Tallinna Ülikool
Digitehnoloogiate instituut
Informaatika

Vabavaralisel näotuvastusel põhinev autentimisrakendus

Bakalaureusetöö

Autor: Ain Arend
Juhendaja: Andrus Rinde

Autor: .....”.....”2017

Juhendaja: .....” .....”2017

Instituudi direktor: .....”.....”2017

Tallinn 2017

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina _____ (sünnikuupäev: _____)

(*autori nimi*)

1. annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   _____
   _____
   _____

   (*lõputöö pealkiri*)

   mille juhendaja on _____,

   (*juhendaja nimi*)

   säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, _____

allkiri ja kuupäev

Sisukord

Sissejuhatus

Näotuvastus on tegevus, millega inimesed puutuvad kokku igapäevaselt. See on üks faktoritest, mille abil loome seoseid, mis aitavad teisi inimesi ära tunda ning mis mõjutab tugevalt meie sotsiaalset käitumist. Tänapäeval on enamustel dokumentidel, millega me enese identiteeti tõendame, meie näoga pilt kõrval ning näotuvastamine ja selle abil seoste loomine on enamuse inimeste jaoks üsnagi harjumuspärane tegevus. Seega püütakse seda ka arvutite ning tehisintellekti juures rakendada, kuid selle puhul tuleb välja, et tegu on palju keerukama probleemiga kui esmapilgul võib tunduda.

Automaatse näotuvastuse valdkond hakkas arenema koos arvutite arenguga 1960. aastatel, kui teadlased alustasid tööd arvutite kallal, mis suudaksid tuvastada inimeste nägusid (Dowden, kuupäev puudub). Sellest ajast peale on näotuvastussüsteemid teinud suure sammu edasi ning tänapäeval on tegu vägagi populaarse valdkonnaga.

Kaasaja tehnikahiiud, nagu Microsoft, Google, Facebook ja Apple, panustavad väga suuri summasid tehisintelligentsi uuringutesse ning üheks selliseks näiteks on 2012. aastal toimunud Iisraeli ettevõtte Face.com, mis pakkus näotuvastust, ostmine Facebooki poolt 55 miljoni dollari eest (Constine, 2012).

Modernsed näotuvastussüsteemid suudavad kohati nägusid ära tunda juba sama hästi kui inimesed. Kuid seda ainult teatud tingimuste juures ning vajakajäämisi on veel palju, kuna näotuvastust mõjutavad äärmiselt palju erinevaid tegureid – nurk, emotsioon, valgustus, keskkond jms (Li & Jain, 2011).

Vaatamata puudustele, on näotuvastusel näiteks silmaiirise või sõrmejälje tuvastusega võrreldes, mitmed eelised. Seda saab teha suhteliselt kaugelt, ilma et inimene peaks peatuma ning oma sõrme millelegi panema või silma kuhugi fokuseerima. Lisaks puudub vajadus selle protsessi käigus inimesega suhelda või teda juhendada (NEC, kuupäev puudub).

Näotuvastussüsteemid on leidnud juba ka aktiivset rakendamist. Näiteks kasutatakse seda edukalt sotsiaalmeediates nagu Facebook ja Snapchat. Kui laadida Facebooki üles pilt, mis sisaldab inimeste nägusid, üritab süsteem automaatselt sildistada näo mõne kasutaja sõbraga. Snapchat kasutab samas nägudele erinevate filtrite lisamise võimalusi, rakendades selleks samuti näotuvastuse erinevaid meetodeid.

Samas ei ole näotuvastus ainult suurte tehnoloogiahiidude pärusmaa, vaid on olemas ka vabavaralisi lahendusi, mis võimaldavad igal arendajal näotuvastust oma rakendustes kasutada.

Antud töö eesmärgiks on anda ülevaade näotuvastuse olemusest, kirjeldada erinevaid olemasolevaid raamistikke näotuvastuse rakendamiseks ning luua vabavaralisel näotuvastusel põhinev autentimisrakendus. Loodav rakendus võiks olla aluseks arendajatele, kes soovivad oma rakendusse näotuvastuse abil autentimist lisada.

Autentimine tähendab kasutaja tuvastamist ja kontrollimist, kas ta on tegelikult ka see, kes väidab ennast olevat. Seega peab loodav autentimisrakendus peaks olema piisavalt lihtne, et seda saaks rakendada erinevates tingimustes ning samas piisavalt täpne, tagamaks süsteemi turvalisuse. Näotuvastust autentimisrakenduse juures plaanib autor rakendada ühe osana sisse logimise protsessist.

Töö autor valis antud teema isiklikust huvist masinõppe ja tehisintelligentsi vastu. Eestis küll tehakse antud valdkonnas tööd, näiteks on Tartu Ülikoolis kaitstud 2016 aastal inglise keelne bakalaureuse töö Zepp Uibo poolt, pealkirjaga „Näotuvastuseks treenitud tehisnärvivõrkude võrdlemine“, kuid eesti keelset kirjandust näotuvastuse valdkonnas sisuliselt ei õnnestunud leida, seega võiks antud töö näidata lugejatele, et näotuvastusega alustamine ei ole liiga keeruline ning julgustama arendajaid valdkonnaga rohkem tegelema.

Esimeses peatükis antakse ülevaade näotuvastust puudutavast teooriast, kuidas see toimib ja millised meetodeid kasutatakse. Teine peatükk keskendub näotuvastust pakkuvatele tööriistadele, lisaks pannakse paika kriteeriumid vabavaraliste näotuvastussüsteemide võrdlemiseks ning nendele põhinedes tehakse ka analüüs neljale konkreetsele vabavaralisele tööriistale. Lisaks antakse ülevaade tasulistest lahendustest ning nende võimalustest. Kolmas peatükk kirjeldab loodavat rakendust – selle nõudeid, arendamise protsessi kui ka testimist.

1. Näotuvastus

Käesolevas peatükis antakse ülevaade näotuvastusest. Automaatne näotuvastus, kui oma ette valdkond, hakkas arenema 1960. aastatel ning on muutunud tänapäeval vägagi pupulaarseks. Samas on tegu keerulise valdkonnaga, mille probleemid ulatuvad privaatsusest kuni keeruliste algoritmide kasutamiseni.

Alljärgnevates alapeatükkidest kirjeldatakse esmalt näotuvastuse süsteemide arengut läbi aja, seejärel teises alapeatükis tuuakse välja erinevaid aspekte, mis muudavad näotuvastuse rakendamise keerulisemaks. Kolmas alapeatükk räägib sellest, kuidas näeb välja näotuvastamise protsess. Viimases alapeatükis antakse ülevaade näotuvastuse metoodikatest ja mõningatest olulistest algoritmidest.

1.1. Näotuvastussüsteemide ajalugu

Tavalise inimeste jaoks on näotuvastus äärmiselt lihtne ja automaatne tegevus, arvutite jaoks on tegu aga keerulise probleemiga, milles on nad alles hiljuti jõudnud inimestega võrreldavate tulemusteni. Automaatne näotuvastus, kui valdkond ei ole väga vana, see hakkas arenema koos arvutite ning raalnägemise (ingl *computer vision*) ja mustrite tuvastamise (ingl *pattern recognition*) arenguga (Dowden, kuupäev puudub).

Näotuvastuse üheks esimeseks oluliseks verstapostiks ajaloos peetakse 1960ndaid, kuna sel ajal töötas välja Woodrow W. Bledsoe, esimese poolautomaatse näotuvastussüsteemi, mida kasutasid USA luureagentuurid. Süsteemi puuduseks oli see, et inimene pidi piltidelt tuvastama silmade, kõrvade, nina ja suu asukoha. Tarkvara põhines võimel eraldada kasutatavaid omadusi ning arvutada nende punktide vahelist kaugust ja suhet, mida seejärel omakorda võrreldi olemasolevate andmetega.

Järgmine samm automatiseerituse poole astuti 1970ndatel, kui Goldstein, Harmon ja Lesk lõid süsteemi, mis kasutas 21 spetsiifilist markerit, näiteks juuksevärv ja huulte paksus. Sarnaselt Bledsoe loodud süsteemile, leidis antud süsteem rakendust USA luureagentuurides. Ka selle süsteemi puhul oli vaja inimesepoolset panust, et süsteem suudaks teha arvutusi (Mayhew, 2015). Seega oli näotuvastuse teadusharu endiselt väike ning pigem niši ala.

Suurem samm näotuvastuse automatiseerimise poole, astuti 1990ndatel aastatel, kui näotuvastuse protsessis hakati rakendama matemaatilisi meetodeid. See areng oli märgilise

tähtsusega, sest selle abil näidati, et näo võimalikuks tuvastamiseks, juhul kui pilt oli õige nurga all ning süsteemi jaoks kohandatud, läks vaja vähem kui sadat parameetrit ja väärtust. Sellele järgnes omaväärtusvektorite kogumi (ingl *eigenface*) kasutamise protsessi teke, mis on teerajaja tänapäevasele näotuvastusele, tänu sellele protsessile, suudeti nägusid reaalajas tuvastada (Mayhew, 2015). Omaväärtusvektorite arvutamise protsessis kasutati vektoreid, et luua ruudukujulisi väikseid alasid – omaväärtusvektoreid – näost, mida kasutati identifitseerimiseks ja võrdlemiseks.

See avastus oli teerajajaks näotuvastuse populariseerimisele, tõstes huvi näotuvastuse kui valdkonna vastu, lihtsustades ja standardiseerides näotuvastuse protsessi (Dowden, kuupäev puudub). Esimene suurem avalikus kohas näotuvastuse kasutamine toimus 2001. aastal USA-s Super Bowli suurvõistluse käigus, kui turvakaameratest saadud pilte võrreldi kurjategijate politsei andmebaasis olevate piltidega. Sellega suudeti tuvastada 19 isikut, kelle puhul oli avatud protsess vahistamismääruseks.

Peale omaväärtusvektorite protsessi teket hakkasid näotuvastuse valdkonnaga seotud projektid saama rohkem rahastust, nii avalikust sektorist kui ka eraettevõtetelt. Peale seda on automaatne näotuvastus arenenud kiirelt edasi. Näotuvastussüsteemid on muutunud odavamaks, lihtsamaks ja kiiremaks, tänu millele on nad leidnud kasutust ka erasektoris. Tänapäevane automaatne näotuvastussüsteem toimib teatud tingimustes juba inimesega võrdväärsel tasemel, kuid siiski on üldiseid probleeme, mis vajavad lahendamist.

1.2. Probleemid näotuvastuses

Näotuvastussüsteemidel on endiselt palju erinevaid probleeme. Need probleemid jagunevad üldiselt kahte suurde kategooriasse – sotsiaalsed ja tehnilised probleemid. Mõlemad kategooriad tõstatavad keerulisi probleeme, mis raskendavad näotuvastuse kasutamist.

Sotsiaalsed probleemid on näiteks see, kus ja kuidas inimestele aktsepteeritavalt näotuvastussüsteeme kasutada. Tehniliste alla paigutuvad erinevad probleemid, mis on seotud näotuvastuse protsessiga – alates pildilt näo ja selle võtmefaktorite leidmisega, kuni näo võrdluseni, et isikut tuvastada. Samas on sotsiaalsed ja tehnilised probleemid ka paljuski omavahel seotud, näiteks võib tuua turvalisusprobleemi – kuidas ja millised andmed inimese kohta kuhugi salvestatakse.

1.2.1. Sotsiaalsed probleemid näotuvastusel

Näotuvastussüsteemi edukaks rakendamiseks on enamasti vajalik, et inimesed oleksid nõus tegema süsteemiga koostööd, see tähendab nad ei varja oma nägu ega takista kuidagi muud moodi süsteemi tööd. Mitte koostööd tegev inimene on näotuvastuse jaoks jällegi eraldi tehniline probleem, mis tõstab süsteemi keerukust märgatavalt (Li & Jain, 2011). Samas on hea süsteemi ülesehituse puhul võimalik tagada inimese koostöö, selleks peaks inimene teadma kuidas, miks ja millal süsteem teda jälgib.

Tänapäevastel näotuvastussüsteemidel on palju erinevaid võimalikke rakendamise kohti. Need süsteemid peavad suutma toimida ilma, et inimene peaks peatuma ja kuhugi pikemaks ajaks vaatama ning seejuures peavad süsteemid tagama näotuvastuse täpsuse. See omakorda loob võimaluse, et sellised süsteemid võivad toimida täiesti anonüümselt, nii et inimene ei ole süsteemi olemasolust teadlik, näitena saab tuua ühe sellise rakendamise kohana turvakaamerate süsteemid. Sellisel puhul jääb inimesel ära võimalus jälgimisest keeldumiseks. Seega on privaatsus oluline aspekt, millele tuleb mõelda näotuvastussüsteemide rakendamisel.

Kuigi enamus inimesi ei mõtle selle peale, tegelevad nad nägude tuvastamisega igapäevaselt. Seetõttu peaks olema näotuvastamise protsess inimese jaoks loomulik tegevus. Tegelikult ta seda ei ole, sest enamasti me ei mõtle teist inimest kohates selle peale, et meie aju on parajasti tegemas näotuvastust – see toimub meie jaoks automaatselt. 2015. aastal viidi Suurbritannias läbi 1000 poes käiva inimese seas küsitlus, kas tehnoloogiad, mis muudaksid peos käigu personaalsemaks, oleksid toredad või jubedad, paigutati erinevad näotuvastuse rakendused jubeda poolele (RichRelevant, 2015). See tähendab, et inimesed peavad selgelt mõistma, miks näotuvastussüsteeme kasutatakse ning milline on inimese jaoks süsteemi olemasolust saadav kasu. Vastasel juhul võib loodetav kasu pöörduda süsteemi jaoks hoopis kahjuks.

1.2.2. Tehnilised probleemid näotuvastusel

Näotuvastuse probleem on tehnilisest aspektist mittelineaarne, kus näotuvastamise õnnestumine sõltub iga pildi puhul paljudest erinevatest faktoritest. Neid faktoreid on erinevaid – valgus, kaamera nurk ja kvaliteet, näoilmed jms. Tehniliste probleemide all võib välja tuua neli üldisemat alamkategooriat:

- Suur varieeruvus näo omadusjoontes;

- Mitmemõõtmelisus;
- Väike võrdlusandmete kogum;
- Keeruline mittelineaarne mitmekesisus.

(Li & Jain, 2011).

Suur varieeruvus näo omadusjoontes tähendab seda, et näotuvastussüsteem peab olema suuteline tuvastama sama isikut, kellel on piltidel erinev näoilme või kes näiteks kannab habet. Lisaks peab ta suutma aru saada muudest inimlikest faktoritest nagu vananemine, haigused ja vigastused.

Inimene on **mitmemõõtmeline** objekt, samas on pildid kahemõõtmelised. Kuigi tänapäeva uusimad tehnoloogiad, nagu Facebooki DeepFace[1], suudavad rakendada normaliseerimise etapis 3D modelleerimist, mis tõstab märgatavalt näotuvastuse täpsust. Näo leidmist pildilt segab enamasti näotuvastuse jaoks pildil kaasas olev ebaoluline info, mida on samuti vaja töödelda.

Väikese võrdlusandmete kogumi all mõistetakse seda, et enamasti on näotuvastussüsteemidel inimese kohta, kelle nägu üritatakse tuvastada, väga väike treening- ja võrdlusandmete baas. See tähendab, et nendest andmetest ei suudeta luua piisavat üldistust eduka näotuvastuse tarbeks (Li & Jain, 2011).

Keeruline mittelineaarne mitmekesisus tähendab seda, et tänapäevased näotuvastus-süsteemid peavad olema suutelised tuvastama nägu piltidelt, kus taust, valgus, nurk ja muu ümbritsev ei ole seda soodustav, milleks varasemad lineaarsed meetodid ei olnud suutelised.

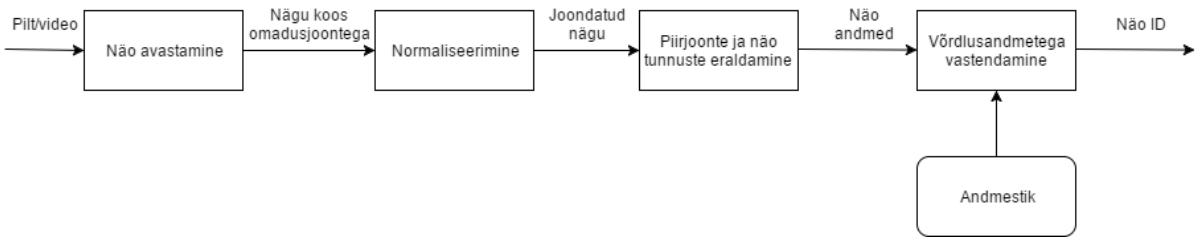
1.3. Näotuvastuse protsess

Näotuvastus on visuaalne mustrituvastuse probleem, kus nägu on kujutatud kolmemõõtmelise objektina, mida mõjutavad valgus, poos, emotsioonid ja muud faktorid, mis tuleb tuvastada tuginedes olemasolevatele võrdlusandmetele ehk piltidele. Enamus rakendusi kasutavad näost kahemõõtmelisi pilte, kuid rakendused, mis vajavad kõrgemat turvalisuse ja kindluse taset, peavad kasutama kolmemõõtmelisi pilte (Li & Jain, 2011).

Automaatne näotuvastuse protsess koosneb klassikaliselt neljast etapist. Esimene on kujutiselt näo avastamine, teiseks normaliseerimine, kolmandaks on piirjoonte ja tunnuste eraldamine ning viimasena vastendamine võrdlusandmetega (vt Joonis 1). Iga samm kirjeldab keerukust,

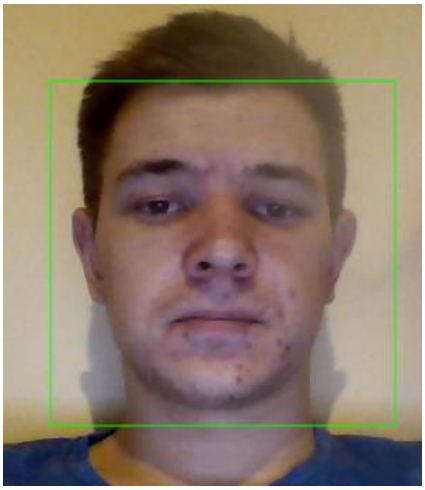
[1] https://pdfs.semanticscholar.org/6235/a7dfeecd449396ac03573bc117b54849397e.pdf

millega peab üks tavaline näotuvastussüsteem hakkama saama, et toimida edukalt (Introna & Nissenbaum, 2009).



Joonis 1. Näotuvastuse protsessi toimimine. Allikas: (Li & Jain, 2011)

1. **Näo avastamine** pildilt või videolt on inimese jaoks üsna lihtne tegevus, kuid ei ole seda arvuti jaoks. Arvuti peab suutma otsustada, millised pildi pikslid on osa näost, millised mitte (vt Joonis 2). Tavalise passipildi pealt, mille taust on selge, on seda lihtne saavutada, aga nii kui pildi taustale tekivad teised objektid, muutub see probleem kordades keerukamaks (Introna & Nissenbaum, 2009). Selleks, et videost nägusid avastada, kasutatakse mitmeid kaadreid ja näo jälgimise komponenti. Näo avastamise abil tehakse kindlaks, kas ja kus pildil asub nägu, kuid selle alla kuulub veel ka näo piirjoonte ja omaduste –silmad, nina, suu jms – asukoha kindlaks tegemine. Selleks, et nägu avastada, kasutatakse enamasti, kas piirjoonte või näo rühmitamise mooduleid. Veel uuritakse ka kogu pilti kui tervikut ruumi, kus nägu on esindatud ainult väikeses osaruumis, mis lihtsustab pildilt näo leidmist (Li & Jain, 2011). Näo avastamine on üldise tuvastamise protsessi mõistes äärmiselt oluline samm, ilma milleta puudub süsteemil nägu, mida võrrelda.



Joonis 2. Pildilt tuvastatud nägu..

2. Kui pildilt on nägu avastatud ning taustast eraldatud, tuleb nägu **normaliseerida** (vt Joonis 3)**.** See on protsess, mille käigus viiakse näo pilt süsteemi jaoks standardsele

kujule, et ta vastaks suuruselt, valguselt, poosilt jms süsteemi võrdlusandmetega (Introna & Nissenbaum, 2009). Normaliseerimist tehakse nii geomeetriliselt kui ka pildi põhiselt. Geomeetrilise normaliseerimise käigus teisendatakse pilt standardsetele mõõtmetele läbi pildi kärpimise. Pildipõhise normaliseerimise puhul standardiseeritakse pildi valguse ja hallskaala (ingl *grey scale*) omadusi (Introna & Nissenbaum, 2009). Normaliseerimine tagab selle, et tuvastatud näo pilti oleks võimalik võrrelda süsteemil olemasolevate võrdlusandmetega.

Joonis 3. Süsteemi jaoks normaliseeritud pildid.

3. **Piirjoonte ja näo tunnuste eraldamise** käigus eraldatakse normaliseeritud pildilt esile tõusnud oluline info, mida kasutatakse erinevate inimeste nägude eristamiseks. Selle käigus luuakse nendest andmetest biomeetriline mall, mis sisaldab matemaatilist kujutist näo oluliste omadusjoonte kohta. Saadud mall salvestatakse ning see on aluseks kõikidele näotuvastus võrdlustele (Introna & Nissenbaum, 2009).
4. Näo **võrdlusandmetega vastendamise** protsessis võrreldakse pildilt eraldatud näo omadusjooni süsteemis olemasolevate andmetega, kas ühe konkreetse või mitme pildiga. On olemas erinevaid mooduleid näo võrdlusandmetega vastendamiseks, sõltuvalt sellest millist vastust süsteemilt oodatakse. Selle protsessi lõpptulemuseks võib olla üks-ühele vaste „jah“ või „ei“. Vastuseks võib-olla ka vastete kogum, üks-mitmele suhte puhul, kus kõige ülemine vaste on „jah“ vastus, mille saavutamiseks peab süsteem ületama talle määratud kindluse määra, alumised vastused on võimalikud teised variandid, mille süsteem tuvastas. Kui süsteem ei ületa kindluse määra, ei suutnud süsteem antud nägu tuvastada ning vastuseks jääb „tundmatu“ (Li & Jain, 2011). Selle sammu suurim väljakutse on õige sarnasuse mõõdiku määramine, et otsustada, kas pildil oleva näo puhul on tegu sama näoga, mis on võrdlusandmetes.

Näotuvastamise täpsus sõltub tugevalt sellest, kuidas avastatakse kujutistelt näod ja kui hästi suudetakse leitud näod süsteemile kohaselt standardiseerida ning õiged andmed eraldada. Ehk nagu näha, on tegu keeruka probleemiga, milles edukaks olemisel on süsteemil palju erinevaid tegureid.

1.4. Näotuvastusalgoritmid

Kuna näotuvastus on keeruline probleem, siis on läbi aja üritatud sellele läheneda erinevate nurkade alt ning erinevatel viisidel. Kuid üldiselt on näotuvastamiseks algusest peale olnud kaks peamist lähenemisviisi, kuidas seda saavutada. Nendeks on:

- geomeetriline lähenemine;
- pildipõhine lähenemine.

(Tambasis, 2000).

Geomeetriline lähenemine on vanem ning põhineb näo omadusjoonte tuvastamisel ja eraldamisel, mis võib olla väga keeruline sõltuvalt valgusest, emotsioonidest, nurgast jms. Pildipõhisel lähenemisel võeti aluseks iga näo üldisemad mustrid, mis põhinevad inimese ja pildi iseloomulikel omadustel (Introna & Nissenbaum, 2009). Selle abil loodi mallid, millega suudeti erinevaid nägusid tuvastada (Tambasis, 2000).

Need kaks erinevat lähtepunkti on endiselt aluseks paljudele tänapäevastele näotuvastuse algoritmidele ning nende rakendamiseks on loodud palju erinevaid algoritme. Modernsed näotuvastussüsteemid peavad arvesse võtma ning hakkama saama väga paljude faktoritega – vananemine, nurk, valgus jne. Seega on väljatöötatud palju erinevaid algoritme, kõikide nende probleemide lahendamiseks. Järgnevalt annab autor ülevaate kolmest olulisest traditsioonilisest algoritmist, mida kasutatakse näotuvastamisel ning mis aitavad mõista tehnilist poolt, kuidas näotuvastamist teostatakse. Neid kasutatakse ka tänapäevastes näotuvastussüsteemides (näiteks osaliselt raamistikes, mida tutvustatakse peatükis 2. „17Näotuvastuse raamistikud“) koostöös teiste algoritmidega, tagamaks süsteemi võimalikult hea tulemuse.

1.4.1. Peamise komponendi analüüs

Peamise komponendi analüüsis (ingl *Principal Components Analysis*, lühend *PCA*) teisendatakse kahemõõtmeline pilt ühemõõtmeliseks vektoriks, mis jagatakse peamisteks komponentideks, mida nimetatakse omaväärtusvektoriteks. See meetod valib näo omadusjooned, mis eristuvad kõige rohkem ülejäänud pildist. Sellele järgneb lagundamine (ingl *decomposition*), mille käigus jäetakse kõrvale mitteolulised andmed, kuna 90% olulisest infost näo kohta, sisaldub 5-10% leitud komponentidest. Mis tähendab, et näo

identifitseerimiseks vajalik info, on ainult väike kild kogu pildil peituvast infost. (Introna & Nissenbaum, 2009).

Igat näo pilti esindab peamiste komponentide ehk omaväärtusvektorite kaalutud summa (omaduste vektor), mis on salvestatud ühemõõtmelises massiivis. Iga komponent esindab kindlat näo omadusjoont, mis võib olla esindatud originaalpildis. Saadud näopilti võrreldakse andmebaasis olevate võrdlusandmetega, leides nendelt piltidelt komponentide vektorite vahelise kauguse. Samas ei suuda PCA hakkama saada erinevate variatsioonidega, seega selleks, et PCA edukalt toimiks, peavad pildid saama süsteemi jaoks korrektselt normaliseeritud. (Introna & Nissenbaum, 2009).

1.4.2. Lineaarne diskriminandi analüüs

Lineaarne diskriminandi analüüs (ingl *Linear Discriminant Analysis*, lühend *LDA*) sarnaneb PCA meetodile, kuna kasutab samamoodi vektoreid ning üritab samuti nägu jagada erinevateks komponentideks. LDA klassifitseerib tundmatuid nägusid põhinedes treeningandmetele, milleks on süsteemile teadaolevad näod. See tehnika leiab vektorite abil näo omadusjooned, mis aitavad maksimaalselt nägu eristada teistest klassidest ning minimaliseeriks variatsioonid, mis esinevad ühe näo piires. Sellise klassifitseerimise abil suudab algoritm teha vahet erinevatel inimestel, samas tuvastada ka üht konkreetset inimest, kui tema näopiltidel ei ole toimunud suuremaid muudatusi (väiksemad muutused emotsioonis, nurgas, valguses jne) (Introna & Nissenbaum, 2009).

Selleks, et LDA töötaks hästi, on vaja normaliseeritud andmebaasi treeningandmetega, kus on esindatud iga näo kohta erinevad otsevaate võtted, erineva valgusega, näoilmega, taustaga jms. On loogiline, et mida suurem on andmebaas, seda paremini ka LDA toimib ning suudab võimaldada näotuvastamist ka väiksemate variatsioonidega tingimustes. Kuid üldiselt nagu ka PCA, vajab LDA tuvastatava pildi head standardiseerimist, et pilt oleks võimalikult treeningandmete sarnane.

1.4.3. Elastse graafide kobara võrdlemine

Elastse graafide kobara võrdlemine (ingl *Elastic Bunch Graph Matching*, lühend *EBGM*) erineb eelmisest kahes algoritmist selgelt, kuna selle põhimõtteks on fakt, et näos peitub palju erinevaid mittelineaarseid karakteristikuid, näiteks valgus ja näoilmed, mida lineaarsete meetodite nagu PCA ja LDA ei suuda arvesse võtta (Introna & Nissenbaum, 2009). EBGMi

puhul on näod esindatud graafidena, kus silmad, kõrvad jms on servadeks ning tippudeks on kauguse vektorid (Li & Jain, 2011).

EBGM meetod paigutab väikseid numbrite kollektsioone, mida nimetatakse Gabori filtriteks, erinevatele näo omadusjoonte aladele. Lisaks arvutatakse ka Gabori filtri kollektsioonidele väärtus, mida nimetatakse jugadeks (ingl *jets*). Neid kollektsioonide alasid saab vastavalt väiksematele variatsioonidele kohandada. Gabori filtrite kasutamise eeliseks on fakt, et sellega saab kõrvaldada enamuse valguse ja kontrasti muutusest tekkivad variatsioonid, olles samal ajal piisavalt robustne, et taluda väiksemaid muudatusi. Seda tänu sellele, et suudetakse paremini esindada peamisi näo omadusjooni, nagu silmad, nina ja suu (Introna & Nissenbaum, 2009). See algoritm on näotuvastussüsteemide võimet, tulla toime erinevate variatsioonidega poosis, nurgas ja emotsioonides, tugevalt parandanud.

2. Näotuvastuse raamistikud

Käesoleva töö üldisemaks eesmärgiks on luua vabavaralisel näotuvastusel põhinev autentimisrakendus. Antud peatükk annab ülevaate näotuvastuse raamistikest, seda nii vabavaralistest kui ka kommertslahendustest, et oleks täielik ülevaade selle kohta, millised erinevused või sarnasused on vabavaralistel ja kommertslahendustel. Lisaks teostatakse ka vabavaraliste raamistike võrdlus.

Esimeses alapeatükis pannakse paika vabavaraliste näotuvastus raamistike võrdlemise alused ja meetodid. Teises võrreldakse ja antakse ülevaade vabavaralistest näotuvastusrakendust. Viimases alapeatükis antakse ülevaade tasulistest lahendustest, nende võimalustest ja piirangutest.

2.1. Vabavaraliste raamistike võrdlemise meetod

Selleks, et vabavaraliste näotuvastusraamistike võrdlemine põhineks samadel alustel, tuleb paika panna konkreetsed tingimused ja kriteeriumid, mida iga raamistiku juures analüüsitakse. See tagab, et igast raamistikust antakse põhjalik ülevaade ning lihtsustab valiku tegemist.

Kaasaegsed näotuvastussüsteemid kasutavad oma protsessis enam kui kolme traditsioonilist algoritmi, mida kirjeldati peatükis 1.4 „Näotuvastusalgoritmid“. Lisaks keerukatele algoritmidele kasutatakse ka masinõpet, et parandada süsteemi võimet nägusid tuvastada ning paremini klassifitseerida. Seega on palju erinevaid aspekte, millega raamistike võrdlemise käigus arvestama peab.

Vabavaraliste näotuvastusraamistike võrdlemiseks toob autor välja viis põhilist kriteeriumi, millele pööratakse järgnevas võrdluses tähelepanu. Nendeks on:

- Toetatud platvormid ja programmeerimiskeel;
- Litsents;
- Näotuvastuse protsess;
- Andmestik ning saavutatud tulemused;
- Dokumentatsioon ning koodinäited.

Tarkvara juures on oluline teada, millistel **platvormidel** on võimalik seda kasutada, ehk millistele süsteemidele selle paigaldamist toetatakse. **Programmeerimiskeele** punkti juures

tuuakse välja, mis keeles on raamistik kirjutatud ning, juhul kui see on võimalik, tuuakse välja, millistes keeltes veel saab raamistikku kasutada, näiteks kui seda on võimaldatud kasutades raamistiku pakendamist (ingl *wrapper*) vastavasse keelde.

Kuna võrdlusesse võetakse ainult vabavaralised raamistikud, siis on oluline, et nende **litsentsid** oleksid sellele nõudele vastavad. Selleks peab litsents võimaldama tarkvara vabalt kasutada, jagada ning täiendada, selliseks näiteks on BSD[2] tüüpi litsentsid.

Näotuvastuse protsessi punkti all üritatakse lahti kirjutada raamistiku protsessi, kuidas jõutakse kujutiselt näo avastamisest võrdlusandmetega vastendamiseni ning ka seda, milliseid samme selleks läbitakse ja mis algoritme kasutatakse.

Kaasaegsed näotuvastussüsteemid kasutavad masinõpet oma näotuvastus mudelite treenimiseks erinevatel **andmestikel**, et tagada võimalikult hea **tulemus**. Seega on oluline teada, milliseid andmestikke raamistik oma mudelite treenimiseks kasutab ning kuidas ta seda teeb. Nimelt on siin oluline, et mudeli õpetamiseks kasutatud andmed erineksid täielikult andmetest, mida kasutatakse testides, kus tuvastatakse mudeli täpsust (Günther, Wallace, & Marcel, 2012).

Selleks, et võrrelda kui hästi süsteem suudab nägusid tuvastada, vaadeldakse süsteemi tulemusi LFW[3] (lühend inglise keelsest nimest *Labeled Faces in the Wild*) nägude andmebaasil. LFW on andmebaas, kuhu on kogutud 13 233 pilti 5 750. inimesest. Tulemuse jaoks peavad süsteemid tuvastama, kas talle ette antud pildipaaril on samad inimesed (Amos, Ludwiczuk, & Satyanarayanan, OpenFace: A general-purpose face recognition, 2016).

Lisaks on raamistike puhul veel oluliseks ka põhjalik **dokumenteeritus** ning korralikud õpetused ja **koodinäited**. Dokumentatsioon on oluline, sest ilma konkreetse juhiseta võib osutuda keerukate süsteemide, mida enamus näotuvastussüsteemid on, tööle saamine probleemiks. Veel tagavad korralik dokumentatsioon ja koodinäited esmase koha abi otsimiseks.

2.2. Vabavaraliste näotuvastus raamistike võrdlus

Näotuvastusega tegelevaid raamistikke, teeke (ingl *library*) ja API-sid (ingl *Application Programming Interface*) on palju, kuid enamus neist paraku on tasulised. Seega toob autor välja alljärgnevates alapeatükkides välja neli tööriista, mis on aktiivselt ülal peetud,

[2] https://et.wikipedia.org/wiki/BSD-litsents
[3] http://vis-www.cs.umass.edu/lfw/

vabavaralised ja võimaldavad näotuvastusega tegelemist. Lisaks teostatakse nende võrdlus, põhinedes eelmises peatükis kirjeldatud aspektidele.

2.2.1. OpenBR

OpenBR[4], mille täispikk inglise keelne nimi on *Open Source Biometric Recognition*, on raamistik, mille esialgne idee pärineb vajaduse järele uute masinõppe algoritmide kiiremaks prototüüpimiseks. Hetkel kirjeldab OpenBR ennast kui raamistikku, millega saab uusi meetodeid katsetada ning olemasolevaid algoritme täiustada, luues liidese erinevate rakenduste ning OpenBR-i raamistiku vahel. Lisaks on raamistikus rakendatud peatükis 1.4 „Näotuvastusalgoritmid“ kirjeldatud PCA ja LDA algoritmid, mida kasutatakse näotuvastus protsessis ning lisaks veel mõned teised algoritmid (OpenBR, kuupäev puudub).

Süsteemi on võimalik paigaldada kõikidele suurtematele operatsioonisüsteemidele – Windows, OS X, Android ja Linux ning OpenBR põhineb C++ **programmeerimiskeelel**. Seda on võimalik kasutada ka otse läbi käsurea liidese, mis on omakorda pakendatud kujul programmeerimiskeele C API. Lisaks on OpenBR pakendatud ka Pythoni jaoks, sisaldades kõiki esialgse C API versiooni funktsionaalsusi, kuid sel juhul toimib süsteem ainult Linuxi ja OS X operatsioonisüsteemidel. Veel on süsteemil C++ pistikprogrammide (ingl *plugin*) API, millega luuakse võimekus raamistikuga uusi algoritme testida ning olemasolevaid täiendada (OpenBR, kuupäev puudub).

OpenBR on avalikustatud Apache 2[5] **litsentsiga**. Seega võib raamistikku kasutada nii akadeemiliste kui ka erasektori projektide puhul, nii kaua kuni projektile nõuetekohaselt viidatakse (Klontz, Klare, Klum, Jain, & Burge, 2013).

Kuigi OpenBR on raamistik, mis võimaldab teha erinevaid katsetusi masinõppe ja biomeetrilise tuvastamisega, on selle loojad pannud suurt rõhku just raamistiku näotuvastuse võimekusele. Näotuvastamise **protsessis** kasutab OpenBR 4SF algoritmi, mis läbib näotuvastamiseks viis etappi:

1. Näo avastamine;
2. Normaliseerimine;
3. Matemaatiline esindamine (mille käigus kasutatakse PCA-d);
4. Tunnusjoonte eraldamine (mille käigus rakendatakse LDA-d ja uuesti PCA-d);

[4] http://openbiometrics.org/
[5] https://www.apache.org/licenses/LICENSE-2.0

5. Vastendamine.

(Klontz, Klare, Klum, Jain, & Burge, 2013).

Süsteemi **tulemuseks** LFW andmebaasi peal, on avaldatud keskmiseks õigesti arvatud piltide paariks ühe tuhande vale arvamise kohta, 12±2 pilti (Klontz, Klare, Klum, Jain, & Burge, 2013). Arvestades, et LFW sisaldab 13 233 pilti, pole see väga hea tulemus.

OpenBR-i **dokumentatsioon** on väga põhjalik. On olemas info selle kohta, kuidas raamistikku vastavalt platvormile paigaldada kui ka õpetused selle kohta, kuidas OpenBR-i ja näotuvastust kasutada. Lisaks on olemas iga toetatud programmeerimiskeele kohta korralik funktsionaalsusi kirjeldav dokumentatsioon. Koodinäited on leitavad raamistiku lähtekoodi */app/examples*[6] kaustas ning on kirjutatud programmeerimiskeeles C++.

Siiski on OpenBR üsnagi võimekas raamistik, mida saab kasutada väga paljudel eesmärkidel. Raamistikku on sisseehitatud korralik näotuvastuse algoritm, mille tulemused on teatud tingimustes võrreldavad kaubanduslike süsteemidega. Samas pole tegu puhtalt näotuvastuse probleemiga tegeleva raamistikuga. See tähendab, et tarkvaraga tuleb kaasa funktsionaalsust, mida ei pruugita süsteemide puhul, kus soovitakse rakendada ainult näotuvastusega seonduvat, kunagi kasutada.

2.2.2. Dlib

Dlib[7] on teek, mis sarnaselt OpenBR-ile, on kirjutatud C++ keeles ning mis sisaldab endas masinõppe algoritme. Selle esialgne loomise idee seisneski selles, et luua C++ jaoks vabavaraline teek, milles oleks võimalikult palju erinevaid tööriistu erinevate algoritmide ja meetodite kasutamiseks vastavas keeles (King D. E., 2009). Näotuvastus on üks meetoditest, mis on Dlib-i implementeeritud.

Teek kasutab enda kompileerimiseks CMake tööriista, seega on seda võimalik paigaldada erinevatele **platvormidele**, nii Windowsi, Maci kui ka Linuxi seadmetesse. Lisaks C++ **programmeerimiskeelele**, on Dlib-i võimalik kasutada ka Pythoni API-t, mis võimaldab kasutada enamuses C++ versiooniga sama funktsionaalsust.

[6] https://github.com/biometrics/openbr/tree/master/app/examples
[7] http://dlib.net/

Dlib toetub Boosti teekile, mis on kogum erinevatest C++ teekidest ning ka sellest lähtuvalt, on Dlib-il äärmiselt liberaalne Boost tarkvara **litsents**. See jätab süsteemi rakendamiseks vabad käed nii tasuta kui ka tasulistele lahendustele.

Tuvastamine selle raamistiku abil, koosneb järgmistest osadest:

1. Näo avastamine;
2. Näo joondamine;
3. Normaliseerimine;
4. 128-mõõtmelise vektoriga näo kujutise loomine (kasutatakse masinõpet ja sügavaid tehisnärvivõrke (ingl *deep neural networks*));
5. Nägude vastendamine (kasutatakse kobaras graafe).

(King D. , 2017).

Selle protsessi tulemusena suutsid teegi autorid saavutada võrdlusaluse LFW andmebaasi peal 99.38% **tulemuse**, mis tähendab, et süsteemile anti kaks näopilti ning ta suudab tuvastada 99.38% korral, kas tegu on sama inimesega. See tulemus on võrreldav teiste kaasaegsete näotuvastussüsteemidega, ka kaubanduslike, ning saavutamiseks treeniti tehisnärvivõrku kolme miljoni pildi peal, mis koguti erinevatest andmebaasidest, mille seas ei olnud LFW andmebaasi (King D. , 2017).

Kogu teeki katab väga põhjalik **dokumentatsioon**, nii C++ kui ka Pythoni versioonile, ning on olemas ka juhised, kuidas teeki paigaldada. Teegi C++ versioonile on kirjutatud ka väga palju **koodinäiteid**, sealhulgas ka põhjalik ja hästi kommenteeritud näotuvastuse rakendamise näide, mis on olemas ka Pythoni versioonile.

Dlib on väga hea tööriist, kuna teegiga tuleb kaasa väga palju masinõppe algoritme, seal hulgas modernne näotuvastusalgoritm. Samas eksisteerib Dlib-i puhul sarnane probleem, mis tuli välja ka OpenBRi puhul – tööriist pakub väga laialdase valiku võimalusi, kus näotuvastus on ainult üks väike osakene suurest pildist.

2.2.3. OpenCV FaceRecognizer

OpenCV[8], täispika inglise keelse nimega *Open Source Computer* Vision, puhul on tegu teegiga, mis võimaldab reaalajas raalnägemisega (ingl *computer vision*) tegelemist. Selle algse arendamisega tegeles Intel ning see põhines C keelel. Teek sisaldab nii klassikalisi kui

[8] http://opencv.org/

ka kaasaegseid raalnägemise ja masinõppe algoritme, millega saab näiteks objekte tuvastada või videos olevat inimtegevust klassifitseerida (OpenCV, 2016). Näotuvastuse jaoks on olemas eraldi moodul FaceRecognizer[9], mida tutvustati teegi 2.4 versioonis.

Programmeerimiskeelena kasutatakse OpenCV uusima versiooni 3.2. puhul, sarnaselt nii OpenBR-ile kui ka Dlib-ile, C++ keelt, samas on tarkvarale loodud liidesed kasutamiseks Pythoni, C ja Java programmeerimiskeeltes. **Platvormidest** töötab OpenCV kõikide suuremate süsteemide peal – Windows, Android, Linux ja OS X ning teek antakse avalikkusele kasutamiseks, varasemalt mainitud BSD 3-klausi versiooni **litsentsiga** (OpenCV, kuupäev puudub).

Näotuvastamiseks on võimalik FaceRecognizeri mooduli abil kasutada kolme erinevat meetodit: omaväärtusvektorite kogum (ingl *eigenfaces*) (kasutab PCA algoritmi), fisherfaces (kasutab LDA algoritmi) ja lokaalne binaarsete mustrite histogramm (ingl *Local Binary Patterns Histogram,* lühen *LBPH*). Iga mooduli **protsess** on erinev, seega ei saa üldistust luua. Esimesed kaks õpivad andmestikus oleva ühe näo kohta omadusjoonte võtmeväärtused kõikidelt saadavatelt piltidelt ning loovad selle põhjal omale matemaatilise esitluse näost. LBPH analüüsib igat pilti kui eraldiseisvat ja iseseisvat juhtumit, mille tõttu on see meetod lihtsam, kuid halvemate tulemustega (Arubas, 2016). Moodul eeldab, et andmestik ja võrreldav pilt on ühtemoodi normaliseeritud, ehk kasutatav pildisuurus on kõikidel piltidel sama (OpenCV, 2016).

Kahjuks pole FaceRecognizeri puhul võimalik ametlikult välja tuua süsteemi **tulemusi** LFW andmestiku peal. Töö autor leidis küll ühe kommertssüsteemi, mis tõi enda süsteemi LFW tulemuse võrdluse juures välja ka OpenCV tulemuse, milleks oli 49.6%, kuid mille puhul ei ole välja toodud, kuidas see saavutati, seega tuleb antud tulemusse suhtuda skeptiliselt (Sighthound, kuupäev puudub).

Üldiselt on OpenCV teek hästi **dokumenteeritud**, kuid FaceRecognizeri dokumentatsioon võiks olla põhjalikum, kuid samas kõik esmaselt vajalik on välja toodud. Moodulil puuduvad ametlikud **koodinäited**, kuid erinevaid versioone ja näiteid rakendustest, mis selle põhjal on ehitatud, on leitavad Internetist.

FaceRecognizeri moodul võib olla esmalt keeruline, millega alustada ning seda korrektselt tööle saada. Samas, kuna tegu on mooduliga OpenCV teegile, siis on võimalik ehitada väga võimas näotuvastussüsteem, kasutades ära kõiki teegi masinõppe ja raalnägemise võimalusi.

[9] http://docs.opencv.org/3.2.0/db/d7c/group__face.html

2.2.4. OpenFace

OpenFace[10] on näotuvastussüsteem, mis põhineb Dlib-i ja OpenCV meetoditel, lisaks kasutab süsteem veel sügavat tehisnärvivõrku. Erinevalt eelmistest väljatoodud süsteemidest, on selle puhul tegu tarkvaraga, mis keskendub ainult näotuvastuse probleemile (Amos, Ludwiczuk, & Satyanarayanan, OpenFace: A general-purpose face recognition, 2016).

Raamistik on kirjutatud Pythoni **programmeerimiskeeles** ning kasutab ka Torchi, mis võimaldab rakendada süsteemi otse protsessoris. **Platvormidest** on ametlikult toetatud OS X ja Linux. Sarnaselt OpenBR-ile, on OpenFace avaldatud Apache 2.0 **litsentsiga** (OpenFace, kuupäev puudub).

Näotuvastuse **protsess** koosneb järgmistest sammudest:

1. Näo avastamine (kasutab Dlib-i või OpenCV-d);
2. Joondamine ja normaliseerimine (kasutab Dlib-i ja OpenCV-d);
3. Omadusjoonte võtmeväärtuste leidmine (kasutab Dlib-i) ja tehisnärvivõrgu treenimine;
4. Näovastendamine.

OpenFace-l on oma tehisnärvivõrgu jaoks, töö kirjutamise ajaks, avalikustatud neli erinevat mudelit. Neist viimast kasutades on OpenFace autorid suutnud LFW andmestiku peal suutnud saavutada **tulemuseks** 92.9%, mis on päris hea tulemus (Amos, Accuracy and Neural Network Training Improvements, 2016).

Tarkvara **dokumentatsiooni** leht võiks olla põhjalikum, hetkel tundub, et sellele pole väga palju rõhku pandud, kuid tarkvara lähtekood on see-eest põhjalikult ja korralikult kommenteeritud. Samas on süsteemile tehtud väga põhjalikud ja korralikud **koodinäited**, mis katavad väga laia osa võimalikust rakendamise spektrist, sisaldades lihtsamaid näiteid piltide võrdlemisest kuni tehisnärvivõrgu treenimiseni välja.

OpenFace puhul on tegu väga konkreetselt tarkvaraga, mida luues peeti silmas ainult näotuvastuse probleemi ning süsteem omab ka väga häid koodinäiteid. Samas on süsteem ametlikult toetatud ainult OS X ja Linuxi süsteemidel, kuid see ei tohiks olla suur takistus, kuna süsteemid, millel OpenFace tugineb, toetavad kõiki suuremaid platvorme. See omakorda

[10] https://cmusatyalab.github.io/openface/

püstitab küsimuse, miks mitte kasutada kohe Dlib-i või OpenCV-d ning nende vastavaid näotuvastussüsteeme. Selle vastuseks on OpenFace tehisnärvivõrgu kasutamine, mis tagab süsteemi eduka näotuvastus võimekuse.

2.3. Ülevaade tasulistest lahendustest

Tasulised lahendused pakuvad näotuvastuse teenust enamasti rakendusliidese (ingl *Application Programming Interface*, lühend *API*) kujul. Seega jäetakse muretsemine, kuidas näotuvastus enda süsteemis tööle saada, teenusepakkuja hoolde, võimaldades API väljakutsumisega saada vastusena oma päringu kohta infot.

Samas luuakse sellise lahendusega oma süsteemile sõltuvus välise teenusega, mida ise ei kontrollita. Sellisel juhul võib korduda stsenaarium, mis juhtus 2012. aastal, kui Facebook ostis tol hetkel turu ainuliidriks olnud Face.com teenusepakkuja ning sulges selle teenuse kuu peale ostmist, jättes need, kes teenusest sõltusid, sisuliselt alternatiivita, kuna sel hetkel teisi võrdväärseid API pakkujaid ei olnud (Constine, 2012).

Käesoleva töö kirjutamise ajaks, on aga seis näotuvastuse API-t pakkuvate teenuspakkujate seas tunduvalt parem, kus oma versioone pakuvad nii suured tehnoloogia hiiud nagu Microsoft[11] ja Google[12] kui ka alustavad ning väiksemad ettevõtted nagu Face.com-i esimese alternatiivina loodud Lamba Labs[13] või Kairos[14], kuid variante on veel teisigi.

API kasutusele võtt loob küll sõltuvuse välise teenusega, kuid samas eemaldab see arendajalt kohustuse õppida, kuidas näotuvastus toimib, mil viisil seda on võimalik oma süsteemis rakendada ning töös hoida. Selle kõigega peab tegelema vastav teenusepakkuja ning arendaja peab looma ainult sideme oma rakenduse ning teenuse vahele ja tegema korrektseid päringuid API-le. See omakorda säästab arendaja jaoks olulist aega ning võimaldab tegeleda ainult oma tarkvara arendamisega.

Enamus teenusepakkujaid võimaldavad ka oma API-t mingi piirini tasuta katsetada või kasutada. Piirangud võivad olla nii ajalised kui ka mahus. Lisaks tavalisele nägude vastendamisele pakuvad enamus API-sid ka lisavõimalusi, nagu soo, juuste värvi, emotsiooni, vanuse ja tähelepanu tuvastamine. Läbi nende võimaluste, saab suhteliselt kergelt muuta oma rakendust interaktiivsemaks ning kasutaja jaoks lisa väärtust.

[11] https://www.microsoft.com/cognitive-services/en-us/face-api
[12] https://cloud.google.com/vision/
[13] https://lambdal.com/face-recognition-api
[14] https://www.kairos.com/features

API-d enamasti saadavad päringule vastuse JSON formaadis, seega ei sea nad arendaja jaoks programmeerimiskeeltes ja platvormides enamasti suuri piiranguid, piisab kui süsteem suudab API-ga ühenduse luua ning korrektselt päringu esitada. Küll aga tuleb nõustuda teenusepakkuja kasutustingimustega ja aktsepteerida fakti, et teenus võidakse sulgeda, ilma väga pika etteteatamisajata. Samuti pole enamasti teada, millised algoritmid on süsteemi taga, mis vastutavad selle eest, et nägusid avastada ning tuvastada või milline näeb välja kogu protsess, kuna valdavalt ei ole tegu avatud lähtekoodiga tarkvaraga, mis on ka loogiline, kuna üritatakse teenust müüa. Veel ei ole enamasti teada, mis tulemusi on süsteem saavutanud näiteks LFW või mõne muu andmebaasi peal. See-eest on enamustel API-del väga korralik dokumentatsioon ning korralikud näited, kuidas päringuid teha.

Võrreldes vabavaralisi lahendusi tasuliste versioonidega, siis suurim vahe on süsteemi kasutusele võtmise lihtsuses. Tasuliste API-de puhul, piisab sellest, kui loodav tarkvara suudab välise teenusega suhelda, vabavaraliste lahenduste puhul tuleb aga oma lokaalsesse masinasse tarkvara paigaldada ning vastavalt seadistada. Lisaks pakuvad enamus API-sid rohkem informatsiooni näotuvastuse tulemuse kohta – sugu, vanus jms. Vabavara puhul küll näiteks OpenBR võimaldab lisaks näotuvastusele ka vanuse ja soo tuvastamist, kuid teiste süsteemide puhul tuleb see kas ise luua või loota, et keegi selle süsteemile lisab.

Juhul kui arendaja eesmärgiks on oma tarkvarasse rakendada näotuvastuse võimalusi ning fakt, et selle toimimine põhineb välisel teenusel, ei ole probleem, siis on API kasutusele võtmine mõistlik. Kui aga ei soovita sõltuda teistest teenustest või ületatakse tasuta API kasutamise võimalusi ja pole soovi teenuse eest tasuda, tuleb pöörduda vabavaraliste lahenduste juurde.

3. Rakenduse loomine

Antud töö eesmärgi üheks osaks on vabavaralisel näotuvastusel põhineva autentimisrakenduse loomine. Selles peatükis antakse ülevaade töö käigus loodavast rakendusest, millised olid selle nõuded, kuidas seda loodi ning missugused olid tulemused.

Esimeses alapeatükis kirjeldatakse rakenduse olemust ning tehakse sellele nõuete analüüs. Järgmises alapeatükis teostatakse, põhinedes eelmises alapeatükis tehtud analüüsile ja peatükis 2.2 „Vabavaraliste näotuvastus raamistike võrdlus“ läbi viidud võrdluse tulemustele valik, millisele raamistikule rakendus tuginema hakkab. Kolmandas alapeatükis antakse ülevaade rakenduse arendamise protsessist ja viimases kirjeldatakse rakenduse testimist ning selle tulemusi.

3.1. Rakenduse analüüs

Lähtudes töö eesmärgist, tahab autor luua veebirakenduse, mis võimaldaks kasutaja autentimist, veebikaamera abil. Soov luua antud töö raames veebirakendus, tuleb autori isiklikust huvist ja tööst veebiarenduses.

Idee luua autentimisrakendus põhineb sellel, et tänapäeval on nutitelefonid vägagi populaarsed ja nende eesmised kaamerad on muutunud piisavalt heaks, et võimaldada neid kasutada näotuvastuses, mistõttu tekkis autoril huvi selle vastu, kuidas oleks võimalik lisada näotuvastus, kui suhteliselt lihtne viis kasutaja sisselogimiseks, veebirakendustesse.

Autentimine tähendab kasutaja identifitseerimist (mitte segi ajada autoriseerimisega, mis tähendab ligipääsuõiguste andmist) ehk tema tuvastamist ja kontrollimist, kas kasutaja on tegelikult see, keda väidab ennast olevat ning enamasti kasutatakse selleks kasutajanime ja parooli. Loodavas rakenduses peab kasutaja saama ennast registreerida, lubama süsteemil enda näost pilte teha ning seejärel saab kasutaja sisse logida, kasutades selleks kasutajanime, parooli ja oma nägu.

Selline rakendus võiks leida kasutust personaalsetel kodulehtedel või portaalides, olles ühe võimalusena osa kaheastmelises sisselogimise süsteemis. Kaheastmeline sisselogimine tähendab seda, et algul sisestab kasutaja oma kasutajatunuse ja parooli ning seejärel küsitakse kasutajalt veel midagi. Enamasti saadetakse kasutajale telefoni sõnumiga kood, mis tuleb

sisestada, et lõpetada sisselogimise protsess. Sellist süsteemi kasutab näiteks Google. Näotuvastuse võiks kasutusele võtta autentimisprotsessi teise astmena.

Kirjeldusest tulevad välja nõuded loodavale süsteemile. Need nõuded võib jagada kaheks – süsteemi ja kasutaja kohta käivateks nõueteks. Süsteemi kohta käivad nõuded on järgnevad:

- Peab toimima enam kasutatavate brauserite Google Chrome, Microsoft Edge, Firefoxi, Safari ja Opera viimaste versioonidega;
- Peab omama registreerimise ja sisse logimise funktsionaalsust;
- Peab kasutama näotuvastust sisse logimise teises astmes;
- Peab salvestama kasutaja andmed;
- Peab kasutajale selgitama, kuidas tema pilte kasutatakse.

Kasutaja kohta käivad nõuded on järgmised:

- Peab saama kontrollida õigust oma veebikaamera kasutamise üle;
- Peab saama hallata andmeid, mida tema kohta süsteemis hoitakse.

Kui rakendus vastab neile nõuetele, siis on tagatud, et loodav rakendus arvestab ka ptk 1.2 „Probleemid näotuvastuses“ välja toodud aspektidega privaatsuse ja kasutaja teadvustamise osas. Lisaks tagavad antud nõuded ka selle, et rakendus vastaks eelnevalt väljatoodud kirjeldusele.

3.2. Rakenduse loomiseks kasutatava raamistiku valimine

Loodava süsteemi põhjaks saava raamistiku valiku aluseks, sai peatükis 2.2 „Vabavaraliste näotuvastus raamistike võrdlus“ läbi viidud vabavaraliste süsteemide võrdlus. Eelmises alapeatükis viidi läbi rakenduse analüüs, et saada teada, millistele tingimustele peab süsteem vastama.

Põhinedes sellele infole, valis töö autor oma rakenduse aluseks OpenFace-i, mida kirjeldati lähemalt peatükis 2.2.4 „OpenFace“. Antud süsteem osutus valituks, kuna töö autoril on rohkem kogemust Python-is kui C++ programmeerimiskeeles. Lisaks, kuigi antud süsteem põhineb nii Dlib-i kui OpenCV tarkvarale (olid samuti võrdluses), siis on tegu süsteemiga, mis tegeleb konkreetselt ainult näotuvastuse probleemiga, kasutades Dlib-ist ja OpenCV-st ainult selle jaoks vajalikke osi.

Veel oli OpenFace kasuks otsustamisel oluliseks faktoriks koodinäited ning just OpenFacel on olemas koodinäide selle kohta, kuidas veebikaamerat kasutades näotuvastust tehakse. See näide saab olema väga oluliseks lähtepunktiks rakenduse loomisel, kuna sisaldab osaliselt loodava rakenduse jaoks vajalikke osi.

OpenFace puuduseks võib nimetada suurt hulka teeke, millele toetutakse. Seega võib süsteemi tööle saamine osutuda keerukaks, eriti töö autorile, kes kasutab Windowsi operatsioonisüsteemi, mille jaoks puudub ametlik paigaldusjuhend, kuid mida saab tuletada Linuxi ja OS X juhendist.

3.3. Rakenduse arendus

Autor kasutas loodava rakenduse arenduseks kose mudelit, kus kõige pealt teostati loodava rakenduse analüüs ning alles seejärel asuti programmeerima. Rakenduse arendus koosnes järgmistest sammudest:

1. OpenFace tarkvara paigaldamine;
2. Baasfunktsionaalsuse tööle saamine;
3. Reaalajas veebil põhineva näite tööle saamine:
4. Reaalaja veebi näite põhjal oma rakenduse arendamine;

Arenduse esimeseks sammuks oli OpenFace lokaalses masinas tööle saamine, mis osutus tagant järele vaadetes ka arenduse kõige keerulisemaks osaks. Selleks, et OpenFace paigaldada, järgis autor nende poolset juhendit[15]. Seal kirjeldatakse, kuidas tarkvara paigaldada Linuxi ja OS X süsteemidele, kuid töö autori isiklikus arvutis on kasutusel Windowsi 8.1 versioon, millele ka esialgu OpenFace paigaldada üritati. Peale esimest ebaõnnestunud katsetust, mis jäi CMake seadistamise taha seisma, otsustati tarkvara paigaldada Ubuntu virtuaalmasinasse. Virtuaalmasinas sai autor OpenFace tööle, järgides väga täpselt etteantud juhendit, otsides vahepeal juhendit kirjeldava teksti seest ja seadistuse failidest teeke, mida paigaldada, et süsteem toimiks.

Peale seda, kui süsteem töötas, läksid arenduse teine ja kolmas punkt lihtsalt, kuna OpenFace-ga kaasa tulnud näited töötasid korrektselt ja nendega probleeme ei tekkinud. Lisaks ei pidanud ise ka tehisnärvivõrke treenima, sest selle mudelid tulevad süsteemi poolt treenituna kaasa, mis muidu eeldaks näopiltide andmebaasi ja korraliku riistvara olemasolu.

[15] https://cmusatyalab.github.io/openface/setup/

Baasfunktsionaalsuse testimiseks prooviti piltide võrdluse näidet, kus käsurealt sai süsteemile ette anda kaks kogumit pilte, mida süsteem omavahel võrdles. Tulemusena kuvas süsteem käsureale, pildil olevate nägude kauguste suhte, mida madalam see oli, seda tõenäolisem oli, et tegu oli sama isikuga (vt Joonis 4). Selle näitega rohkem tööd tehes osutus, et kui suhe oli alla 0.8, oli mõlemal pildil sama isik.

```
ain@ain-VirtualBox:~/Documents/openface$ python ./demos/compare.py images/exampl
es/{lennon*,clapton*}
Comparing images/examples/lennon-1.jpg with images/examples/lennon-2.jpg.
  + Squared l2 distance between representations: 0.701
Comparing images/examples/lennon-1.jpg with images/examples/clapton-1.jpg.
  + Squared l2 distance between representations: 1.083
Comparing images/examples/lennon-1.jpg with images/examples/clapton-2.jpg.
  + Squared l2 distance between representations: 1.198
Comparing images/examples/lennon-2.jpg with images/examples/clapton-1.jpg.
  + Squared l2 distance between representations: 1.402
Comparing images/examples/lennon-2.jpg with images/examples/clapton-2.jpg.
  + Squared l2 distance between representations: 1.626
Comparing images/examples/clapton-1.jpg with images/examples/clapton-2.jpg.
  + Squared l2 distance between representations: 0.397
```

Joonis 4. Esimese katsetuse käivitamise tulemus.

Reaalajas veebil põhinev näide loob kohalikku masinasse Pythoniga veebiserveri ning kasutab kliendipoolel JavaScripti ning WebSocketit, et luua suhtlus brauseris toimuva ja veebiserveri vahel. Avades brauseris kohalikus võrgu (ingl *localhost*) määratud pordil, tuleb lahti veebileht, mis küsib õigust kasutada arvuti veebikaamerat. Selle puudumisel annab süsteem veateate, et ei suutnud veebikaamerat tuvastada ning lõpetab oma tegevuse.

Kui süsteem saab loa veebikaamerat kasutada, siis saab sisestada enda nime ning lülitada süsteemi treeningrežiimi, kus süsteem teeb automaatselt veebikaamerast edastatavast seisust pilte. Kui süsteemil on mingi kogus pilte tehtud, ning lülitades treeningrežiimi välja, suudab süsteem automaatselt piltidelt leida näo ja lisada sinna juurde sildiga, vastavalt treenimisest saadud infole, inimese nime. Oluline on aga teada, et kui on sisestatud ainult ühe inimese info, siis tuvastab süsteem kõiki teisi pildilt avastatud inimesi selle ühe sisestatud nimega.

Need kaks näidet aitasid autoril mõista, kuidas OpenFace praktikas toimib ning kuidas peaks lähenema loodavale rakendusele. Programmeerimist alustas autor sellest, et lõi sisselogimise ja registreerimise vaated, kasutades selleks Bootstrapi raamistikku, mis võimaldab kiiresti luua kohanduva kujundusega veebilehekülgi, ja selle ühte paljudest valmis koodinäidetest. Selle tulemuseks oli vähese vaevaga loodud viisakad sisselogimise ja registreerimise lehed.

Järgmise sammuna tuli integreerida OpenFace Pythoni serveripoolne loogika ja kliendipoolel toimuv. Selles toetuti palju reaalaja veebi näitele, tehes vajalikke muudatusi, et süsteem

toimiks vastavalt peatükis 3.1 „Rakenduse analüüs“ väljatoodud nõuetele. Keerulisemad muutused seisnesid selles, et kui näite puhul võidi süsteemi treenida järjest ükskõik kui palju inimesi ära tundma, siis loodud süsteemis saadetakse serverile ühe konkreetse isiku nimi ja kõik treeningu käigus serveri poolele saadetud pildid seotakse temaga. Lisaks pidi süsteem suutma meelde jätta, registreerinud kasutaja informatsiooni, mida näide ei tee. Selle jaoks kasutati LocalStorage-t, mis võimaldab salvestada informatsiooni kliendi brauserisse.

Sellisel kujul ka antud töö raames loodud projekti arendus lõppes. Seega ei saa kindlasti väita, et tegu oleks valmis versiooniga, mille saab oma süsteemi integreerida ning võimaldada kasutajatel näotuvastuse abil sisse logima hakata. Pigem on tegu rakendusega, mida võib kasutada lähtepunktina enda rakenduse jaoks spetsiifilise lahenduse väljatöötamisel, kuna süsteemi kood on üritatud hoida võimalikult lihtne, mille ühe näitena on ka LocalStorage kasutamine. Reaalses rakenduses ei tohiks olulist kasutaja infot sellisel kujul kindlasti hoida. Töö tulemusena valminud rakenduse kood on leitav GitHubist[16]

3.4. Rakenduse toimimine

Käesoleva töö kirjutamise seisuga töötab loodud rakendus järgnevalt. Kasutaja tuleb veebilehele ning temalt palutakse luba veebikaamera kasutamiseks. Kui süsteemile vastav luba antakse, saab kasutaja ennast registreerida. Selle peale salvestatakse tema kasutajanimi LocalStoragesse (taaskord, reaalne rakendus peaks salvestama kogu info ning soovitatavalt kuhugi mujale, kui LocalStorage, et seda saaks ka teistes masinates kasutada) ja kuvatakse kasutajale veebikaamera pilt ning palutakse, et kasutaja lülitaks sisse treeningrežiimi.

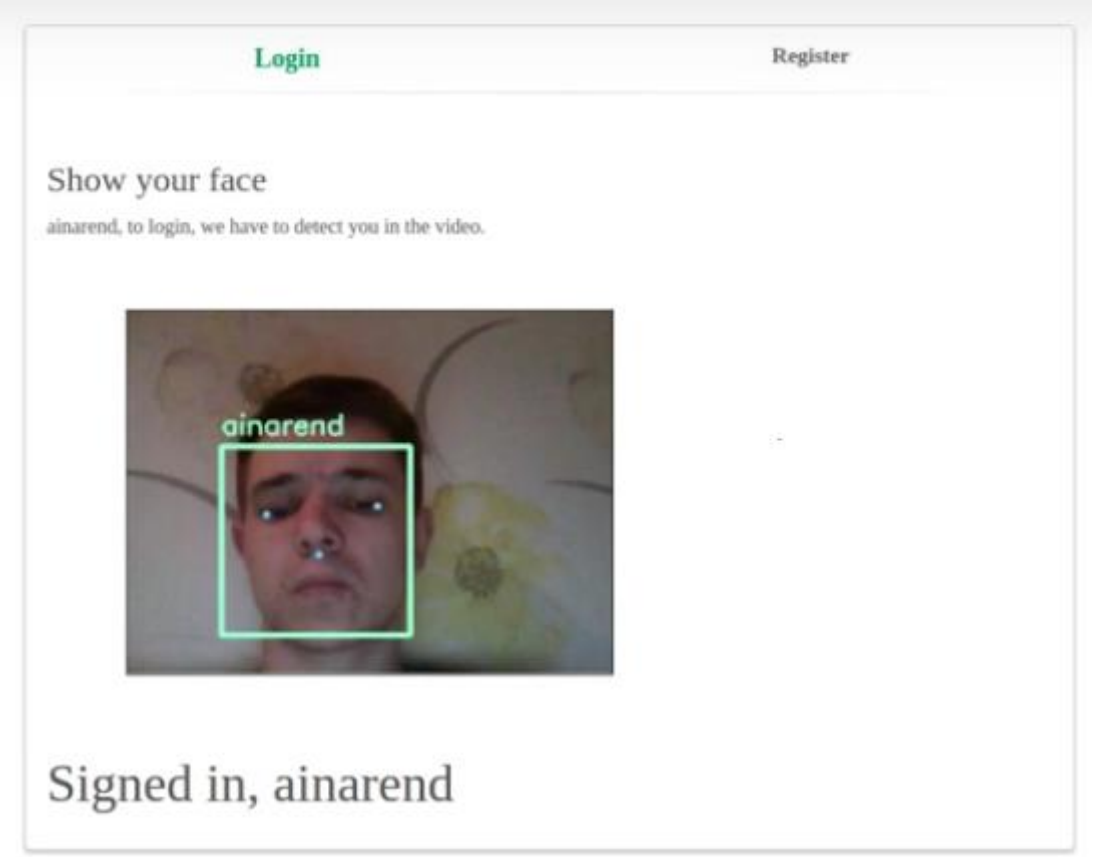
Sellest tuleb välja üks tuleviku edasiarenduse võimalus, kuna praegu teeb süsteem automaatselt, peale treeningrežiimi sisse lülitmist, taustal kasutajast pilte, ilma et kasutaja sellest aru saaks. Rakenduse nõuetes oli kirjas, et kasutaja peab saama oma andmeid hallata, kuid praeguse versiooni puhul kasutaja ei tea, kui ta just ei oska vaadata LocalStoragesse, milliseid pilte temast tehtud on ja kui palju neid on, sest süsteem teeb automaatselt neid järjest, kuni kasutaja treeningrežiimi välja lülitab. Tuleviku versioonis võiks kasutaja saada ükshaaval endast pilte teha ning tuleks võimaldada ka piltide haldamine.

Seejärel, kui kasutaja on süsteemi treeninud ning treeningrežiimi väljalülitanud, kuvatakse kasutajale veebikaamera pilt, kus tema näo ümber on roheline raam ning selle kohal tema kasutajanimi. Kasutaja võib jääda katsetama, kui hästi süsteem suudab tema nägu tuvastada

[16] https://github.com/ainarend/facerecognition

erinevate nurkade all, kuid sellega on registreerimine lõppenud, kuna treeningrežiimi teist korda enam sisse lülitada ei saa.

Sisselogimisel küsitakse kasutajalt kasutajanime ja parooli, kuid hetkel kontrollitakse ainult kasutajanime, kuna parooli haldamine peab tulema iga reaalse rakenduse enda poolt ning selle kontrolli pole keeruline süsteemile juurde lisada. Juhul kui sisestatud kasutajanimi eksisteerib salvestatud andmete seas, siis kuvatakse kasutajale tema veebikaamera pilt ning kui süsteem leiab, et veebikaamera pildil olev nägu on sama, mille kohta omatakse treeningu käigus saadud informatsiooni, siis logitakse kasutaja sisse (vt Joonis 5).



Joonis 5. Sisselogitud kasutaja vaade.

Kui vaadata üle peatükis 3.1 „Rakenduse analüüs“ rakendusele seatud nõuded, siis süsteem toimib edukalt kõikides suuremates brauserites, omab nõutud funktsionaalsust, kuid ei salvesta kogu kasutaja sisestatud informatsiooni. Veel pole ka loodud kasutajale selgitusi, kuidas temast tehtud pilte kasutatakse, samas erineb see iga reaalse rakenduse puhul ning oluline on teada, et reaalses kasutuses peaks seda tegema. Kasutaja saab kontrollida, kas lubab süsteemil oma veebikaamerat kasutada, kuid ei saa hallata seda, mis andmeid tema kohta

salvestatakse. Seega on rakendusel veel arenguruumi ning mille poole pürgida, kuid üldisem, antud töö eesmärgiks püstitatud ülesanne, sai siiski edukalt täidetud.

3.5. Rakenduse testimine

Rakendust testiti erinevates staadiumites ning kokku viie erineva isikuga. Kokku viidi läbi kaks testimise korda, esimest korda kui saadi tööle reaalajas veebipõhine näide ja teistkorda kui toimima saadi rakenduse esimene versioon.

Esimest korda testiti seda, kui hästi suudab OpenFace nägu tuvastada, kasutades reaalaja veebinäidet. Selle jaoks lisati süsteemi kolm inimest ja lasti teha igast ühest pilte. Esimesel inimesel lasti teha viis, teisel 8 ja kolmandal 12 pilti. Pilte lasti teha nii, et oleks varieerumist nurgas, emotsioonis, valguses, kauguses. Pilte tehti Asuse sülearvuti sisse-ehitatud veebikaameraga ning tulemuseks oli see, et kolmanda isiku tundis süsteem ära umbes 2m kauguselt kaamerast ja ka erinevate nurkade all, nii kaua kuni oli näha tuvastuseks vajalikud punktid (silmade nurgad, huuled). Esimest inimest, kellest tehti viis pilti, tuvastas süsteem enamjaolt samuti korrektselt, kuid liikumise peal mitte nii hästi, kui kolmandat või teist inimest. Sooline erinevus testkasutajete puhul näotuvastuse juures süsteemile probleeme ei valmistanud, pigem oli oluliseks faktoriks see, kui palju häid pilte näost treenimise faasis saadi. Optimaalse kogusena tundub, et piisab kui süsteemil on olemas 10 korralikku pilti.

Teisel testimise korral osales samuti kolm inimest, kellest üks osales ka esimesel testimisel. Seekord seletati osalejatele üldiselt kogu protsess, kuidas süsteem toimib ning lasti neil see omal käel läbi teha. Testijad tõid välja suurima puudusena, et süsteem ei kuvanud vigade või õnnestumiste kohta teateid, kui liiguti ühest etapist teise. Testkasutajale, kes osales ka esimeses voorus, jäi segaseks see, miks treenimise käigus enam loodavaid pilte ei näe, nimelt näites kuvati välja ka treenimise käigus tehtud pildid, kuid loodud süsteem seda ei tee. Seega süsteemil on veel arenguruumi, et olla kasutajasõbralikum, kuid üldiselt oli see, kuidas protsess välja nägi, testijate jaoks arusaadav.

Sama testikorra käigus testiti ka seda, kui lihtne on süsteemi ära petta, kuvades näiteks inimese pilti telefoni ekraanilt. Selline petmine toimis edukalt, kuna süsteem teeb videovoost pilte ning seejärel otsib pildilt nägude kohta informatsiooni. Seega pole hetkel antud kujul autentimisrakenduse kasutamine veel turvaline. Ühe võimaliku viisina, kuidas vältida seda, et esinetakse kellegi teisena, pildi või video salvestuse abil, on meetod, mis on kasutusel näiteks BioID API juures ja selleks on näo elavuse tuvastamine (ingl *face liveness detection*). See

tähendab, et süsteem annab inimesele konkreetseid juhiseid, et liigutada nägu vasakule-paremale või üles-alla ning andes neid juhiseid juhuslikult, tagatakse see, et pildi või video salvestusega ei ole võimalik süsteemi ära petta (BioID, kuupäev puudub). Konkreetset meetodit, mis võimaldaks vabavaraliste süsteemidega seda saavutada, pole veel loodud, kuid töö autor peab võimalikuks, et seda on võimalik saavutada, kasutades Dlib-i või OpenCV teeke.

Testimise käigus leiti süsteemist konkreetseid puudujääke, mida peaks võtma arvesse rakenduse tuleviku versioonides, tagamaks et vabavaralist näotuvastust saaks edukalt kasutada autentimise protsessis. Samas andis testimine kindluse, et kasutajate jaoks ei olnud nende nägu sisselogimiseks kasutav süsteem liialt võõrastav, kui nad olid teadlikud, kuidas süsteem toimib.

Kokkuvõte

Käesoleva töö eesmärgiks oli luua ülevaade näotuvastuse toimimisest, kirjeldada erinevaid vabavaralisi tööriistu, mis võimaldavad näotuvastust rakendada ning luua valitud tööriista põhjal autentimisrakendus.

Eesmärgi saavutamiseks tehti esimeses peatükis ülevaade näotuvastuse teooriast, milline on valdkonna ajalugu, erinevad probleemid näotuvastuse rakendamisel ning seejärel uuriti, kuidas näotuvastus konkreetsemalt toimib. Teises peatükis uuriti erinevaid tööriistu, mis võimaldavad näotuvastust oma rakendusse lisada. Selle tarbeks pandi paika konkreetsed kriteeriumid, mida analüüsiti iga välja toodud vabavaralise raamistiku juures. Lisaks tehti ülevaade ka tasulistest võimalustest, millest mugavaim variant on API-de kasutamine. Kolmas peatükk kirjeldab OpenFace tarkvaral põhineva rakenduse loomise protsessi, alates nõuetest kuni lõpptulemuse testimiseni välja.

Töö pea tulemusena valmis esmane versioon näotuvastust kasutavale veebipõhisele autentimisrakendusele, seega sai töö eesmärgiks olnud vabavaral põhinev autentimisrakendus loodud, kuigi sellel on väga selgeid puudujääke, mis tuuakse ka töö käigus välja. Rakenduse näol on hetkel tegu pigem lähtepunktina arendajatele, kes soovivad näotuvastust enda rakenduses autentimiseks kasutada, kuna palju vajalikku funktsionaalsust, mis ühel korralikul autentimisrakendusel olema peab, on veel puudu.

Antud temaatikal võiks edasi uurida, millised on võimalused vabavaraliste näotuvastus tööriistadega rakendada näo elavuse testimist, kuna hetkel on süsteemi liiga lihtne ära petta, et seda saaks kasutada reaalses lahenduses. Tegelikult oleks näotuvastus kaheastmelises autentimissüsteemis hea alternatiiv senistele lahendustele. Konkreetse rakenduse raames tuleks jätkata rakenduse kasutajamugavuste parandamisega, eriti just andmete salvestamise ja kasutajale kuvamise osas.

Summary

Authentication Application Based on Open Source Face Recognition

The goal of this thesis was to create an overview of how face recognition systems work, describe and compare open source toolkits, that allow to implement these systems and to create an authentication application based on an open source toolkit.

To fulfill this goal, in the first chapter an overview to the theory of face regonition is given – what is the history of the field, which are the problems that prevent the systems to be used widely and then more specific look is taken at how traditionally face recognition process works and what algorithms are used. The second chapter describes different tookits, both open source and commercial, that help implement face recognition. To compare open source toolkits a set of criteria rules is put in place, which are analyzed separetly with each open soure solution. Third chaper gives an overview of what it takes to build an open source authentication application, from the it's description to testing of the finished work.

The main result of this thesis work is the alpha version of the open source web based authentication application, which means that the goal of creating an open source authentication application is reached, even though there are several flaws in the software, that are clearly stated also in the thesis. This measn that the software should be taken as a starting point for developers who look to implement face recognition based authentication into their own applications. That is because a lot of key functionality that a good authentication application needs, currently lacks from the software.

Further studies on the topic should focus on implementing the face liveness detection with open source solutions, because currenty it is too easy to fool the authentication system, for it to be used in real applications, but there is potential for this system to be a good alternative in the two step authentication systems. More specificly for this current software, development and research should continue on improving user experience, handling of user data in process of saving it and also displaying it to the end user.

Kasutatud kirjandus

Amos, B. (19. jaanuar 2016. a.). *Accuracy and Neural Network Training Improvements*. Allikas: GitHub: http://bamos.github.io/2016/01/19/openface-0.2.0/

Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (2016). *OpenFace: A general-purpose face recognition.* Pittsburgh: CMU-CS-16-118, CMU School of Computer Science. Allikas: OpenFace: https://cmusatyalab.github.io/openface/#overview

Arubas, E. (6. aprill 2016. a.). *Face Detection and Recognition (Theory and Practice)*. Allikas: Eyal Arubas: http://eyalarubas.com/face-detection-and-recognition.html

BioID. (kuupäev puudub). *Face Liveness Detection*. Allikas: BioID: https://playground.bioid.com/BioIDWebService/LiveDetection

ccv. (kuupäev puudub). *Index*. Allikas: ccv: http://libccv.org/

Constine, J. (9. juuli 2012. a.). *Facebook Acq-Retired Face.com, But Here's Why It Will Bank By Reviving Its Facial Recog API*. Allikas: TechCrunch: https://techcrunch.com/2012/07/09/facebook-facial-recognition-api/

Dowden, J. (kuupäev puudub). *Facial Recognition: The most "Natural" Forensic Technology*. Allikas: Evidence Technology Magazine: http://www.evidencemagazine.com/index.php?option=com_content&task=view&id=1344

Günther, M., Wallace, R., & Marcel, S. (2012). *An Open Source Framework for Standardized Comparisons of Face Recognition Algorithms.* Berlin: Springer.

Introna, L. D., & Nissenbaum, H. (2009). *Facial Recognition Technology: A Survey of Policy and Implementation Issues.* NY: Center for Catastrophe Preparedness and Response, NYU.

King, D. (12. veebruar 2017. a.). *High Quality Face Recognition with Deep Metric Learning*. Allikas: Dlib: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html

King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, 1755-1758.

Klontz, J. C., Klare, B. F., Klum, S., Jain, A. K., & Burge, M. J. (2013). Open Source Biometric Recognition. *Biometrics: Theory, Applications and Systems*.

Li, S. Z., & Jain, A. K. (2011). *Handbook of Face Recognition Second Edition.* London: Springer-Verlag.

Liuliu. (14. detsember 2014. a.). *Readme.* Allikas: GItHub: https://github.com/liuliu/ccv/blob/stable/README.md

Mayhew, S. (15. jaanuar 2015. a.). *History of Biometrics*. Allikas: Biometric Update: http://www.biometricupdate.com/201501/history-of-biometrics

NEC. (kuupäev puudub). *Face Recognition*. Allikas: NEC: http://www.nec.com/en/global/solutions/biometrics/technologies/face_recognition.html

OpenBR. (kuupäev puudub). *Overview*. Allikas: OpenBr: http://openbiometrics.org/docs/index.html

OpenCV. (23. detsember 2016. a.). *FaceRecognizer*. Allikas: OpenCV: http://docs.opencv.org/3.2.0/db/d7c/group__face.html

OpenCV. (23. detsember 2016. a.). *Introduction*. Allikas: OpenCV: http://docs.opencv.org/3.2.0/d1/dfb/intro.html

OpenCV. (kuupäev puudub). *About*. Allikas: OpenCV: http://opencv.org/about.html

OpenFace. (kuupäev puudub). *Overview*. Allikas: OpenFace: https://cmusatyalab.github.io/openface/#overview

RichRelevant. (mai 2015. a.). *Infographic: Creepy or Cool?* Allikas: RichRelevance: http://www.richrelevance.com/insights/infographic-creepy-cool-uk/

Sighthound. (kuupäev puudub). *Face Recognition Performance*. Allikas: Sighthound: https://www.sighthound.com/technology/face-recognition/benchmarks/pubfig200

Tambasis, G. (2000). *Neural Network-based on Face Recognition, using ARENA Algorith.* Surrey: University of Surrey.