid võivad erineda või olla veidi viltu. Seetõttu on mõistlik arvestada varieerivusega ja leida piksleid sama protsentuaalse laiusega piirkonnas kui eelnevalt, ainult et nüüd keskelt. Joonisel 8 on näitena punasega eraldatud reegli jaoks oluline piirkond.

Joonis 8 Tumedad pikslid kujundi 40% keskel

5.4.5 Tumedate pikslite osakaal horisontaalselt 20% ülemises osas

Kui eelnevad kolm analüüsi tulemust leiti otsides tumedate pikslite osakaalu vertikaalselt, siis käesolevas punktis kirjeldatav reegel ja järgnevad kaks teevad seda horisontaalselt. Antud reegli mõte on leida tumedate pikslite osakaal 20% ülemises ääres võrreldes kogu kujundi tumedate pikslite arvuga. Protsendiks on võetud 20, kuna uurides erinevates stiilides olevaid tähemärke, paistab selgelt välja, et kujundid on üldjuhul pikkupidi välja venitatud ja laiuti rohkem kokku surutud. Seega analüüsi jaoks olulise informatsiooni saamiseks piisab 20% ülemisest osast. Joonisel 9 on näitena punasega eraldatud kõnealune piirkond.

Joonis 9 Tumedate pikslite osakaal horisontaalselt 20% üleval

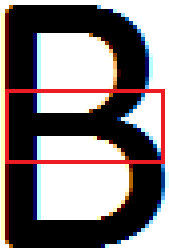
5.4.6 Tumedate pikslite osakaal horisontaalselt 20% alumises osas

Teine reegel, mis toimib horisontaalselt, on tumedate pikslite osakaalu leidmine 20% alumises osas. Toimimiselt on see väga sarnane eelnevale reeglile, erinevus on ainult selles, et kui eelnevalt analüüsiti ülemist kujundi osa, siis nüüd alumist. Joonisel 10 on näitena punasega eraldatud kogu pildi tumedate pikslitega võrreldav piirkond.

Joonis 10 Tumedate pikslite osakaal horisontaalselt 20% all osas

5.4.7 Tumedate pikslite osakaal horisontaalselt 30% keskmises osas

Tumedate pikslite osakaal horisontaalselt 30% keskmises osas on sarnane eelnevatele reeglitele. Erinevus seisneb selles, et nüüd on uuritav piirkond kujundi keskosas ning ulatuseks on võetud 30%. Joonisel 11 on näitena punasega eraldatud horisontaalselt kujundi 30% keskmine osa.



Joonis 11 Tumedate pikslite osakaal horisontaalselt 30% keskmises osas

5.5 Algoritmi realisatsioon

Selleks, et kirjeldatud algoritmist ja pseudokoodist saaks osa töötavast programmist, tuleb see programmeerimiskeeles realiseerida ja seejärel põhjalikult testida. Lõputöö autor otsustas algoritmi realisatsiooni programmeerida C++ programmeerimiskeeles, kuna autoril on sellega

olemas kogemus ja C++ kompileeritud programmikoodi saab integreerida töötama enamuse populaarsete keskkondade ja programmeerimiskeeltega. Oluline on just välja toodud teine aspekt, kuna lõputöö eesmärgiks on välja töötada lahendus, mis töötaks nutitelefonidel. Ja nutitelefonidel ongi palju erinevaid operatsioonisüsteeme, millest kolm enim levinud populaarsemat.

Realisatsiooni käigus selgus, et reaalne lahenduses tuleb veidi erinev, kui on pseudokoodis kirjapandu. See aga oli ette aimatav, kuna algne algoritm oligi kirjeldamaks üldist tegevuste jada. Minnes detailsemaks, tulevad uued piirangud ja võimalused, millega arvestama peab. Järgnevates lõikudes on välja toodud need asjaolud, millega tuli C++ koodi kirjutades arvestada, ning mis on täienduseks varasemas alampeatükis kirjeldatud pseudokoodile.

Uurides milline on erinevate pildiformaatide struktuur sai selgeks esimene piirang, mis realisatsioonis tuleb sisse tuua. Selleks, et hoida programmi võimalikult kompaktsena ja välistada probleeme eoses, tuli piirata sisendiks antava pildi formaati. Nõudeks sai, et sisendiks tulev pilt peab olema BMP [16] formaadis ja bitisügavusega 24. Tänu sellele saab kõiki pilte töödelda ühtemoodi ning ei pea arendama kümneid erinevaid realisatsioone erinevatele formaatidele. Bitisügavus 24 tähendab seda, et piksli värvid on RGB kujul ning iga piksli värvi jaoks on üks bait ehk 8 bitti ruumi. Seda piirangut sisse tuues sai arvestavaks ka see, et kõigil tuntud raamistikel on tugi, mis võimaldab kiirelt ja mugavalt pilte erinevatesse formaatidesse konverteerida. Seega lihtsam ja kindlam on lõpprakenduses teha enne pildi formaadi kontroll ja vajadusel muutus, selle asemel, et pildi analüüsis katta kõik võimalikud erinevad sisendpildi formaadid.

Otsustades, millises formaadis on sisendiks olev pilt, tuli teha ka ülejäänud töötlus vastavalt sellele. Arvestada tuli BMP faili struktuuri [17] [18], kuidas teha kindlaks pildi pikkust ja laiust, ning mille alusel lugeda värvikoode. Programmi jaoks vaja läinud informatsioon on toodud välja tabelis 2, kus vasakul veerus on kirjas, mis infoga on tegu ja paremas veerus asukoht, mitmendal baidil BMP failis see informatsioon asub.

Tabel 2 Olulise info asukoht BMP faili struktuuris

Andmetüüp	**Asukoht BMP struktuuris**
Pildi laius	18-21 bait

Pildi kõrgus	22-25 bait
Värvide alguspunkt (mitmendast baidist hakkab pikslite värviinfo)	10-11 bait

Lisaks tabelis kirjeldatule tuli veel arvestada seda, et 24 bitisel BMP faili iga pildirea lõpus võib asuda nullidega täidetud baite, et kogu pildirea baitide pikkus oleks kindlasti neljaga jagatav arv. [17]

5.6 Testimine

Kuna algoritm realiseeriti C++ programmeerimiskeeles ja kasutades Microsoft Visual Studio 2010 [20], siis sai algoritm arendatud klassiteegi projektina. Testimise jaoks loodi kõrvale teine projekt, lihtne konsoolirakendus. Seda on kasutatud testimiseks, andes programmile sisendiks erinevaid pilte, mille töötlemist ja analüüsi tulemuse õigsust on hinnatud. Testimise võib iseloomult jagada kaheks, esimeses osas kontrolliti kas programm suudab tuvastada mõõtmed ja toimida vastavalt kohustuslikele reeglitele ning teises osas kontrolliti analüüsi tulemuste paikapidavust.

5.6.1 Kohustuslike reeglite testimine

Esimeses osas tuli seega alustada kohustuslike reeglite kontrollimisega, et veenduda programmi õiges toimimises. Selleks, et testida piksli tumeduse kontrolli, kasutas magistritöö autor Gimp 2 [21] pilditöötlusprogrammi, luues uue pildi suurusega 10*10 pikslit. Seejärel sai kolm pikslit täidetud värviga mille RGB väärtuste summa on null. Samamoodi veel kaks kolmest lõiku vastavalt RGB summaga 120 ja 240. Lõpuks joonistati selle ümber ruut, mille pikslid värviti värviga, mille RGB summa oli 300. Magistritöö lisa 3 all oleva esimene testsisend näitab seda piltlikult suurendatud kujul, originaalsisendile on lisatud on veel lisaks ääris ja katkendliku joonega ruut näitamaks kui suure ala 1 piksel hõivab.

Kirjeldatud pilt anti analüüsivale programmile sisendiks ja väljundiks kuvati konsoolil tulemused informatsiooniga, kus oli kirjas leitud tumedate pikslite arv, pildi kõrgus ja laius ning koordinaatide piirkond kus ristküliku sees tume kujund asub. Mõõtkava osas vastas tulemus ootustele ehk nii pildi kõrgus kui laius tuvastati programmi poolt ja kuvati number 10 välja. Muude parameetrite testimiseks tuli muuta tumeda piksli väärtust ehk piksli RGB

väärtuste summat. Testimise tulemused on näha tabel 3, kus esimene rida näitab mis anti programmile sisendväärtuseks, alates mis RGB väärtuste summast võib pikslit lugeda tumedaks. Võrreldi tumedate pikslite koguarvu ja piirkonda, kus tumedad pikslid asuvad. See on tabelis väljendatud kujul *(x,y) (x,y)*, kus esimene koordinaat näitab ristküliku vasakut ülemist nurka ja teine alumist paremat nurka, mille sees tumedad pikslid paiknevad. Vasak ülemine nurk tähistab punkti (1,1) ja iga piksel suurendab väärtust ühe võrra. Lõpptulemuseks tuli see, et väljundid vastasid ootustele ehk testimine oli edukas.

Tabel 3 Tumedate pikslite leidmise reegli testimise tulemused

Tumeda piksli väärtus	**Tumedaid piksleid (oodatud tulemus)**	**Tumedaid piksleid (reaalne tulemus)**	**Tume ala (oodatud tulemus)**	**Tume ala (reaalne tulemus)**
10	3	3	(3:4) (5:4)	(3:4) (5:4)
125	6	6	(3:4) (5:5)	(3:4) (5:5)
250	9	9	(3:4) (5:6)	(3:4) (5:6)
301	25	25	(2:3) (6:7)	(2:3) (6:7)

Teine reegel oli selle jaoks, et teha kindlaks äärise olemasolu pildid ning selle olemasolul puhul leida kujund pildi keskelt ning teha ainult selle põhjal analüütilised arvutused. Veendumaks, et see ka nii toimib sai võetud üks pilt, millel on ääris ja tehtud sellest koopia, mida sai muudetud selliselt, et ääriseks olevad pikslid sai valgeks värvitud. Vastavad pildid on näha ka antud töö lisa 3 alt, testsisend 2.1 ja 2.2. Käivitades test konsoolirakenduse kord andes sisendiks ühe pildi ja siis teise, leiti mõlemalt pildilt võrdselt 645 tumedat pikslit ehk testimine oli edukas, esimese pildi äärises olevad tumedaid pikselid ei arvestatud tulemuses.

Lisaks eelnevale tuli veel vajadusel kontrollida, kas pildil on üks või kaks tähemärki. Selle testimiseks lülitati sisse reegel, et pildil võib esineda kaks tähemärki. Esimesena testiti juba eelnevalt mainitud lõputöö lisa 3 olevat testsisendit 2.1 ja 2.2. Mõlemad andsid vastuseks, et pildil on ainult üks tähemärk, mis on õige tulemus. Seejärel anti programmile sisendiks lõputöö lisa 3 all olev testsisend 3.1, kus on kaks tähemärk. Tulemuseks leitigi kaks tähemärki ja eraldajaks 110 veerg. Vaadates pilditöötlusprogrammiga GIMP [21] pilti, võib tõdeda, et 110 veerg asub tähemärkide vahel ja sobib eraldajaks. See tähendab, et edasine analüüs kummalegi poole tehakse eraldi. Kuna aga eelnevas testis asuski pildi keskpunkt kahe tähemärgi vahel, siis sai tehtud veel lisakatse testsisendiga 3.2. Sellel testsisendil on eelmisest

pildist lõigatud vasakult poolt osa ära, seega keskpunkt jääb tähemärgi 5 peale. Testi eesmärk on kontrollida, kas sellisel juhul leitakse samuti kaks tähemärki ja eraldav veerg nende vahel. Tulemuseks oli edukas test ehk pildilt leiti kaks tähemärki ja eraldajaks 84 veerg, mis pilditöötluse programmiga järgi vaadates tõesti selleks sobib.

5.6.2 Analüüsi reeglite testimine

Teises testimise osas kontrollitakse analüüsi reeglite toimimist, et kas leitud analüüsi tulemused vastavad reaalsusele. Selle jaoks joonistas magistritöö autor pilditöötlusprogrammi GIMP [21] abil ristküliku mõõtmetes 100*100 pikslit kasutades joonelaiust 10 pikslit. Selle paremalt alt nurgast kustutati 50% nii üles kui vasakule ära ja järgi jäi joonisel 12 paiknev kujund.

Joonis 12 Kujund analüüsi reeglite testimiseks

Testimaks, et analüüsiv programm töötab õigesti, sai käsitsi arvutatud, mis peaks tulema analüüsi tulemuseks. Seejärel sai käivitatud programm, et võrrelda tulemusi käsitsi välja arvutatuga. Lõpptulemused on näha tabelist 4, kus tulemused on ümardatud kahe kohani peale koma. Algselt esines küll ka erinevusi, mis oli tingitud programmis tehtud ümardustest, aga need sai ära parandatud ja viimane käivitatud test andis juba samad tulemused käsitsi arvutatuga.

Tabel 4 Analüüsi reeglite testi tulemused

Reegel	**Käsitsi arvutatud tulemus**	**Programmi tulemus**
Tumedate pikslite osakaal	27 %	27 %
Osakaal 40% paremal ääres	29,62 %	29,62 %
Osakaal 40% vasakul ääres	59,25 %	59,25 %
Osakaal 40% keskel	22,22 %	22,22 %
Osakaal 20% vertikaalselt üleval	44,44 %	44,44 %
Osakaal 20% vertikaalselt all	22,22 %	22,22 %
Osakaal 30% vertikaalselt keskel	16,66 %	16,66 %

5.7 Lõpptulemus

Käesolevas peatükis kirjeldati uue algoritmi nõuded ja selle toimimise loogika pseudokoodis. Seejärel realiseeriti see programmiks, mille toimimist testiti eri tüüpi sisenditega. Selle tulemusena valmis eraldiseisev tarkvara komponent, mida saab kasutada piltide analüüsiks. Andes sisendiks pildi ja määrates reeglites mida analüüsida tuleb, tagastatakse pildil oleva kujundi kohta tehtud analüüsi tulemused. Seda kasutav programm saaks selle abil näiteks koostada sarnase tabeli nagu tabel 5 ning kasutada seda uute sisendpiltide puhul võrdluseks tuvastamaks tähemärke.

Tabel 5 Illustreeriv tabel pildi analüüsi tulemustega tutvumiseks

Tähemärk	Pilt	Tumedate pikslite %	Tumedad pikslid 40% paremas ääres	Reegel 3	Reegel 4	...
1	[pilt]	21.56 %	60 %	...	...	...
0	[pilt]	28.75 %	35 %	...	...	...

Selleks, et kasutada C++ arendatud klassiteeki teiste programmeerimiskeeltega koos, on vaja klassiteek ümber konverteerida ja kapseldada sobivasse vormi. Selle jaoks saab kasutada tarkvara SWIG [21], mis on mõeldud C ja C++ kirjutatud programmide ühendamiseks kõrgtaseme programmeerimiskeeltega. See on laialdaselt kasutuses, kuna võimaldab vältida koodi dubleerimist, kui tuumloogika arendatakse valmis kesktaseme programmeerimiskeeles ja kasutajaliidesega ning äriloogikaga seotud tegevused ülemises tasemes. Kasutatakse seda näiteks paljude nutitelefoni mängude arendamisel. Täpsemalt tuleb sellest juttu järgnevates peatükkides, kus seda vaja läheb.

6. Ekspertsüsteemi arendus

Magistritöö teiseks põhieesmärgiks sai püstitatud ekspertsüsteemi arendus, mis kasutab eelnevas peatükis välja töötatud programmi ja suudab selle abil tuvastada piiratud tingimustes piltidel olevaid tähemärke. Lisaks teostab see statistilist analüüsi tuvastamisele edastatud piltide kohta, mis võimaldab märgata tähemärkide eristuvaid omadusi. Seda informatsiooni saab kasutada otsuste tegemiseks eraldiseisvas rakenduses, kus on samuti realiseeritud eelnevas peatükis välja töötatud algoritm. Sellisel juhul võrreldakse uue pildi tuvastuse andmeid mallpildi infoga. Käesolevas peatükis antakse ülevaade ekspertsüsteemi arendusest, selle sisemisest toimimisest ning kasutegurist, mida valmis lahendus pakub.

6.1 Arendusvahendite valik

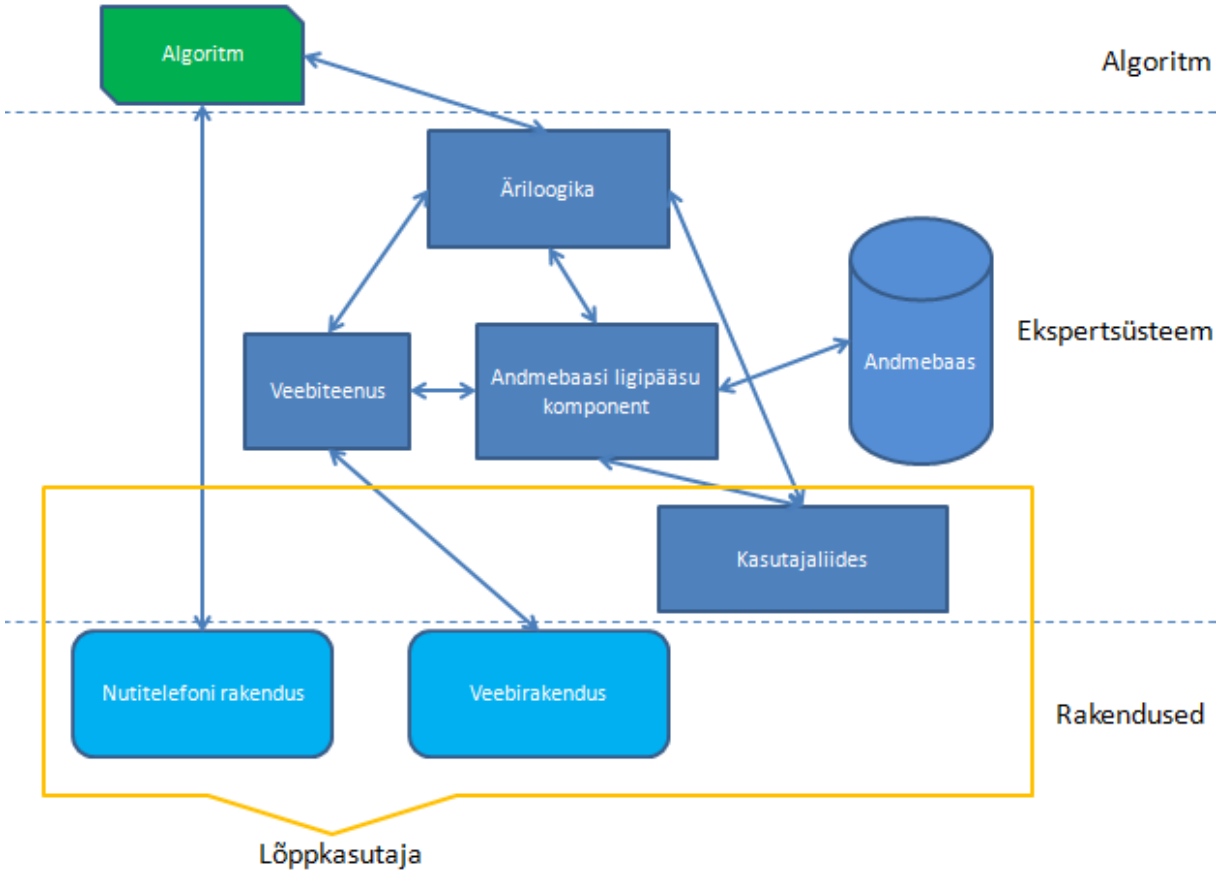
Ekspertsüsteemi arendusvahendite valikul on lähtutud asjaolust, et arendada tuleb erinevat tüüpi komponente, mida peab saama omavahel kergelt integreerida. Erinevateks komponentideks on veebiteenus, andmebaasi ligipääsu kiht ning veebirakendus, mis on pikemalt kirjeldatud järgmises alampeatükis, lahenduse arhitektuuri all. Seetõttu osutus loogiliseks valikuks Microsoft Visual Studio 2010 [20] arenduskeskkond ning programmeerimiskeeleks C#. Selle valiku kasuks oli ka see asjaolu, et magistritöö autoril on juba mitmeaastane töökogemus nii välja toodud arenduskeskkonna kui C# programmeerimiskeelega ja seega on seadistamine kiirem ning arenduses saab keskenduda vajaliku funktsionaalsuse loomisele, mitte keskkonna omaduste tundmaõppimisele. Microsoft Visual Studiost [20] on kasutusel 2010 versioon seetõttu, et lõputöö autoril oli olemas selle litsentseeritud versioon. Uuemate versioonidega ei oleks lisaväärtust tulnud, arendusprotsess oleks vaid veidi mugavam olnud.

Kuna süsteemi arenduskeskkonna osas sai tehtud valik Microsofti toote kasuks, siis andmebaasiks valik osutus samuti Microsoft SQL Serveri [23] kasuks. Seda on kerge teiste komponentidega ühildada ja lihtsa vaevaga saab genereerida andmebaasi tabelitest klassid. Andmeklassid võimaldavad objektide ja meetodite abil suhelda andmebaasiga, ilma SQL kasutamata. See muudab arenduse mugavaks ja kiireks. Lisaks mainitud vahenditele on arendamisel abiks SoapUI [24], mis võimaldab testida veebiteenust. Selle abil saab kohe

kiirelt ilma klientrakendust omamata tagasisidet, kas teenus üldse töötab ning seejärel juba testsisenditega tagastatavate andmete õigsuses veenduda.

6.2 Lahenduse arhitektuur

Arhitektuuriliselt saab ekspertsüsteemi jagada neljaks osaks, mida on tehtud ka realisatsioonis. Ühe Microsoft Visual Studio 2010 [20] lahenduse alla on tehtud neli erinevat projekti. Esimene komponent on vastutav ühiste klasside ja funktsionaalsuse eest ning suhtlusega pilte analüüsiva algoritmi realisatsiooniga. Teine komponent tegeleb andmebaasiga suhtlemisega ehk kõik sisestamise, muutmise ja kustutamise käsud on realiseeritud selles osas. Järele jäävad veebirakendus ja veebiteenus, millest esimene võimaldab näha analüüsi tulemusi ja seadistada pildituvastust ning teine tehtud seadistuste põhjal teha päringuid piltide tekstituvastuseks. Ülevaatlik pilt arhitektuurist on näha joonisel 13, kus on lisaks näidatud veel algoritmi paiknemine ja lõpprakendused üldises vaates.



Joonis 13 Lahenduse arhitektuur

Esimene projekt on realiseeritud klassiteegina, mis sobib kõige paremini baasfunktsionaalsuse eraldamiseks. Selle hulgas on defineeritud kõik baasklassid, mida võib teistes projektides vaja minna. Samuti on ühine loogika, näiteks baidimassiivi pildiks konverteerimine ja vastupidi, realiseeritud selles osas, et mujal ei oleks koodikordusi. Kõige olulisem ülesanne on aga suhtlemine magistritöö põhieesmärgiks olnud algoritmi realisatsiooniga ehk ekspertsüsteemi kõik päringud piltide analüüsiks käivad läbi selle klassiteegi.

Teine projekt on realiseeritud samuti klassiteegina, kuid võrreldes eelnevaga on oluline erinevus see, et lisatud on veel andmebaas ja selle põhjal genereeritud andmeklassid. Alustada tuli aga andmemudeli disainist, mis oleks abiks lahendamaks magistritöö algul püstitatud probleemi. Seega tuli mõelda, kuidas hoida piltide andmeid ja kasutaja poolt sellelt välja selekteeritud tähemärke ning nendega seotud analüüsi tulemusi. Lisaks sellele on arvestatud seda, et ekspertsüsteemi veebirakendust saaks avalikust veebist ligipääsetavaks teha ehk disainis on arvestatud kasutajate ning rollidega. Valminud andmebaasi disaini mudel on nähtav käesoleva töö lisa 4 alt. See tuli üsna lihtne, kuid täidab kõik esialgsed vajadused. Mudeli põhjal loodi vastavad tabelid andmebaasi ning genereeriti nende põhjal andmeklassid kasutades standardseid Microsoft Visual Studio 2010 [20] töövahendeid. Andmeklassidest on palju abi, kuna võimaldavad arendajale kiiret ja mugavat ligipääsu andmebaasi, nii andmeid uuendama kui pärima. Keerulisemate päringute puhul võib küll jõudlus kehvemaks muutuda, kuid antud juhul andmemudel väike ja päringud lihtsad, seega ei ole seda ohtu. Probleemi tekkides võib keerulisemad päringud SQL abil koodis realiseerida.

Kolmandaks projektiks on veebirakendus, mis on arenduse poolest kõige mahukam. Kasutab ta küll teisi klassiteeke piltide analüüsi tulemuste pärimiseks ja andmebaasiga suhtluseks, kuid piltide üles laadimise funktsionaalsus ja tähemärkide selekteerimine jääb selle komponendi ülesandeks. Samuti tuli realiseerida kasutajate sisse logimine, olemasolevate piltide haldamine ja analüüsi tulemuste kuvamine ning erisuste välja toomine. Lisaks selekteeritud pildi alale ka API võtme genereerimine, kui veebirakenduse kasutaja soovib hiljem pildituvastust läbi veebiteenuse kasutada. Käesoleva magistritöö jaoks kõige olulisemad veebirakenduse kasutamise võimalused on välja toodud lõputöö punkti 6.4 all koos illustreerivate joonistega.

Neljandaks projektiks on veebiteenus, mis on arendatud kasutades WCF (Windows Communication Foundation) [25] raamistiku. See on üsna lihtsa ülesehitusega, on ainult üks meetod mida välised süsteemid saavad teenuses kasutada. Selle kasutamiseks on vaja aga teha kasutajal eeltööd eelnevalt kirjeldatud veebirakenduses. Tuleb üles laadida pildi mall,

selekteerida sealt huvi pakkuvad piirkonnad ning tähemärgid. Tähemärgi kujutised omakorda tuleb defineerida, määrates millele nad vastavad ja edastada analüüsiks. Kui see on tehtud saab valitud pildi piirkondade kohta API võtme genereerida. Hiljem saab veebiteenusele sisendiks anda uue pildi, mis struktuurilt sarnaneb mallpildiga ja teise sisendina anda võtme väärtused, mis piirkonna kohta infot tuvastada tuleb. Veebiteenus tagastab selle peale objektid pildituvastuse tulemustega.

6.3 Realiseeritud algoritmi kasutus

Nagu juba eelnevalt mainitud, siis realiseeritud algoritmi kasutab ekspertsüsteemi see osa, mis on realiseeritud klassiteegina ja omab lisaks analüüsi võimalustele ka teistele komponentidele ühiseid klasse ja meetodeid. C++ kirjutatud koodi aga ei saanud kohe C# projektis kasutusele võtta, vaid tuli enne ümber konverteerida sobivale kujule, selle jaoks kasutati ka juba lõputöös varem mainitud SWIG [22] vahendeid.

C++ koodi konverteerimisel sobivale kujule võeti eeskujuks õpetus [26], kus oli kirjeldatud, kuidas C++ kompileeritud teegis olevaid meetodeid välja kutsuda. Magistritöö lahenduse puhul toimis kõik õpetusele sarnaselt ja sujuvalt. Võrreldes viidatud materjalile oli erinevus vaid selles, et klassid ja meetodid ning funktsionaalsus oli vastavalt lõputöö sisule, mis kirjeldatud algoritmi peatüki all. Lisaks kuna analüüsiva meetodi sisendiks on baidimassiiv, siis tuli see liidese failis selget defineerida. Seega kui C++ projektis oli juba valmis funktsionaalsuse realisatsiooni klass ja selles kasutatud klasside ning meetodite kirjelduse fail, tuli lisada ainult liidese fail. Selle põhjal SWIG [22] genereeris C# programmeerimiskeeles ekspertsüsteemi jaoks sobivad liidesfailid. Järgnevalt on välja toodud kirjeldatud liidese fail:

```
%module ImgAnalyzerLibrary
%{
   #include "OcrHelper.h"
%}
%include "arrays_csharp.i"
%apply unsigned char INPUT[]  { unsigned char* bytes }
%include <windows.i>
%include "OcrHelper.h"
```

6.4 Graafiline kasutajaliides

Kolmandaks eraldiseisvaks komponendiks ekspertsüsteemis on varem lühidalt kirjeldatud veebirakendus, mis on realiseeritud ASP.NET raamistikus ja kasutades Razor süntaksi [27]. Arendamisel on arvestatud võimalusega, et see ei oleks kasutatav üksnes mitte ainult isiklikuks otstarbeks, vaid avaliku veebilehena. Sihtrühm on siiski piiratud tarkvara arendajatega, kellele saab pakkuda piltide analüüsi ning tekstituvastust piiratud tingimustes. Seetõttu on pealehel lühike teenuse tutvustus ning kasutaja registreerimisele viitav link ja sisselogimise vorm. Põhiline ja magistritöö kontekstis oluline funktsionaalsus on kasutatav peale süsteemi sisse logimist. Järgnevalt on lühidalt kirjeldatud võimalusi, mida ekspertsüsteemi graafiline kasutajaliides võimaldab. Ning nagu näha 51 leheküljel oleval jooniselt 14 ülalt, on samuti menüü jaotise aluseks:

1) Pealeht – avaneb peale sisse logimist. Kuvatakse informatsiooni, palju pildimalle antud kasutajal on ning kui palju on tehtud veebiteenuse kaudu päringuid mingi pildi piirkonna tuvastamiseks. Samuti võimalus tuua välja viimased muutused süsteemis, mis infoks süsteemi kasutajatele.

2) Piltide haldus – olemasolevate piltide haldus. Kui kasutaja on juba lisanud kolmandas punktis kirjeldatava võimaluse kaudu pildimalle ja määranud algseaded, siis siin saab neid vajaduse muuta ning lisada tähemärke, mida tuvastamisel arvestada tuleks. Samuti on selle menüüpunkti all võimalik genereerida võtme väärtusi pildi piirkondade kohta, et veebiteenuse päringute puhul sisend viia kokku sobivate seadete ja reeglitega. Selle valiku alt saab samuti kustutada pilti või pildipõhiseid seadistusi.

3) Uue pildi lisamine – võimalus laadida ekspertsüsteemi uus pilt. Alguses avaneb lihtne faili üles laadimise vorm. Kui pilt on üles laetud, siis saab selle pealt alasid ja tähemärgi kujutisi selekteerida ning määrata mis reeglid analüüsis rakendatakse. Pikemalt on erinevaid võimalusi kirjeldatud peale käesoleva loetelu lõppu ning visuaalset poolt saab näha jooniselt 14.

4) Analüüsija – näidatakse pildi analüüsi tulemusi. Kui pilt on üles laetud, tähemärgid ning alad selekteeritud ja reeglivalikud tehtud, siis saab selle peale analüüsi teostada. Antud menüüpunkti all näidatakse lõpptulemusi, mille võimalik väljund on näha 52 leheküljel olevalt jooniselt 15. Pikemalt on tulemuste formaat kirjeldatud käesoleva alampeatüki lõpus.

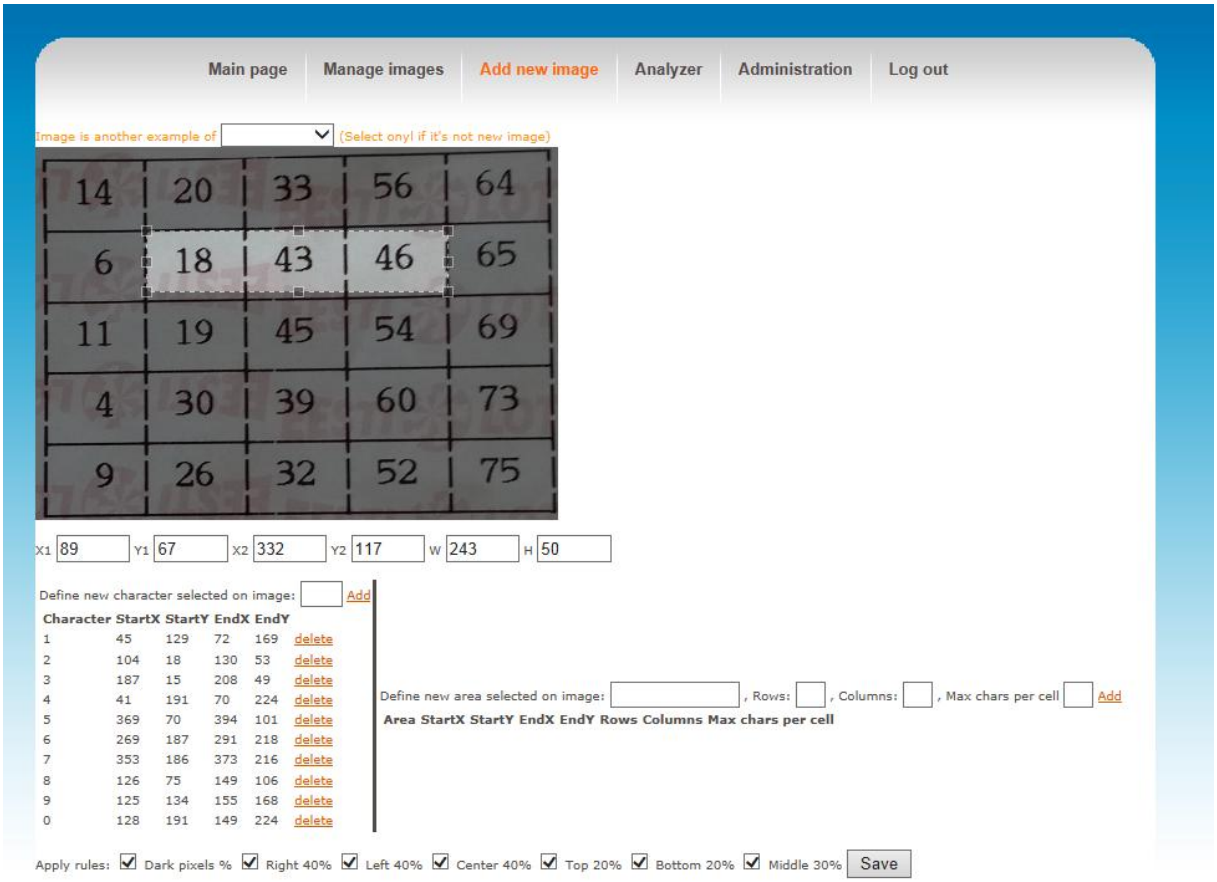
5) Administreerimine – kasutajate haldus. Valik mis on nähtav ja ligipääsetav ainult administraatori õigustes kasutajale. Saab näha kõiki süsteemis olevaid kasutajaid, viimast sisselogimise aega ja nendega seotud pilte. Lisaks kasutajate haldus ja statistika, kui palju on veebiteenuse kaudu piltide tuvastamiseks päringuid tehtud.

6) Süsteemist välja logimine – lõpetatakse kasutaja sessioon ja logitakse ekspertsüsteemist välja.

Ekspertsüsteemi graafilise kasutajaliidese üks olulisemaid osasid on uue pildi lisamine. Kui algul avaneb lihtne pildi üles laadimise vorm, siis peale üles laadimise protsessi avaneb keerukam vaade, mis on näha jooniselt 14. Liikudes ülalt alla, siis esimene valik, mida teha saab, on seotud pildimalli valimine. See funktsionaalsus on selle jaoks, et kui mallpildil ei ole esindatud kõiki tähemärke, millest ekspertsüsteem peaks teadlik olema, siis saab vajadusel sisestada süsteemi uusi pilte, mille põhjal kõik tähemärgid defineeritud saaks.

Järgneb pildi osa, milleks joonisel 14 on nutitelefoniga tehtud pilt Bingo loto [28] mänguväljast. Jooniselt on näha heledam ala tumedamal taustal. See on selekteeritud ala, mille koordinaadid ning kõrgus ja laius on kohe pildi all kuvatud. Seda ala saab kursoriga nihutada üles ja alla või selekteerida uus piirkond altpoolt. Realiseeritud on see *JCrop* [29] *jQuery* teeki kasutades. Selekteerimise esmane eesmärk on eri tähemärkide defineerimine, et määrata mis piirkonnas mingi tähemärk asub. Seda on tehtud joonisel 14 oleval pildil juba 10 erineva märgiga, mida on näha ka joonisel 15 oleva pildi veerus. Selle info põhjal toimub hilisem tähemärgipõhine analüüs. Teine selekteerimise eesmärk on piirkondade defineerimine. Kui näiteks sisendpildiks oleks alati kogu lotopilt, siis oleks võimalus seadistada, mis piirkonnas asub oluline numbrite osa. Antud juhul oleks aga õige määrata üksainus piirkond, milleks on kogu pilt ja ridade ning veergude arvuks valida 5. Selle informatsiooni põhjal oskab ekspertsüsteemi veebiteenus uusi sisendeid analüüsi jaoks tükkideks jagada ning tagastada tuvastamise tulemused 5*5 maatriksina.

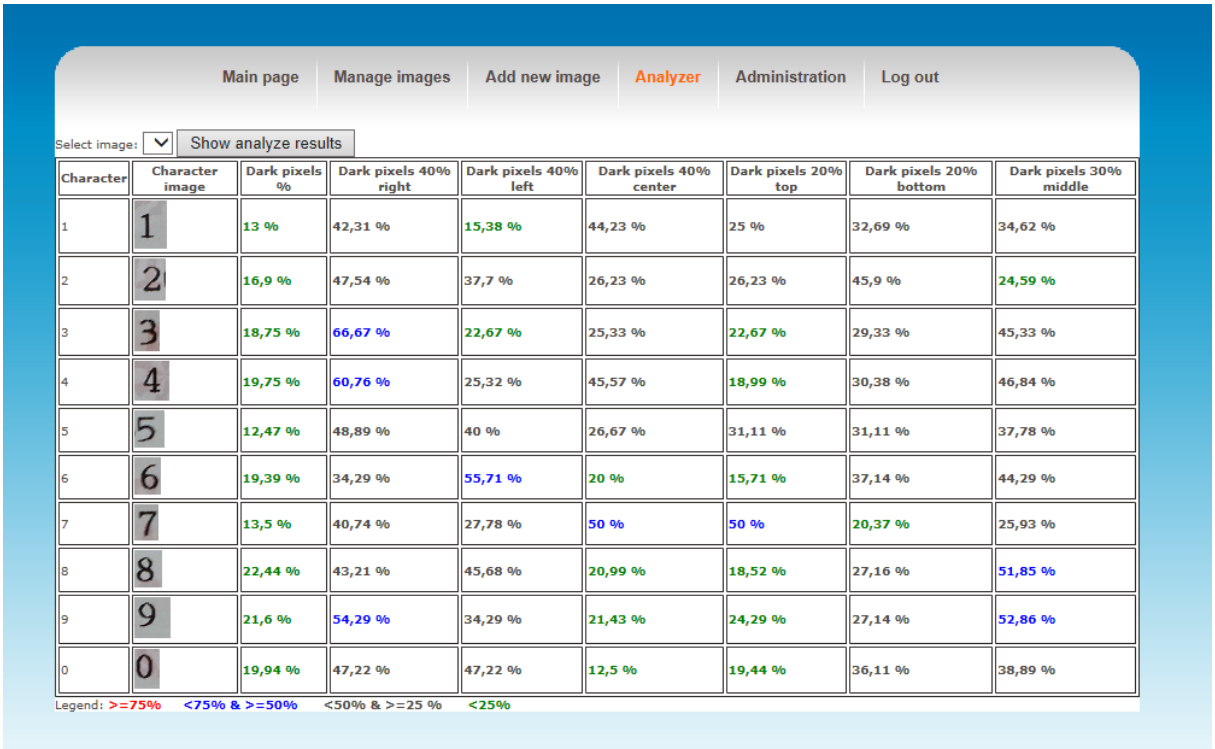
Viimane valik, mis tuleb teha enne uue pildi analüüsi saatmist, on reeglite märkimine. See määrab, mida järgnevalt analüüsi etapis rakendada tuleb. Vaikimisi on kõik valitud ehk analüüsi osas arvutatakse kõik parameetrid välja. Seda saab hiljem vajadusel piirata ehk kui selgub, et mingi reegel ei näita piisavalt hästi erisust ega mõjuta lõpptulemust, siis analüüsi kiiruse parandamiseks lülitatakse see välja.



Joonis 14 Uue pildi ekspertsüsteemi sisestamine

Teine veebirakenduse oluline komponent, mis samuti visuaalselt palju informatsiooni näitab, on analüüsi tulemuste lehekülg. Kui ekspertsüsteemi kasutaja on uue pildi seadistused ära teinud ja vajutab salvesta nupule, siis suunatakse ta edasi analüüsi tulemuste juurde, mis on näha jooniselt 15.

Kui vaikimisi suunatakse kasutaja analüüsi lehel äsja sisestatud pildi tulemusteni, siis võimalik on vaadata ka varasemalt sisestatud piltide analüüsi tulemusi, mis on sellel leheküljel esimene valik. Nagu jooniselt 15 näha, järgneb tabel tulemustega. Esimeses veerus on kasutaja poolt defineeritud tähemärk, teises selle tähemärgi jaoks selekteeritud pildi piirkond ning edasi tulevad reeglite põhjal tehtud analüüsi tulemused. Protsendid on kuvatud eri värvides, vastavalt legendile, et eristuvad omadused paremini välja tuleks.



Character	Character image	Dark pixels %	Dark pixels 40% right	Dark pixels 40% left	Dark pixels 40% center	Dark pixels 20% top	Dark pixels 20% bottom	Dark pixels 30% middle
1	1	13 %	42,31 %	15,38 %	44,23 %	25 %	32,69 %	34,62 %
2	2	16,9 %	47,54 %	37,7 %	26,23 %	26,23 %	45,9 %	24,59 %
3	3	18,75 %	66,67 %	22,67 %	25,33 %	22,67 %	29,33 %	45,33 %
4	4	19,75 %	60,76 %	25,32 %	45,57 %	18,99 %	30,38 %	46,84 %
5	5	12,47 %	48,89 %	40 %	26,67 %	31,11 %	31,11 %	37,78 %
6	6	19,39 %	34,29 %	55,71 %	20 %	15,71 %	37,14 %	44,29 %
7	7	13,5 %	40,74 %	27,78 %	50 %	50 %	20,37 %	25,93 %
8	8	22,44 %	43,21 %	45,68 %	20,99 %	18,52 %	27,16 %	51,85 %
9	9	21,6 %	54,29 %	34,29 %	21,43 %	24,29 %	27,14 %	52,86 %
0	0	19,94 %	47,22 %	47,22 %	12,5 %	19,44 %	36,11 %	38,89 %

Joonis 15 Pildi analüüsi tulemused

6.5 Ekspertsüsteemi kasutusvõimalused

Arendatud ekspertsüsteemi saab kasutada mitmel eesmärgil. Kui läheneda lõpprakenduse vaatenurgast, siis on võimalusi kaks. Esimene võimalus on kasutada seda kui analüüsivat töövahendit leidmaks erinevatele tähemärkidele iseloomulikke parameetreid ning saadud info põhjal märkide tuvastamine realiseerida lõpprakenduses. Teine võimalus on lisaks tähemärkide analüüsimisele määrata mallpildil huvipakkuvad piirkonnad ning hiljem kasutada ekspertsüsteemi veebiteenust piltide tekstituvastuseks. Järgnevalt on kirjeldatud mõlemad võimaluse, nende eelised ja puudused.

6.5.1 Abistav töövahend

Esimene võimalus on kasutada ekspertsüsteemi kui ainult analüüsivat ja abistavat töövahendit. Toimib see selliselt, et kui on vaja mingit tüüpi piltidelt tähemärke tuvastada, siis esiteks tuleb laadida üks näidispilt ekspertsüsteemi. Seejärel kasutajaliidese abil selekteerida sealt välja need tähemärgid, mida soovitakse lõpptulemusena tuvastada ning edastada need analüüsiks. Kui sellel ühel näidispildil olid tähemärgid vaid osaliselt esindatud, siis võib veel pilte üles laadida ja selekteerida sealt puuduolevad märgid, ning määrata, mis põhimalliga nad

seotud on. Seejärel saab vaadelda läbiviidud analüüsi tulemusi, kust leida mingitele tähemärkidele ainuomaseid tunnuseid ja iseloomustavate parameetrite väärtusi.

Kui analüüsi tulemused on teada, saab lõpprakenduses defineerida konstandid, millele omistada analüüsi käigus leitud väärtused. Uue pildi puhul, millel soovitakse teostada tähemärkide tuvastamist, käivitatakse algul rakendusse integreeritud analüüsiva algoritmi realisatsioon. Seejärel võrreldakse tulemusi defineeritud konstantidega, et leida täpseim vaste ning selle põhjal otsustada, mis tähemärgiga on tegu. Kuna magistritöös arendatud algoritm toetab kuni kahe märgi analüüsi korraga, siis tuleb rakenduse tasemel vajadusel teha pildi tükeldus sisendi sobivale kujule viimiseks.

Kasutades ekspertsüsteemi kui abistavat töövahendit on negatiivseks küljeks see, et analüüsi andmete põhjal tehtav tuvastamise otsus tuleb realiseerida alati lõpprakenduses. Lisaks peab pildi tükeldama, kui on võimalus, et seal on rohkem kui kaks tähemärki. Kuid on ka positiivseid külgi, miks magistritöö autor selle võimaluse välja pakkus. Tänu pildituvastuse realiseerimisele lõpprakenduses kaob vajadus reaalajas töötava veebiteenuse jaoks. Oluline on see just nutitelefonide rakendustes, kuna maailma kontekstis on interneti ühendus tihti piiratud või siis kallilt tasustatud. Kui aga rakendus ei sõltu ühestki teenusest ning kogu loogika toimib rakenduse siseselt, siis pole võrguühenduse olemasolu vajalik. Samuti ei pea sellisel juhul arendaja muretsema veebiserveri kättesaadavuse ja kiiruse üle.

6.5.2 Pildi tekstituvastuse teostaja

Teine võimalus on kasutada ekspertsüsteemi kõiki pakutavaid võimalusi ja vastupidi esimese lahendusega kogu tuvastamise loogika lõpprakenduse kihist välja tõsta. Alguses tuleb sarnaselt eelmise võimalusega toimida ehk selekteerida mallpiltidel huvipakkuvad tähemärgid, mida analüüsiks edasi saata. Esimene erinevus tuleb aga selles osas, lisaks huvipakkuvatele tähemärkidele on võimalus selekteerida pildilt konkreetseid alasid. Aladele saab määrata parameetreid, mitu rida ja veergu seal andmeid on ning kas konkreetse rea veerus võib esineda üks või enam tähemärki.

Kui kirjeldatud esialgsed seadistused on tehtud, siis saab tutvuda analüüsi tulemustega ning seejärel teha reeglite hulgast valik, mida hilisemal tuvastamisel kasutama peaks. Et pildituvastus oleks kasutatav, tuleks viimasena genereerida selekteeritud aladele võtme väärtused ning määrata, mis meetodeid tuvastamisel kasutatakse. Võimalik on võrrelda uute kujundite puhul kas ainult erinevust mallkujundist või lisaks kontrollida kas mingi tähemärgi

jaotused vastavad ka eelnevale analüüsile. Ehk kui näiteks algselt pildi mallilt selekteeritud kujundil oli tumedate pikslite arv paremas osas suurem kui vasakus, siis peab see kehtima ka hilisemas tuvastuse faasis. Vastasel korral ei anta tulemust.

Kui lõpprakenduses on tehtud pilt või lõigatud suuremast pildist välja pildiosa, mis vastab struktuurilt ja elementide paigutuselt mallpildile, saab seal olevate tähemärkide tuvastamiseks kasutada eelnevalt tehtud töö tulemust. Ekspertsüsteemi veebiteenusele antakse sisendiks pilt ja varasemalt genereeritud võtme väärtused. Selle põhjal leiab ekspertsüsteem vastava mallpildi ning suurendab või vähendab sisendpildi mõõtkava, et see vastaks malli mõõtmetele. Seejärel saab võtme väärtuste põhjal leida uuelt pildilt alad, mida tuvastama peaks ja teostada nendes piirkondades olevate kujundite analüüs. Enne seda jaotatakse vajadusel selekteeritud ala väiksemateks osadeks selle informatsiooni põhjal, mis rea ja veeru seaded algsel mallpildil tehti. Kui analüüsi tulemused on selgunud, siis võrreldakse neid andmebaasis olevate tähemärkide väärtustega, mis leiti algsel mallpildi sisestusel. Kõige täpsem vaste määratakse leitud tulemuseks. Vaste leidmine toimub selliselt, et võrreldakse kõiki parameetreid, mis reeglites võrdluseks lubatud oli ja arvutatakse protsendiline erinevus uue sisendiga. See tähemärk, kelle parameetrite summeeritud erinevus on kõige väiksem, on täpseim vaste. Pseudokoodis on see kirjeldatud lisa 5 all. Lõpuks tagastab veebiteenus lõpprakendusele tuvastatud tähemärgid, kus erinevate alade tähemärgid on võtme väärtuse järgi eraldatud ja sellisel kujul nagu algselt ekspertsüsteemis määrati.

6.6 Testimine

Ekspertsüsteemile eraldi iseseisvaid teste koostatud ei ole. Seda sellel põhjusel, et magistritöö demonstreerivate rakenduste arenduse käigus proovitakse ja kasutatakse kõiki olulisi ekspertsüsteemi funktsioone ning veendutakse nende toimimises. Rakendused seega katavad suure osa testimise vajadustest. Ülejäänud osas veendus lõputöö autor, et graafilises kasutajaliideses on kõik vajalikud nupud ning võimalused esindatud, testides seda nii Internet Exploreri kui Google Chrome veebibrauseriga.

7. Lahenduse rakendused

Näitamaks kuidas magistritöös välja töötatud lahendust kasutada saab, on kolmandaks töö oluliseks osaks demonstreerivate rakenduste arendus. See võimaldab anda ülevaate kui hästi piltidel olevate tähemärkide tuvastamine reaalses olukorras toimib ning kas midagi vajaks parandamist või täiendamist. Käesolevas peatükis on kirjeldatud kaks lõpprakendust, mis on arendatud prototüübina. Realiseeritud on see osa, mis tegeleb tähemärkide tuvastamisega. Ülejäänud funktsionaalsus on kirjeldatud, kuid arendamine jääb magistritöö skoobist väljajäämise tõttu hilisemaks.

7.1 Võimalik kasutusulatus

Idee magistritöö jaoks tuli seoses sellega, et töö autor tegeleb hobi korras nutitelefonid rakenduste arendusega Androidi operatsioonisüsteemile. Seetõttu on arendusel mõeldud sellele, et lahendus oleks võimalikult kompaktne ning sobilik rakendamiseks kaasaskantavatel nutiseadmetel, millel on sisseehitatud kaamera. Kuid sellega pole võimalused piiratud, samuti saaks magistritöö tulemust kasutada veebirakendustes. Näiteks leida skänneritud dokumentide pealt küsimustiku vastuseid nagu näidatud magistritöö punkti 7.3 all arendatud näidisrakenduses.

Nutitelefoni rakenduste puhul ei ole mõeldud ainult Androidi operatsioonisüsteemi peale. Kuna sellega on lõputöö autor kõige enam kokku puutunud, siis on seda näitena toodud ning üks demonstreerivad rakendus on samuti arendatud selle suunitlusega. Pikaajalisem eesmärk on rakenduste arendus lisaks väljatoodule ka *iOS* ja *Windows Phone* operatsioonisüsteemile. Seetõttu sai ka algoritm realiseeritud C++ programmeerimiskeeles. Kui ekspertsüsteemi kasutatakse kui ainult abistavat töövahendit nagu kirjeldatud magistritöö punkti 6.5.1 all, siis on võimalik pildi analüüsi funktsionaalsust realiseeriv osa SWIG [22] kasutades kiirelt vabalt valitud platvormil kasutusele võtta.

Käesoleva magistritöö raames arendatakse ainult rakenduste prototüübid, milles rõhk on pandud pildituvastuse funktsionaalsusele. Valmis tooteks, mida saaks rakenduste poest alla laadida, saab järgnevas alampeatükis kirjeldatud lahendus alles hilisema jätkutöö käigus. Seda just seetõttu, et lisaks pildituvastuse funktsionaalsusele tuleb veel tegeleda rakenduse disaini,

erinevate seadmete ühilduvusega ja kõrvalfunktsionaalsuse loomisega ning arvestada veel mitmete erinevate aspektidega.

7.2 Nutitelefoni rakendus

Esimene praktiline näide, mis kasutab magistritöös loodud lahendust, on Androidi operatsioonisüsteemile arendatud lotopileti tuvastuse ja numbrite kontrollimise rakendus. Algne idee selline rakendus luua oli ka magistritöö sisendiks. Võimalikule lahendusele mõeldes oli selge, et kõige keerulisem on arenduses numbrite tuvastamise osa, mis vajas põhjalikumat uurimist sobiva lahenduse leidmiseks. Järgneval on kirjeldatud rakenduse põhifunktsionaalsus ning seejärel pikemalt see osa, mis tegeleb tähemärkide tuvastamisega. Viimasena on toodud välja kuidas toimus testimine.

7.2.1 Funktsionaalsus

Lotopileteid ostetakse kas poest või siis internetist. Rakendus on mõeldud neile inimestele, kes esimest varianti kasutavad. Nende hulgas on kindlasti suur hulk neid, kellele meeldib piletil olevaid numbreid kontrollida vaadates televiisiorist lotosaadet. Samas on ka neid inimesi, kellel selleks aega ei jagu või kes soovivad teada ainult lõpptulemust, kas midagi võideti või mitte. Kui lotosaade jäi nägemata või seda polnudki plaanis vaadata, on võimalik minna poodi ning lasta seal pilt kontrollida. Või siis internetist tulemusi vaadata ning ühekaupa numbrid üle kontrollida. Arenduses olev rakenduse eesmärk on seda protsessi mugavamaks muuta.

Esimene tegevus kasutaja poolt oleks piletist pildi tegemine. Rakenduses avaneb kaamera vaade nagu näha jooniselt 16 ning ekraanil oleva suure ruudu ning juhatavate joonte abil saab teha õigest pileti osast pilti. Seejärel käivitub pildi tuvastus mille järel kuvatakse kasutajale tabelina pildilt loetud numbrid ja valik loosimise numbri valimiseks. Loosimise numbri lugemist on samuti võimalik automaatseks muuta, kuid see jääb hilisema arenduse faasi. Kui ilmneb, et midagi läks halvasti ja numbreid ei suudetud tuvastada, siis on võimalus uus pilt teha või muuta mõne üksiku ruudu väärtust, kus esines viga.

Peale kasutaja poolset lotonumbrite kinnitamist kontrollitakse välisest süsteemist loosimise toimumise aega. Kui tegemist ei ole kõige uuema piletiga ehk loosimine on juba toimunud ning tulemused teada, siis avaneb võimalus kõik numbrid kohe ära kontrollida ja resultaat eri värvi abil selgelt esile tuua. Teine võimalus on käivitada animatsioonidega kontroll. Selle

valiku puhul hakkavad võidunumbrid ekraani alumises osas järjest ilmuma, üks number iga kahe sekundi tagant. Samaaegselt märgitakse ekraani keskel paiknevas tabelis eristava värviga need numbrid, mis kasutaja piletil olemas. See annab kasutajale põnevust juurde ning tõstab rakenduse väärtust.

Juhul kui loosimine pole veel toimunud või tulemused pole veel kättesaadavad, siis käivitub teine stsenaarium. Tuvastatud numbrid salvestatakse koos loosimise numbriga. Iga teatud aja tagant käivitub taustaprotsess, mis teeb päringuid loosimise tulemuste pärimiseks. Päring tehakse vaid juhul kui on loosimise päeva õhtu või järgneva päeva hommik. Ning kui vastus on kord saadud, siis enam hiljem välist serverit ei koormata. Vastust saades saadetakse kasutaja nutitelefoni ekraanile väike teavitus, mis ilmneb teate ikoonina. Seda avades on võimalik käivitata juba eelnevas lõigus kirjeldatud numbrikontroll. Seejärel siis kas näidatakse tulemused kohe välja või käivitatakse animatsioonidega kontroll.

7.2.2 Teksti skaneerimise vaade

Eelnevalt kirjeldati rakenduse põhifunktsionaalsus ja peamised kasutusjuhud. Selleks, et piletil olevate numbrite tuvastus töötaks, tuli arendada eraldi vaade pildi tegemiseks. See on näha jooniselt 16. Keskmine ala on kaamera eelvaates eraldatud seetõttu, et suunata kasutajat pilti tegema täpselt sellest piirkonnast, mis on vajalik edasiseks töötluseks. Selle jaoks, et kasutajale õige kõrguse ja laiusega piirkonda vaates luua, tuli esmalt selgeks teha sobiv mõõtkava. Selle jaoks mõõtis magistritöö autor füüsilisel Bingo Loto [28] piletil numbritega mänguvälja ning sai kõrguse ja laiuse suhteks 36:50. Saadud informatsiooni põhjal oli võimalik arendada loogika, mis rakenduses kaamera vaatevälja kesksele sama suhtega ristküliku tekitas ja väljaspool seda piirkonda tausta hallikamaks muutis.

Teine võte kasutaja suunamiseks skaneerimise vaates on abistav ruudustik, mis on samuti näha jooniselt 16. Kuna see kattub osaliselt piletil endal oleva ruudustikuga, siis joonisel see kohe selgelt ei eristu, kuid reaalses rakenduses on see kohe märgata. Selle abil jaotatakse mänguväli 25 ruuduks nagu on piletil olevate numbrite jaotus.

Eelnevalt kirjeldatud kahe abistava võtte abil on tehtud kasutajale ilmselgeks see, mis piirkond peab jääma pildi keskele. Kui kasutaja järgib juhiseid ja asetab kaamera õigesti, siis saab kasutada antud töös välja töötatud lahendust. Tänu sellele ei pea tegelema pildi õigesse suunda keeramise ja mänguvälja leidmisega. Rakenduses saab peale kasutajapoolset pildi

tegemist kohe eraldada õige piirkonna ning jagada selle 25 osaks, millele igaühele analüüs teostada.

Joonis 16 Ekraanipilt nutitelefoni rakenduse pildi skaneerimise vaatest

7.2.3 Testimine

Veendumaks, et rakenduse pildituvastamine töötab, teostati tegevusi mitmes osas. Esiteks tuli teha eeltööd ekspertsüsteemis, laadida üles lotopileti mall ning selekteerida erinevad numbrid. Selekteerimise vaade oli näidatud joonisel 14 ning toimimine kirjeldatud samuti ekspertsüsteemi peatüki all. Kui kõik võimalikud märgid olid selekteeritud, sai vaadata analüüsi tulemusi nagu oli näha joonisel 15. Analüüsi tulemuste vaates kuvatud andmete põhjal sai kirjeldada rakenduses staatilised klassid, kus defineeriti tähemärgi ja nende omadused.

Teiseks tuli rakenduses arendada loogika, mis leiaks uuel sisendpildil olevale kujutisele kõige parema vaste. Selle arendamise aluseks võeti magistritöö lisa 5 all olev pseudokood.

Realiseeriti see eraldi klassiteegina, et järgnevate rakenduste puhul ei peaks seda loogikat uuesti arendama või kopeerima, vaid piisab ühe klassiteegi projekti lisamiseks. Sellega on kõik ettevalmistused tehtud ning jääb vaid lõplik testimine.

Testimiseks võeti lotopiletit ja skaneeriti sisse vastavalt 7.2.2 peatükis kirjeldatud juhistele. Selle peale lõikas rakendus taustal keskmise olulise infoga piirkonna ülejäänud pildist välja ja jagas selle omakorda 25 väiksemaks piirkonnaks, mis igaüks oli sisendiks SWIG [22] abil Androidi projekti integreeritud analüüsivale algoritmile. Analüüsi tulemust omakorda kasutati eelmises lõigus kirjeldatud parima vaste leidmise sisendiks. Seda loogikat kasutades leiti kõigi 25 piirkonna sees olevad numbrid ning kuvati nutitelefoni vaates ekraanile komaga eraldatud viies reas.

Joonisel 16 olevate numbrite töötlus esimestel katsetel ei olnud kohe 100% edukas, tuvastada suudeti ligi 70-80%. Uurimisel selgus, et tumeda piksli definitsioonis oli liiga väike väärtus. Kuna mõned tähemärgid valguse peegelduse mõjul olid heledamad kui mallis kasutatud, siis ei suudetud sarnases koguses tumedaid pikselid leida. Muutes tumeda piksli definitsiooni väärtust suuremaks, oli uus katse edukas ehk kõik numbrid suudeti tuvastada. Edasi korrati katset veel kahe erineva pileti peal. Need testid osutusid samuti edukaks.

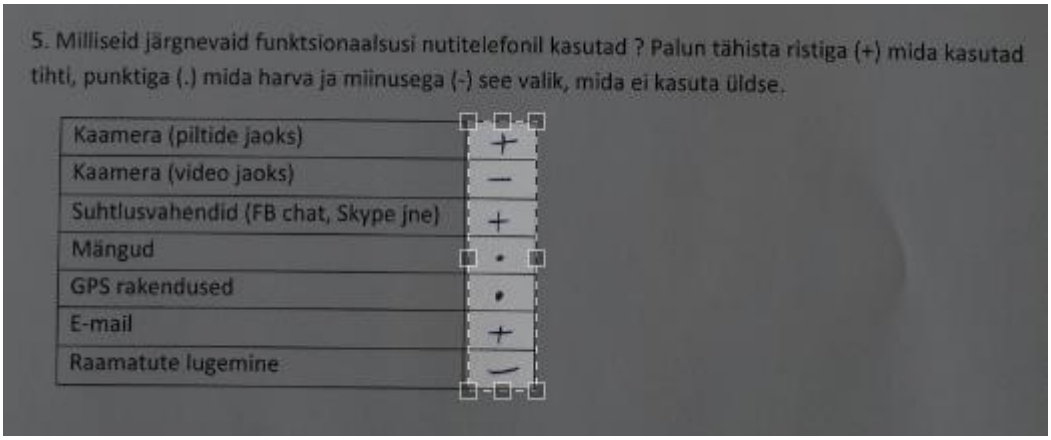
7.3 Veebirakendus

Teine praktiline näide erineb esimesest selles poolest, et lõpprakenduseks on arendatud veebirakendus. See on lihtsa ülesehitusega ja mõeldud ainult demonstratiivsetel eesmärkidel näitamaks ekspertsüsteemi kasutusvõimalusi.

On koostatud üks lihtne küsimustiku vorm, mis on näha käesoleva töö lisa 6 alt. Vorm koosneb 5 erinevast küsimusest, vastuseks tuleb kasti kirjutada sobiva valiku tähiseks olev täht suure trükitähena või siis 5-nda küsimuse puhul üks juhendil olevatest tähistest. Eesmärk on lugeda küsimustikule antud vastused automaatselt süsteemi. Selle jaoks kas skaneeritakse või tehakse pilt A4 olevast küsimustikust ja laetakse see veebirakendusse, kust see edastatakse ekspertsüsteemi veebiteenusele tuvastamiseks.

Enne seda, kui on võimalik ekspertsüsteemi tuvastamise funktsionaalsus kasutada, tuleb seadistada pildi mall. Selle jaoks on võetud magistritöö lisa 6 all olev vorm malliks. Seal on esindatud kõik võimalikud vastuste kombinatsioonid, seega saab ühe pildiga ekspertsüsteemi seadistused ära teha. Seadistus toimib sarnaselt varem kirjeldatud lahendustele, seekord tuleb

ainult lisaks tähemärkidele määrata piirkonnad, mis tuvastamisele lähevad. Neid on sellel pildimallil 5, lõputöö lisa 6 vaadates on need kastid, kuhu küsimustikule vastajad peaks oma vastuse kirjutama. Järgnevalt on joonisel 17 näitena välja toodud viimase piirkonna selekteerimine. Ekspertsüsteemis seda piirkonda selekteerides peab kindlasti määrama, et seal on ühes veerus seitse rida. Selle informatsiooni põhjal saab hiljem selle ala osadeks jagada ja igale väiksemale piirkonnale tuvastamise funktsionaalsust rakendada.



Joonis 17 Küsimustiku vastuste piirkonna selekteerimine

Kui kõik piirkonnad ja tähemärgid on selekteeritud, siis tuleb seadistused salvestada, mille järel on võimalus näha statistikat tähemärkide analüüsi kohta. See on välja toodud joonisel 18, kus tasub eraldi välja tuua viimased kolm tähemärk. Ristile on vastavusse pandud number 3, punktile number 1 ja kriipsule 0. Seda on tehtud seetõttu, et lõpprakenduse vaatenurgast on oluline saada summaarne number nutitelefoni kasutuse kohta. Ja kuna numbrilist väärtust on parem töödelda ning võrrelda, siis on kasutatud ekspertsüsteemi võimalust ise defineerida, mis tähemärgile mis väärtus vastab. Teine aspekt, mis välja paistab, on miinuse (mille vasteks seatud 0) analüüsi tulemused. Kuna algoritmi tööpõhimõte tumedate pikslite protsendi jaoks oli selline, et võetakse aluseks tähemärgi kõrgus ning selle põhjal luuakse ruut ja arvutatakse selle pikslite hulk. Seejärel jagatakse kogu pildi tumedate pikslite arv ruudu pindalaga ning saadakse tumedate pikslite protsent. Selle arvutus aga annab tumedate pikslite osakaaluks 600%.

Character	Character image	Dark pixels %	Dark pixels 40% right	Dark pixels 40% left	Dark pixels 40% center	Dark pixels 20% top	Dark pixels 20% bottom	Dark pixels 30% middle
A		29,86 %	67,44 %	23,26 %	23,26 %	23,26 %	39,53 %	67,44 %
B		28,57 %	32,14 %	50 %	35,71 %	23,21 %	28,57 %	53,57 %
C		14,29 %	21,43 %	53,57 %	21,43 %	32,14 %	32,14 %	35,71 %
D		22,45 %	38,64 %	52,27 %	29,55 %	25 %	34,09 %	45,45 %
E		45 %	31,11 %	46,67 %	48,89 %	33,33 %	31,11 %	37,78 %
3		35 %	22,86 %	48,57 %	60 %	14,29 %	14,29 %	74,29 %
1		68 %	47,06 %	52,94 %	58,82 %	17,65 %	58,82 %	82,35 %
0		600 %	66,66 %	66,66 %	66,66 %	100 %	100 %	100 %

Legend: >=75% <75% & >=50% <50% & >=25 % <25%

Joonis 18 Küsimustiku tähemärkide analüüsi tulemused

Vaatamata sellele, et ühe tähemärgi tulemused langevad äärmustesse, ei mõjuta see üldist tuvastust. See on väga eristav omadus ja parima vaste leidmisel tuleb see välja. Küll tasub ekspertsüsteemi edasi arendades arvestada selliste juhtudega. Näiteks kõikidel sellistel juhtudel, kus laius on suurem kui kõrgus, määrata üks tõeväärtuse parameeter, millega seda eristada saab ning teha muud arvutused teistest erinevalt. Nii saaks hallata olukordi kui on mitu miinusmärgi sarnast tähemärki.

Viimaks testimaks seadistuste ning veebirakenduse toimimist, täitis lõputöö autor veel 5 sellist küsimustiku erinevate vastustega, tegi nendest pilti ning sisestas veebirakenduse vormi kaudu need järjest süsteemi. Veebirakendus tegi iga pildi puhul taustal päringu ekspertsüsteemi veebiteenusele andes sisendiks uue pildi ja võtme väärtused, mis eelneva seadistuse käigus genereeriti. Tulemuseks oli vastus järgneva struktuuriga:

```
public class OcrResponse
{
    public List<ResponseArea> ResponseAreas { get; set; }
}
```

Kus `ResponseArea` on struktuurilt:

```
public class ResponseArea
{
    public string ApiKey { get; set; }
    public int RowCount { get; set; }
    public int ColumnCount { get; set; }
    public List<ResponseCharacter> ResponseCharacters { get; set; }
}
```

Kus `ResponseCharacter` on struktuurilt:

```
public class ResponseCharacter
{
    public string Character { get; set; }
    public int Row { get; set; }
    public int Column { get; set; }
}
```

Struktuur on keerukas just seetõttu, et võimaldab väga dünaamilist kasutust, mille põhjal klientrakenduses oleks vastust kerge ja arusaadav töödelda. Eelneva päringu vastus

konverteeriti veebirakenduses tabeli kujule, mille järel sai veenduda tuvastamise õigsuses. Sama protseduuri korrati veel 4 erineva küsimustiku vormiga mis olid juhuslike andmetega täidetud ning kõigil juhtudel andis tuvastamine korrektsed vastused. Vastused tõlgendati enne kuvamist sobivale kujule, vastuse tähemärkidele seati vastavusse tähendusega ja viimase küsimuse punktid liideti kokku. Veebirakenduse kuvatav lõpptulemus on näha tabelist 6.

Tabel 6 Küsimustiku vastused töödeldud tabelkujul

	Omab nutitelefoni	**Vanus**	**Nutitelefoni kasutuse aeg**	**Ostul määravad faktorid**	**Kasutusulatus skaalal 0-21**
Vastus 1	Jah	21-25 a.	rohkem kui 4h	disain, tarkvara, hind	11
Vastus 2	Ei	-14 a.	0-30 min	hind, disain, tarkvara	0
Vastus 3	Jah	26-40 a.	2h-4h	kaamera, hind, sensorid	8
Vastus 4	Jah	21-25 a.	1h-2h	tarkvara, disain, hind	15
Vastus 5	Ei	üle 41 a.	0-30 min	hind, disain, kaamera	0

Tuvastamine õnnestus tänu sellele nii hästi, et tähemärgid olid teistest üsna erinevad ning piiratud hulgal. Ning samuti olid kõik küsimustiku vormid A4 paberi kujul, mis tähendas, et vastuste erinevaid piirkondi oli ekspertsüsteemis peale algset malli seadistamist hiljem juba kerge kindlaks teha. Selleks võeti süsteemi salvestatud piirkondade koordinaadid. Seejärel võrreldi uue pildi suurust mallpildiga ning erinevuse puhul arvestati koefitsienti, mille alusel uuelt pildilt õiged piirkonnad pikslite alusel leiti.

8. Hinnang

Käesolevas peatükis antakse hinnang arendatud lahendusele ja lõpptulemusele. Välja tuuakse negatiivsed ja positiivsed küljed, mis töö käigus selgusid. Antakse hinnang kui hästi seda rakendada saab, võttes aluseks testimise tulemused ja viimasena arendatud prototüüplahendused. Seejärel võrreldaks magistritöö käigus arendatud lahendust alternatiivsete lahenduseta, mida kirjeldati peatüki 4.2 all. Viimasena tuuakse välja edasiarendused, mis on magistritöö autoril plaanis jätkutööna teha. Lisaks kirjeldatakse võimalused, mida rakendades saaks välja töötatud lahendust veel paremaks muuta.

8.1 Tulemuse rakendatavus

Magistritöö autori hinnang välja töötatud lahendusele on positiivne. Seda saab kasutusele võtta töö algul püstitatud probleemi lahendamiseks, kui pildid on kindla struktuuriga ja tähemärgid sama stiiliga. Esimeste katsetuste käigus kõik kohe sujuvalt ei edenenud, kuid uurimise käigus sai probleemide põhjused teada ning ka lahendus leitud. Ilmnes, et suuremat tähelepanu tuleb pühendada järgnevatele testimise käigus välja tulnud aspektidele:

1) Tumeda piksli väärtus – määrata piisavalt kõrge väärtus. Üks parameetritest, mis tuleb analüüsiva algoritmi realisatsioonile sisendiks anda, on tumeda piksli maksimaalne väärtus. See väljendas RGB eri värvide väärtuse summat. Kui esialgu testiti tumeda piksli väärtusega 150, siis ilmnes osade märkide puhul tuvastamisel probleeme. Uurimisel selgus, et valguse peegelduse mõjul või pildi kerge hägususe puhul võib sellise piksli RGB väärtuste summa olla isegi kuni 300-400 vahel, kuigi ideaalsetes tingimustes tehtud pildi puhul jääb see all 100. Seega lahenduseks on tumeda piksli ulatuse laiendamine, arvestades siiski piisava varuga, et kujundi taustal olevad pikslid sellesse ulatusse ei langeks.

2) Kvaliteetne mallpilt – selge ning suure resolutsiooniga. Esialgne ekspertsüsteemi analüüs tuleb teha mallpildile, mis on võimalikult selge. Eesmärk on vähendada võimalust, et hägusused või näiteks kaamera ekraanil olev mustus muudaks tulemust. Samuti on oluline, et pilt oleks võimalikult suure resolutsiooniga, et üksikute tähemärkide alla jääks võimalikult palju pikselid. Sellisel juhul on erinevad analüüsi parameetrid adekvaatsemad ja võimalik vea oht on väiksem.

3) Õige reeglite valik – vältida reeglite rakendamist, mis on väga väikse väärtusega. Näiteks kui esialgse mallpildi analüüsis selgub, et mingi tähemärgil moodustavad alumises 20% piirkonnas olevad tumedad pikslid vaid 6% kõigist võimelikest, siis tasuks seda reeglit parima valiku leidmisel antud tähemärgi puhul mitte rakendada. Seda seetõttu, et kuna parima valiku leidmine töötab protsente võrreldes, siis uue pildi puhul võib teise resolutsiooni tõttu mõni piksel teistmoodi olla, mis aga protsentide võib tähendada näiteks 6% tõusu 9%. See aga on protsendilist väga suur erinevus ja võib seetõttu moonutada parima valiku tulemust.

Arvestades välja toodud aspektidega on võimalik vähendada potentsiaalsete vigade ohtu. Samuti reeglite piiramine muudaks programmi tööd kiiremaks, kuna on vähem tingimusi mida kontrollida. See on ka antud lahenduse üks tugevatest külgedest, et reegleid saab vajadusel sisse ja välja lülitada. Ehk kui tähemärke on rohkem või on sarnaseid palju, siis tasub võimalikult palju reegleid rakendada. Kui aga tähemärke on vähe või nad on väga erinevad, siis piisab vaid mõne reegli kontrollist ja rakendus töötaks seeläbi väga kiirelt.

8.2 Võrdlus sarnaste olemasolevate lahendustega

Valmis lahendust saab alternatiivsete lahendustega võrrelda erinevate mõõtude järgi, järgnevalt on seda tehtud täpsuse ja kiiruse osas. Võrdluses on arvestatud magistritöö alguses seatud piirangutega, mis tingimuste puhul uus lahendus on mõeldud töötama. Seega täpsuse puhul on eeldatud, et on teada õige piirkond ning pilt on õiges asendis.

8.2.1 Täpsus

Kui võrrelda magistritöö käigus arendatud lahenduse täpsust alternatiividega, siis tuleb tõdeda, et lihtsamates olukordades on täpsus võrdselt hea, kuid keerulisemates olukordades on alternatiivid eelisolukorras. Lihtne olukord tähendab seda, kui tähemärk on kuni 15 ja nad on piisavalt erineva kujuga. Erinev kuju tähendab seda, et silmaga vaadeldes on kaks tähemärki selgelt eristuvad ning ei teki kahtlust juhul, kui peaks valima ühe. See oli samas loogiline ja etteaimatav tulemus, kuna välja töötatud lahendus on mahult üsna väike võrreldes olemasolevate lahendusega. Vaatamata sellele täidab tulemus seatud eesmärke, kuna võimalike lahendustena oligi silmas peetud lihtsamaid olukordi, kus erinevate tähemärkide koguarv jääb maksimaalselt kümne lähedale.

Välja arendatud lahendus omab eeliseid aga selles osas, et ekspertsüsteemi seadistades saab ette määrata, mitmeks reaks ja veeruks tuvastatav piirkond jaguneb. See võimaldab tabelite puhul, kus lahtrid on sama suurusega, saavutada väga täpset tulemust. Ridade ja veergude eraldajad ning tühikud ei sega õige lõpptulemuse saavutamist.

8.2.2 Kiirus

Üks olulisi aspekte, miks oli vaja uus lahendus arendada, oli kiirus. Tänu algoritmi lihtsusele ning selle realiseerimisele C++ ja täpse vaste leidmise kontrollide vähesusele oli oodata, et see peaks töötama kiirelt. Selle testimise ja võrdlemise jaoks võttis magistritöö autor Bingo Loto [28] pileti, mis oli välja toodud varem joonisel 14 ja mõõtis tähemärkide tuvastamise kiirust. Selle jaoks kasutati Visual Studio 2010 [20] keskkonda, kus üldine loogika kirjutati C# keeles ning kiiruse mõõtmiseks kasutati raamistiku võimalusi. Sama tehti ka Tesseracti [3] ja ABBYY [13] puhul. Nii magistritöös arendatud lahendus kui ABBYY suutsid tuvastada kõik numbrid. Tesseracti puhul õnnestus algul 60% numbritest, lülitades põhjalikuma analüüsi režiimi sisse, siis 80% numbritest. Probleem tekkis äärisjoontega, mida ei osanud see OCR mootor osadel juhtudel tõlgendada. Testi tulemused on näha tabelist 7, kus Tesseracti puhul on välja toodud põhjalikuma analüüsiga variant. Ühikuks on aeg millisekundites.

Tabel 7 Tekstituvastuse lahenduste kiiruse võrdlus

	Katse 1	Katse 2	Katse 3	Keskmine aeg
Magistritöö lahendus	171 ms	163 ms	158 ms	164 ms
Tesseract	1125 ms	1143 ms	1107 ms	1125 ms
ABBYY	2450 ms	2671 ms	2502 ms	2541 ms

Ootuspäraselt on kõige kiirem magistritöö käigus välja arendatud lahendus. Sealjuures tasub arvestada, et kõik analüüsi reeglid olid sisse lülitatud ehk seda annaks pikema testimise ja katsetamise käigus veel kiiremaks muuta. Otsest vajadust midagi optimeerida aga ei ole, kuna 164 millisekundit on piisavalt kiire aeg. Kasutaja jaoks tundub see kui kohene vastus. Lõpprakenduses kuluks rohkem aega kasutajaliidese kuvamiseks ja uue vaate avamiseks, kui pildi töötlemiseks.

Tesseracti puhul tasub ära mainimist see, et sai valida mitme erineva tuvastuse tüübi vahel. Kõige kiirem neist suutis tuvastuse teha isegi kiiremini kui magistritöös välja töötatud lahendus ehk ligi 100 millisekundiga, kuid selle täpsus oli alla 60%. Seetõttu valiti võrdluseks seda tüüpi valik, mis edukamalt tuvastamist teostas. Antud valiku negatiivne külg on aga see,

et kasutatakse treenitud andmeid, mis lisab vähemalt 20 MB mahtu. Arvestades, et nutitelefonide puhul on mälu veel üsna piiratud, siis on see piirav tegur.

ABBYY puhul võttis tähemärkide tuvastamine nii kaua aega, kuna tegemist on reaalajas töötava veebiteenusega ning 3 MB pildi saatmine võttis suure osa ajast. Samas tõi see selgelt välja selle, et nutitelefonis tekstituvastamise kasutamiseks peaks olema tekstituvastuse loogika arendatud lõpprakenduse kihis. Kui arvestada seda, et telefonide kaamerad lähevad paremaks ja teevad parema kvaliteedi ning suurema pikslite arvuga pilte, siis pildi suurused kasvavad veel. Kirjeldatud test viidi läbi arvutis, millel on kiire võrguühendus. Mobiilse andmeside puhul võib 3 MB pildi saatmine aega võtta 10 sekundit või isegi rohkem. See aga oleks kasutajale väga ebamugav, kui peaks millegi taga nii kaua ootama. Pildituvastust saab küll teha taustal ning kasutajale kuvada midagi muud, kuid arvestades rakenduse põhiloogikat ei oleks see hea valik.

8.3 Võimalikud edasiarendused

Plaanis olevad edasiarendused puudutavad just nutitelefoni rakendust mis kirjeldatud peatüki 7.2 all. Magistritöö autoril on varasemast juba Androidi rakenduste poes avaldatud kolm tasuta rakendust, neist kõige populaarsemat on kahe esimese kuuga installeeritud enam kui 30 000 nutiseadmesse üle maailma ning igapäevasel kasutab seda rohkem kui 1 750 inimest. See on rakendus nimega IR Remote Control [30], mis võimaldab nutitelefoni või tahvelarvutit kasutada televiisiori, muusikakeskuse, õhukonditsioneeri ja paljude muud seadmete puldina. Eelduseks on see, et nutiseade suudab väljastada infrapuna valgust ja Androidi operatsioonisüsteemi versioon on vähemalt 4.4. Tänu sellele on magistritöö autoril huvi, teadmised ja kogemused olemas, kuidas käesolevas töös arendatud rakenduse prototüüp toimivaks tooteks muuta ning samuti avalikule turule viia.

Väga oluline on esmamulje, seetõttu ei ole kiirustatud rakenduse arendusega, vaid keskenduti töös pildituvastuse funktsionaalsusele. Edaspidi on kavas realiseerida magistritöös kirjeldatud nutitelefoni rakenduse puuduolev funktsionaalsus ning seejärel veelkord testida võimalikud kasutusjuhud üle ning vajadusel täiendusi ja parandusi sisse viia. Üks võimalik täiendus oleks lisakontroll enne pildi tükeldamist ja analüüsiks edastamist. Lisakontrolli eesmärk oleks tuvastada lotopileti mänguvälja nurgad, ning analüüs teha ainult selle sisse jäävale piirkonnale. Algses versioonis võetakse selleks piirkonnaks kasutaja ekraani keskel olev ilma halli taustata piirkond. Kui aga kasutajal pilti tehes käsi väriseb või pilet on veidi kortsus, siis

võib tuvastuse sisendiks sattuda vale piirkond ja mõjutada kogu tuvastamise tulemust. Teine täiendus oleks lisaks loosimise numbri automaatne tuvastamine, et seda ei peaks kasutaja valima. See aga ei ole nii oluline, kuna kuupäeva alusel on võimalik suure tõenäosusega kohe õige loosimise number pakkuda. Kui aga tegemist on vanema piletiga, siis saab kasutaja seda ühe nupuvajutusega kiirelt muuta.

Viimases arenduse etapis on plaanis lisada Google Analytics [31] jälgimine. See võimaldab koguda rakenduse kohta erinevat statistikat, milliseid vaateid on kasutatud ja kuidas kasutaja rakenduses on ringi liikunud. Lotopileti tuvastamise rakenduses saab seda kasutada lisaks mainitule veel sündmuste salvestamiseks. See võimaldaks näha statistikat, kas programm suutis tähemärke tuvastada ja kas kasutajad on enne kinnitamist numbreid muutnud või mitte. Selle informatsiooni põhjal saab hinnata võimalikke probleeme ja saada ülevaade nutitelefonide mudelitest, millel probleemne käitumine esines.

Lisaks rakenduse lõpuni arendusele on võimalusi algoritmi ja ekspertsüsteemi täiendamiseks. Algoritmi oleks võimalik paremaks muuta, lisades sinna veel uusi reegleid, mis kontrollivad uusi seni veel katmata omadusi. Näiteks tähemärgi parema ja vasaku diagonaali pikslite suhet. Ekspertsüsteemi saaks integreerida veel lisaks Tesseracti [3] OCR mootori ning lisada võimaluse kasutada mingi piirkonna tähemärkide tuvastamiseks hoopis selle pakutavaid meetodeid. See oleks abiks kui tähemärke on palju või tekst on sellisel kujul, et tähed on omavahel ühendatud. Kuna viimane variant pole antud magistritöö lahenduse puhul toetatud, siis saaks sellistel juhtudel kasutada keerulisemat ja mahukamat kuid mainitud juhul täpsemat Tesseracti OCR mootorit.

9. Kokkuvõte

Käesolev magistritöö sai alguse töö autori hobist, milleks on rakenduste arendus nutiseadmetele. Tekkis idee arendada rakendus, mis suudaks tuvastada lotopiletist tehtud pildilt mänguvälja numbrid ning kontrollida kas tegu on võidunumbritega. Leidmaks arenduse kõige keerulisemale osale ehk tähemärkide tuvastamisele hea lahenduse, sai magistritöö aluseks võetud probleemi põhjalik uurimine ning parima lahenduse leidmine. Peale nõuete detailselt kirjeldamist ning potentsiaalsete valikute võrdlemist sai selgeks, et probleemi lahendamise parimaks valikuks on uue lahenduse väljatöötamine.

Uue lahenduse väljatöötamine sai seega magistritöö põhieesmärgiks. Vaja oli pildi tekstituvastuse funktsionaalsust piiratud tingimustes, kus piltide struktuur ja võimalikud esinevad tähemärgi on ette teada. Oluline oli, et lahendus oleks kiire, täpne, väikse mahuga ja sobiks kasutuseks nutitelefoni rakendustes. Lisaks pidi lahendus töötama erinevate tähemärkidega ning kujunditega, ning mitte olema piiratud vaid numbrite või tähtedega. Seetõttu seati töö üheks põhieesmärgiks algoritmi väljatöötamine, mis suudaks tähemärke analüüsida. Teiseks eesmärgiks sai ekspertsüsteemi arendus, mille abil saaks seadistada mida on vaja analüüsida ning kus oleks tulemusi näha. Samuti pidi selle abil olema võimalik tähemärkide tuvastamist teostada. Viimasena eesmärgina oli seatud kahe prototüüplahenduse arendamine, mis testiks ja demonstreeriks eelneva töö toimimist.

9.1 Töö tulemused

Magistritöö esimese tulemusena töötati välja algoritm, mis on võimeline pilti analüüsima seitsme erineva parameetri järgi. Võrreldakse tähemärgi tumedate pikslite osakaalu erinevates kujundi piirkondades võrreldes kõigi tumedate pikslitega. Tumeda piksli definitsioon on sealjuures algoritmile sisendiks, jäädes vahemiku 0 kuni 765 ning näidates RGB eri värvide väärtuse summat. Lisaks võimaldab algoritm kindlaks teha, kas pildil on ääris ning leida mitu tähemärki seal esineb. Selle alusel jäetakse analüüs tehes äärise tumedad pikslid välja ning igale tähemärgile tehakse analüüs eraldi. Algoritm realiseeriti C++ programmeerimiskeeles võimaldamaks selle kasutust võimalikult paljudel erinevatel platvormidel.

Teise tulemusena valmis ekspertsüsteem, mis on jaotatud neljaks erinevaks komponendiks. Üks neist on vastutav äriloogika eest, koondades kõikidele osadele ühised funktsioonid ning

infovahetuse algoritmi realisatsiooniga. Teine komponent on andmebaasi ligipääsu kiht, mis võimaldab andmebaasi andmete salvestamist, muutmist ja kustutamist. Mõlemad mainitud komponenti kasutab kolmandaks olev veebirakendus, kus on võimalik uusi pildimalle sisestada ning pilte hallata. Kogu lahenduse jaoks olulise funktsionaalsusena saab selle abil määrata, millised on pildil tuvastamist vajavad piirkonnad ning selekteerida pildilt tähemärke, mille peal analüüs tuleb läbi viia. Viimane ekspertsüsteemi komponent on veebiteenus, millele saab anda sisendiks uusi pilte ning mis eelnevalt ekspertsüsteemi veebirakenduses tehtud seadete põhjal teostab tähemärkide tuvastamist ning tagastab tulemused.

Kolmanda olulise tulemusena arendati kaks prototüüplahendust demonstreerimaks välja töötatud lahenduse toimimist. Esimene oli Androidi operatsioonisüsteemile mõeldud nutitelefoni rakendus, mis telefoni kaamera abil tehtud pildilt lotopileti numbreid tuvastas ja võidunumbritega võrdles. See lahendus kasutas ekspertsüsteemi ainult analüüsiks. Rakenduses kasutati otse algoritmi realisatsiooni võrdlemaks uute piltide analüüsi tulemusi ekspertsüsteemist saadud mallpildi analüüsi tulemustega. Teine prototüüp oli lihtne veebirakendus, mis kasutas ekspertsüsteemi veebiteenust A4 formaadis kindla struktuuriga küsimustiku vastuste tuvastamiseks.

9.2 Eesmärkide saavutamine ja järeldused

Magistritöö algul püstitatud eesmärgid suudeti täita. Välja töötati uus algoritm ning see realiseeriti, samuti valmis abistav ekspertsüsteem, kus graafilise kasutajaliidese abil mugavalt uusi pildimalle on võimalik seadistada. Selle testimiseks ja kasutusvõimaluste demonstreerimiseks valmisid rakendused, kus realiseeriti see osa, mis oli seotud piltide tekstituvastusega. Rakenduste testimine näitas, et tulemus on piisavalt täpne ja kiire, et seda magistritöö algul püstitatud probleemi lahendamiseks rakendada.

Lisaks tuli välja, et kuna nutitelefonide pildi resolutsioon on muutumas aina suuremaks, siis pildi mahu tõttu on mõttekas kogu tähemärkide analüüs realiseerida rakenduse sees kasutamata väliseid veebiteenuseid. See on ka käesoleva töö arenduse puhul toetatud, kui esialgne analüüs teha ekspertsüsteemis ning seejärel defineerida rakenduses konstandid saadud tulemustega ning integreerida algoritmi realisatsioon. Sama valik tehti ka esimeses demonstreerivas Androidi operatsioonisüsteemile mõeldud rakenduses.

9.3 Edasiarendused

Edasiarendusena on magistritöö autoril esmalt plaanis lõpetada esimeseks demonstreerivaks rakenduseks olnud nutitelefoni rakendus Androidi operatsioonisüsteemile. Selle jaoks realiseeritakse magistritöös küll kirjeldatud kuid mitte arendatud funktsionaalsus, milleks oli ilusam disain, võidunumbrite pärimine ja animatsioonidega tulemuse näitamine. Lisaks tuleb teostada muud rakenduse avalikustamisega seotud tegevused. Pikaajalisema eesmärgina on kasutada magistritöös arendatud lahendust ka teistes rakendustes, kus sarnast pildituvastuse funktsionaalsust vaja peaks minema. Ning seejuures mitte piirduma Androidi operatsioonisüsteemiga, vaid lisaks laieneda ka *iOS* ja *Windows Phone* operatsioonisüsteemile. Seetõttu sai ka algoritm realiseerud C++ programmeerimiskeeles, et pildituvastuse tuumfunktsionaalsust saaks kergelt integreerida erinevate platvormide jaoks.

Muud lisaarendused võivad tulla seoses algoritmi täiustamisega. Kui hilisemas faasis selgub, et mingite juhtudel esineb tuvastamisel probleeme, saab algoritmile lisada uusi reegleid, mille põhjal pilte analüüsida. Võimalik on ka ekspertsüsteemi avalikuks kasutuseks lubamine, et teised arendajad üle maailma saaks selle võimalusi kasutada kui nende rakenduses on vaja teostada piltide tekstituvastusi sarnastel piiratud tingimustel kui antud töö puhul. See aga jääb kaugemasse tuleviku, kui esimesed edasiarenduse plaanid on juba edukalt teostatud ja toimimist tõestanud.

Summary

The initial idea for this thesis derived from the author's hobby of developing smartphone applications. The idea emerged to develop an application that would be able to detect lottery ticket numbers in digital photographs and check these against winning numbers. The thesis aimed to find the best solution for the most sophisticated part of this functionality, which is number recognition in a given image. After delving into details of the problem and comparing different possible solutions, a decision was made that the best option was to develop a new solution.

Developing a new solution thus became the main objective of this master's thesis. It was necessary to have image OCR functionality in limited circumstances, where different images have the same structure and possible characters are known in advance. A requirement of the solution was that it had to work with a variety of shapes and characters and not be limited to merely numbers or letters. Another important aspects were speed, precision, small size and suitability for smartphone applications. Thus, the main aim of the work was the development of an algorithm that is able to analyze characters. The second goal was the development of an expert system, which can be used to configure analysis settings and to show the final results. With help of the algorithm, it also had to be capable of identifying characters. The third and final goal was to develop two prototype applications to test and demonstrate the results of the prior work.

Project results

As the first result of the project, a new algorithm was developed, which is capable of analyzing an image based on seven different parameters. It compares the dark pixels of a character in different areas of the image in relation to all the dark pixels. Dark pixels definition is an input parameter for the algorithm. It has a value between 0 and 765 and shows the sum of different RGB color values. Moreover, the algorithm determines whether the image has a border and the number of characters present. On the basis of this information, the dark pixels on the border are not included in computation and analysis is done separately for each character. The algorithm was implemented using the C++ programming language to enable its later use on many different platforms.

The second important result of the project was the expert system, which can be divided into four different components. One of them is responsible for business logic, which assembles common functions for all parts and is a middle layer for data transfer between algorithm implementation and its usage. The second component is the database access layer, which allows data insertion, modification and deletion. Both of these components are used in the web application, where the user can enter new image templates and manage old ones. In addition, areas that are sent to recognition and characters that need to be analyzed are selected using a web application user interface. The last part of the expert system is the web service, which takes the image as an input and returns character recognition results as an output by following the rules configured earlier in the web application.

The third result of the project was the development of two prototype applications to demonstrate the usage of the whole system. The first of these was an application for the Android operating system which aimed to detect lottery numbers from a smartphone camera capture and later compared these against winning numbers. This solution used the expert system as an analysis tool only. Algorithm implementation was used directly and results were compared against constant values gained earlier from the expert system. The second prototype was a simple web application that used the expert system web service to determine responses for an A4-format questionnaire with a fixed structure.

Conclusions

The goals set at the beginning of the project were successfully achieved. The new algorithm was developed and implemented. The expert system was completed with the possibility of using a convenient user interface to configure image template settings. Two prototype applications with image recognition functions were developed to test and demonstrate the usage of the preceding two results. Testing demonstrated that the solution developed in the thesis is accurate and fast enough to solve the initial problem.

During the course of the project, it turned out that due to increasingly larger image resolutions in smartphones, it makes sense to have all the image recognition logic within the application instead of using web service. This option is also supported by the solution developed in the master's thesis, by doing the initial analysis in the expert system and copying results to the application as constants. In the smartphone application, new image analysis is carried out by

integrated algorithm implementation, comparing constant values defined earlier. The same solution was also used when developing the first prototype Android application.

Future developments

The author of the master's thesis has planned to finish the Android application that was originally the prototype. In order to do this, the functionality that was described in the project but was not implemented due to the scope of the project, must next be implemented. This includes the application's design, querying the winning numbers and displaying animated results, in addition to other steps that must be taken to release application to the market. The long-term goal is to also use the solution developed in the thesis in other applications where conditions are similar and image recognition functionality can add value. In doing this, the scope would not be only for Android phones, but also for smartphones running the iOS and Windows Phone operating system. That is why the algorithm is implemented using the C++ programming language, making it easily adaptable to any platform.

Additional changes may come into consideration with the improvement of the algorithm. If it turns out at a later stage that there are problems with the accuracy of image recognition, additional rules can be added to improve the algorithm's analysis. It is also possible to release the expert system for public use, so that developers around the world can use image recognition capabilities when developing applications that need OCR functionality in limited circumstances, as in this project. But currently that will remain a long-term future plan which can be considered after the first additional developments are ready.

Kasutatud kirjandus

[1] H. Vallaste, „e-teatmik,“ [Võrgumaterjal]. http://www.vallaste.ee/. [Kasutatud 2 Märts 2014].

[2] „IT terminstandardi sõnastik,“ Eesti Keele Instituut, [Võrgumaterjal]. http://eki.ee/dict/its/. [Kasutatud 15 Aprill 2014].

[3] Smith, Ray, „Tesseract-ocr,“ Hewlett-Packard, Google, [Võrgumaterjal]. http://code.google.com/p/tesseract-ocr/. [Kasutatud 2 Märts 2014].

[4] R. Theis, „Tess-two,“ [Võrgumaterjal]. https://github.com/rmtheis/tess-two. [Kasutatud 2 Märts 2014].

[5] M. Negnevitsky, Artificial intelligence : a guide to intelligent systems, New York: Addison-Wesley, 2004.

[6] E. Borovikov, „A survey of modern optical character recognition techniques,“ 3 Mai 2004. [Võrgumaterjal]. http://www.academia.edu/3732087/A_survey_of_modern_optical_character_recognition_techniques. [Kasutatud 15 Aprill 2014].

[7] T. Kasar ja A. G. Ramakrishnan, „Alignment of Curved Text Strings for Enhanced OCR Readability,“ *International Journal of Computer Vision and Signal Processing*, kd. 3, p. 9, 2013.

[8] S. Z. Li ja A. K. Jain, Handbook of Face Recognition, Springer, 2011.

[9] V. Shrivastava ja N. Sharma, „Artificial Neural Network Based Optical Character Recognition,“ *Signal & Image Processing*, kd. III, p. 8, 2012.

[10] R. Smith, „An Overview of the Tesseract OCR Engine,“ 8 Juuli 2007. [Võrgumaterjal]. http://tesseract-ocr.googlecode.com/svn-history/r367/trunk/doc/tesseracticdar2007.pdf. [Kasutatud 15 Aprill 2014].

[11] T. Breuel ja DFKI, „ocropus,“ Google, [Võrgumaterjal]. https://code.google.com/p/ocropus/. [Kasutatud 16 Aprill 2014].

[12] M. Bryant, T. Blanke, M. Hedges ja R. Palmer, „Open Source Historical OCR: The OCRopodium Project,“ 2010. [Võrgumaterjal]. http://link.springer.com/chapter/10.1007/978-3-642-15464-5_72. [Kasutatud 16 Aprill 2014].

[13] ABBYY, „ABBYY Cloud OCR SDK,“ ABBYY, [Võrgumaterjal]. http://ocrsdk.com/. [Kasutatud 16 Aprill 2014].

[14] „Sudoku,“ Meriam-Webster, [Võrgumaterjal]. http://www.merriam-webster.com/dictionary/sudoku. [Kasutatud 20 Aprill 2014].

[15] Department of Computer Engineering, University of California at Santa Cruz, „Hough Transform for Lines and Curves,“ [Võrgumaterjal]. http://classes.soe.ucsc.edu/cmpe264/Fall06/Lec6.pdf. [Kasutatud 20 Aprill 2014].

[16] „Bitmap Image File (BMP), Version 5,“ [Võrgumaterjal]. http://www.digitalpreservation.gov/formats/fdd/fdd000189.shtml. [Kasutatud 20 Aprill 2014].

[17] Microsoft, „BITMAPFILEHEADER structure,“ 12 Juuni 2013. [Võrgumaterjal].

http://msdn.microsoft.com/en-us/library/dd183374(v=vs.85).aspx. [Kasutatud 20 Aprill 2014].

[18] Microsoft, „BITMAPV5HEADER structure,“ 12 Juuni 2013. [Võrgumaterjal]. http://msdn.microsoft.com/en-us/library/dd183381(v=vs.85).aspx. [Kasutatud 20 Aprill 2014].

[19] D. Lancaster, „Exploring the .BMP File Format,“ [Võrgumaterjal]. http://www.tinaja.com/glib/expbmp.pdf. [Kasutatud 20 Aprill 2014].

[20] Microsoft, „Introducing Visual Studio,“ [Võrgumaterjal]. http://msdn.microsoft.com/en-us/library/vstudio/fx6bk1f4(v=vs.100).aspx. [Kasutatud 20 Aprill 2014].

[21] „GIMP - The GNU Image Manipulation Program,“ The Gimp Team, [Võrgumaterjal]. http://www.gimp.org/. [Kasutatud 26 Aprill 2014].

[22] „SWIG,“ [Võrgumaterjal]. http://www.swig.org/index.php. [Kasutatud 20 Aprill 2014].

[23] Microsoft, „Microsoft SQL Server,“ [Võrgumaterjal]. http://msdn.microsoft.com/en-us/library/bb545450.aspx. [Kasutatud 27 Aprill 2014].

[24] SmartBear, „SoapUI,“ [Võrgumaterjal]. http://www.soapui.org/. [Kasutatud 27 Aprill 2014].

[25] Microsoft, „What Is Windows Communication Foundation,“ [Võrgumaterjal]. http://msdn.microsoft.com/en-us/library/ms731082(v=vs.110).aspx. [Kasutatud 27 Aprill 2014].

[26] happyuk, „Getting started with SWiG for interfacing between C# and C++ Visual Studio projects,“ 22 Veebruar 2013. [Võrgumaterjal]. http://www.technical-recipes.com/2013/getting-started-with-swig-interfacing-between-c-and-c-visual-studio-projects/. [Kasutatud 27 Aprill 2014].

[27] Tom FitzMacken, „Introduction to ASP.NET Web Programming Using the Razor Syntax (C#),“ Microsoft, 7 Veebruar 2014. [Võrgumaterjal]. http://www.asp.net/web-pages/tutorials/basics/2-introduction-to-asp-net-web-programming-using-the-razor-syntax. [Kasutatud 28 Aprill 2014].

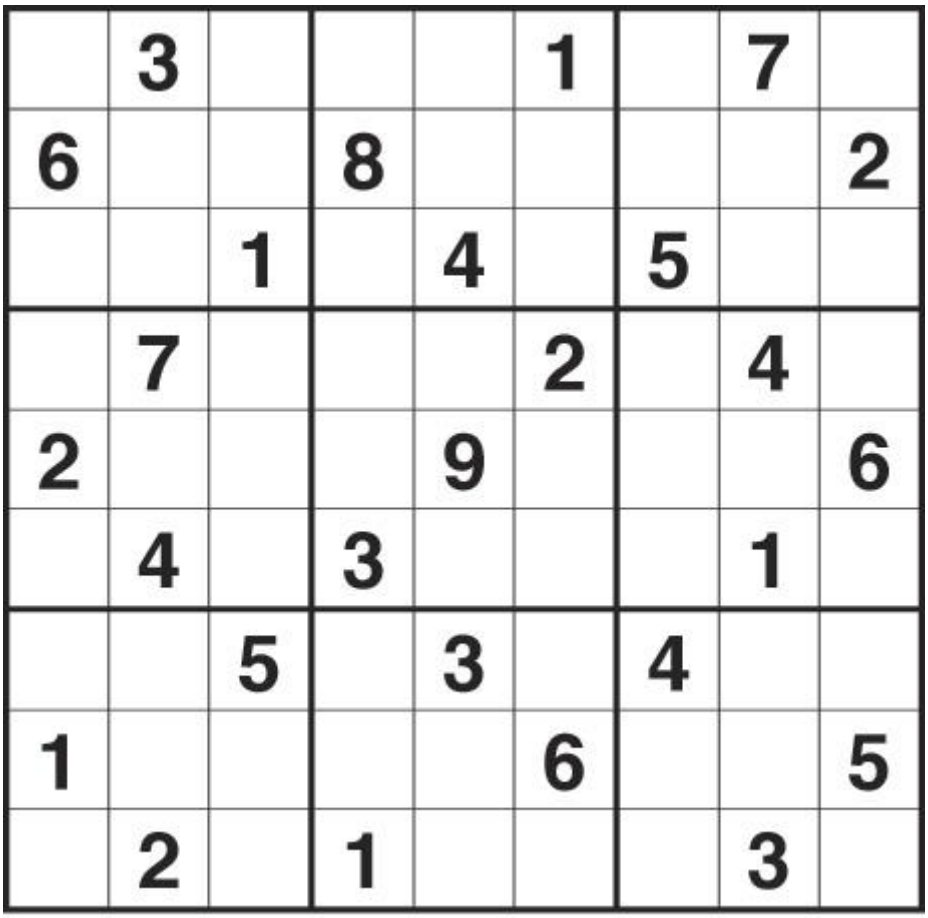
[28] Eesti loto, „Eesti loto,“ [Võrgumaterjal]. https://www.eestiloto.ee/. [Kasutatud 28 Aprill 2014].

[29] Tapmodo Interactive LLC, „The jQuery Image Cropping Plugin,“ 3 Veebruar 2013. [Võrgumaterjal]. http://deepliquid.com/content/Jcrop.html. [Kasutatud 28 Aprill 2014].

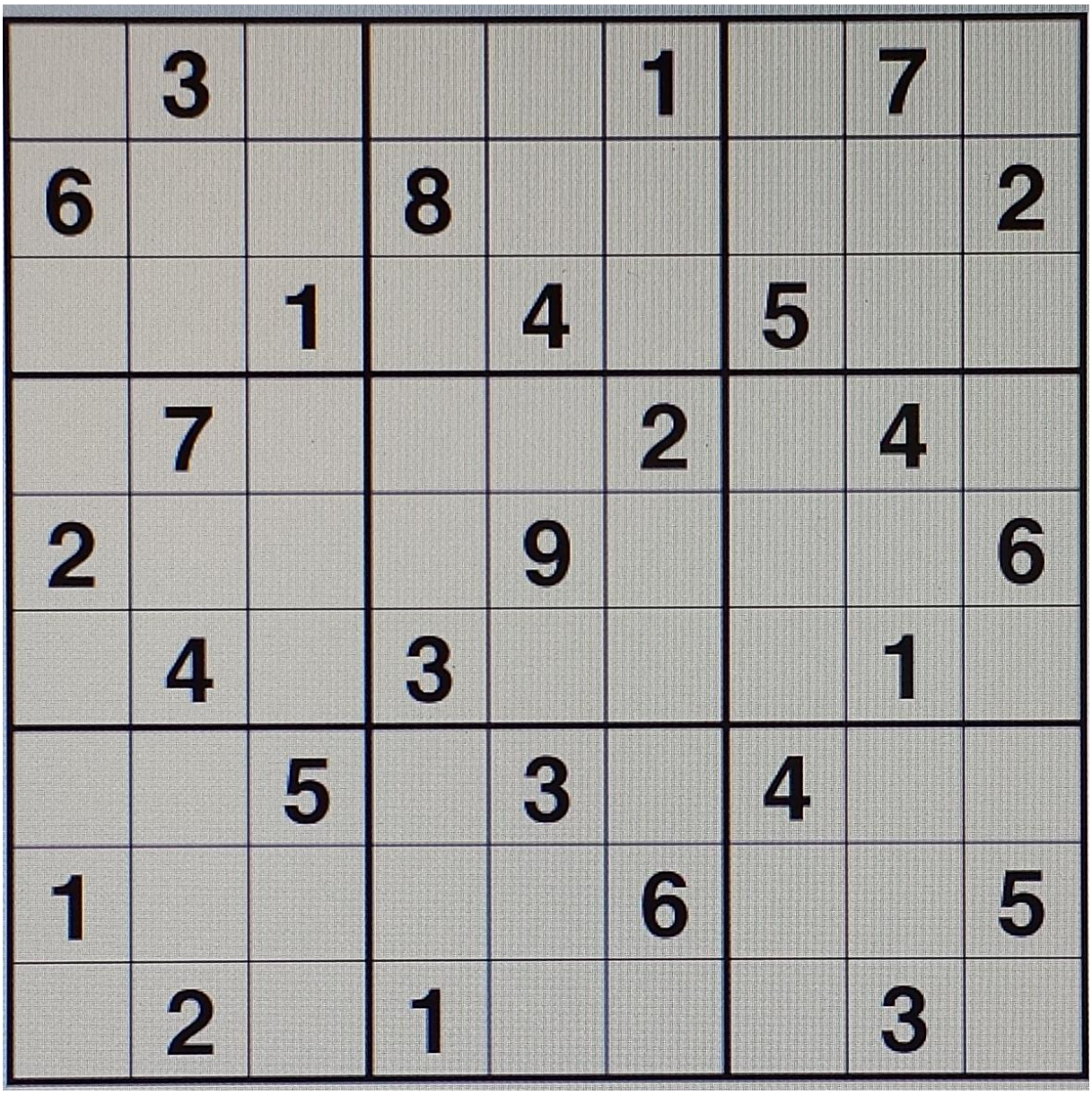
[30] „IR Remote Control,“ [Võrgumaterjal]. https://play.google.com/store/apps/details?id=ee.rautsik.irremotecontrol. [Kasutatud 29 Aprill 2014].

[31] Google, „Google Analytics Official Webpage,“ [Võrgumaterjal]. http://www.google.com/analytics/. [Kasutatud 3 Mai 2014].

Lisa 1. Arvuti abil joonistatud Sudoku väli



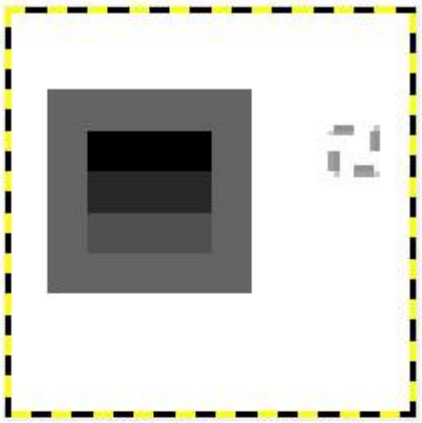
Lisa 2. Nutitelefoni pilt arvutil joonistatud Sudoku väljast



	3				1		7	
6			8					2
		1		4		5		
	7				2		4	
2				9				6
	4		3				1	
		5		3		4		
1					6			5
	2		1				3	

Lisa 3. Testsisendid algoritmi realisatsiooni testimiseks

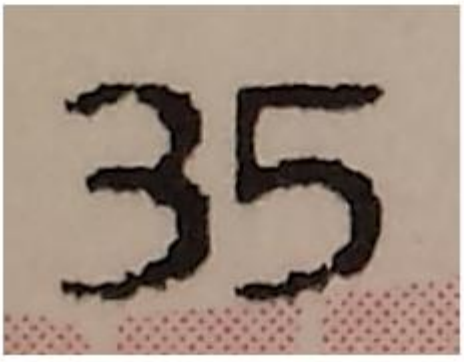
Testsisend 1



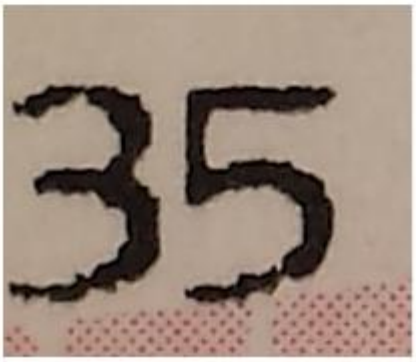
Testsisend 2.1

Testsisend 2.2

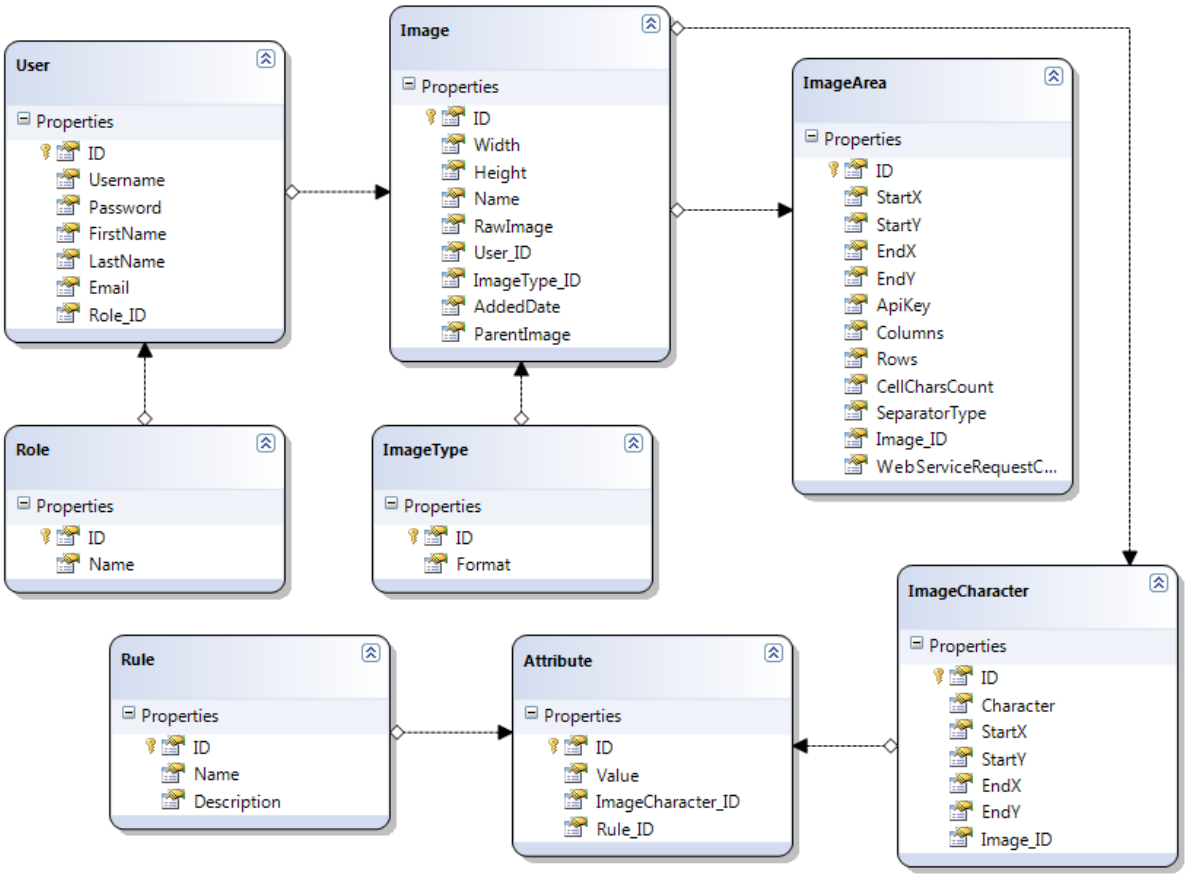
Testsisend 3.1



Testsisend 3.2



Lisa 4. Ekspertsüsteemi andmebaasi mudel



Lisa 5. Pseudokood parima tähemärgi vaste leidmiseks

```
INPUT malliTähemärkid, uuePildiAnalüüsiTulemused
DECLARE tulemused, erinevus
FOR EACH tähemärk IN malliTähemärkid
        SET erinevus TO 0
        FOR EACH reegel IN tähemärk
                erinevus +=Math.Abs(1.0 – (väärtus FROM reegel)/ (samaReegliVäärtus FROM
uuePildiAnalüüsiTulemused))
        END FOR
        ADD (tähemärk, erinevus) TO tulemused
END FOR
OUTPUT (tähemärk FROM tulemused WHERE MIN(erinevus))
```

Lisa 6. Veebirakenduse küsimustiku vorm

Küsimustik

*Palun kirjuta küsimuste 1,2, 3 ja 4 puhul kasti sobiv vastusevariant kasutades suurt trükitähte.

1. Kas sa omad nutitelefoni ?

A) Jah B) Ei

A

2. Kui vana sa oled ?

A) kuni 14 a. B) 15-20 a. C) 21-25 a. D) 26-40 a. E) üle 41 a.

C

3. Mitu palju aega sa päevas nutitelefonile kulutad (arvesse lähevad kõik tegevused välja arvatud traditsiooniline helistamine) ?

A) 0-30 min B)31 min -1 h C) 1 h - 2h D)2h - 4h E) rohkem kui 4h

E

4. Palun järjesta olulisuse järjekorras 3 uue telefoni ostul kõige olulisemat faktorid.

A) Hind B) Disain C) Tarkvara D) Kaamera kvaliteet E) Olemasolevad sensorid

B	D	A

5. Milliseid järgnevaid funktsionaalsusi nutitelefonil kasutad ? Palun tähista ristiga (+) mida kasutad tihti, punktiga (.) mida harva ja miinusega (-) see valik, mida ei kasuta üldse.

Kaamera (piltide jaoks)	+
Kaamera (video jaoks)	–
Suhtlusvahendid (FB chat, Skype jne)	+
Mängud	.
GPS rakendused	.
E-mail	+
Raamatute lugemine	–