TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Pärlin Luhila 142361 IAPB

SISUPÕHIST PILDIOTSINGUT KASUTAV LOOMADE E-VARJUPAIK

Bakalaureusetöö

Juhendaja: Martin Rebane
MSc

Tallinn 2017

Autorideklaratsioon

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

Autor: Pärlin Luhila

22.05.2017

Annotatsioon

Käesoleva töö põhieesmärgiks on kasutada sisupõhist pildiotsingut sarnaste loomade leidmiseks. Sisupõhine pildituvastus võimaldab eraldada pildilt vajalikud andmed ning moodustada neist seda iseloomustav omadusvektor. Arvutades omadusvektorite vahelised kaugused, saab määrata piltide sarnasuse – mida väiksem kaugus, seda sarnasem pilt. Kuna pildilt eraldatavaid omadusi on mitmeid, analüüsiti töö käigus, millised antud ülesande lahendamiseks kõige paremini sobivad. Selle põhjal loodi veebiliides pildilt omaduste eraldamiseks ja sarnaste loomade otsinguks.

Töö käigus valmib ka veebileht, mis võimaldab koondada erinevate Eesti loomade varjupaikade loomade info. Eestis on varjupaikades tuhandeid loomi ning umbes 80% lemmikloomadest on kiibistamata. Seetõttu on kadumise korral looma leidmine päris keeruline. Rakendus kasutab loodud sisupõhise pildiotsingu veebiliidest ning võimaldab kasutajatel nii pildi kui erinevate parameetrite alusel otsida kas enda kadunud või näiteks unistuste looma.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 6 peatükki, 21 joonist, 2 tabelit.

Abstract

An Animal Shelter Application Using Content-Based Image Retrieval

The purpose of this thesis is to create an application using content-based image retrieval (CBIR) for finding similar animals. CBIR allows to extract different features from images, eg. color, shape and texture. The combination of those features forms the feature vector. When comparing the feature vectors of two images, it is possible to tell how similar they are. For that purpose the distance between vectors is found – the smaller the distance, the more similar the images. Since there are multiple ways of extracting features and many different features to extract, it is analyzed which suit best to solve the present problem of similar animals. A web interface is created for extracting features and enabling search for similar animals.

There are thousands of homeless animals in Estonia and around 80% of our pets are not microchipped. In case a pet goes missing, it is difficult to find them. This thesis sets the target to create an application which would raise awareness, popularize taking pets from shelters and simplify finding unmicrochipped lost pets. The application allows to consolidate information about animals in different Estonian shelters and allows the users to choose their favourites. The most important part of the application is the search. The user can enter an image and/or set different parameters about the animal, like age, species, location, gender, done procedures and the date since it is sheltered. The application is not completely finished and has many ways to improve to allow users more possible actions and make the process as comfortable as possible for animal shelters.

The thesis is in Estonian and contains 32 pages of text, 6 chapters, 21 figures, 2 tables.

Lühendite ja mõistete sõnastik

CBIR	*Content Based Image Retrieval.* Sisupõhine pildiotsing – pildi otsimiseks kasutatakse märksõnade asemel pildilt eraldatud informatsiooni värvi, tekstuuri ja kuju kohta.
GLCM	*Grey Level Co-occurrence Matrix.* Halli taseme kaasesinemise maatriks – maatriks, mis kirjeldab pildil esinevate pikslite jaotust.
HOG	*Histogram of Oriented Gradients.* Orienteeritud gradientide histogramm – objekti tuvastamisel kasutatav omaduste kirjeldaja.
HSV	Värviruum, kus erinevaid värvitoone esitatakse kolme suuruse abil – toon (*hue*), küllastus (*saturation*), heledus (*value*)
Hu momendid	*Hu Moments*. Kuju iseloomustavad suurused, mille väärtused ei sõltu pildi suurusest, rotatsioonist ega moonutamisest.
LAB	Värvimudel, mis püüab kõrvaldada sõltuvust seadmetest ning selle alusekson moodus, kuidas inimsilm värvi tajub. Värv esitatakse kolme komponendi abil: L – heledus, A – komponent rohelisest punaseni, B – komponent sinisest kollaseni
LBP	*Local Binary Patterns.* Lokaalsed binaarmustrid – raalnägemises kasutatav tekstuuri kirjeldaja.
Pilditöötlus	*Image processing*. Pildi analüüsimine meetoditega, mis suudavad identifitseerida varje, värve ja suhteid, mida inimsilm ei erista.
Raalnägemine	*Computer Vision*. Viis, kuidas arvutid koguvad ja tõlgendavad ümbritsevast füüsilisest maailmast pärinevat visuaalset informatsiooni.
RGB	Liitvärvimudel, milles erinevaid värvitoone saadakse kolme põhivärvuse – punane, roheline ja sinine – liitumisel.
Värvimomendid	*Colour moments*. Pildi värve kirjeldavad suurused.
YcbCr	Värviruum, mis kirjeldab värve kolme komponendiga: heledus, sinine heledus (*blue-luminance*) ja punane heledus (*red-luminance*), kasutatakse värviinfo digitaalseks kodeerimiseks.
YUV	Värviruum, mis on väga sarnane YcbCr värvimudelile; kasutati näiteks telepildi analoog-kodeerimiseks.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Igal aastal ootavad tuhanded loomad varjupaikades uut kodu. Nendest umbes pooled seda siiski ei leia [1] . Loomade info kuvamiseks on paljudel varjupaikadel oma kodu- ja Facebooki lehed. Lisaks nendele on 2016. aasta kevadest avatud ka portaal Loom24, mis ühe osana hõlmab varjupaikades viibivate loomade andmeid. Varjupaikadel on portaali aga ebamugav kasutada ning ühtlasi ei ole see eriti kaasa aidanud loomadele kodu leidmisel (vt Lisa 1).

Suur osa varjupaigas viibivatest loomadest on kellegi pereliikmed. Kuigi kiibistamine aitab paljudel neist koju tagasi jõuda, on siiani ligi 86% loomadest kiibistamata [2] .

Nende probleemide lahendamiseks pakub autor välja sisupõhist pildiotsingut kasutava veebirakenduse, mis koondab kokku erinevates varjupaikades olevate loomade info. Ühtne varjupaikade portaal, mis hõlmaks ka sotsiaalset poolt, võiks kaasa aidata looma varjupaigast võtmise populariseerimisele. Kiibistamata loomade leidmisel tuleks abiks sisupõhine pildiotsing ehk kasutaja saab kadunud looma pildi üleslaadimisel vastuseks hulga varjupaikadesse saabunud sarnastest loomadest.

Sisupõhine pildiotsing (*Content Based Image Retrieval*[1]) on vägagi uuritud teema. Paljud teadustööd on üritanud töötada välja võimalikult efektiivset viisi piltide otsimiseks multimeedia andmebaasidest. Kuigi arendatud on väga palju erinevaid metoodikaid, ei ole siiski veel olemas ühte universaalset lahendust. Antud töö eesmärgiks ei ole töötada välja uut viisi piltide otsimiseks, vaid kasutada juba valmisolevaid lahendusi, mis võiksid lahendada sarnaste loomade leidmise probleemi.

Antud töö raames seab autor endale kaks peamist eesmärki:

- luua lahendus, mis võimaldab otsida pildil olevale loomale sarnaseid;

[1] Siin ja edaspidi tõlked [15]

- luua veebirakendus, mis kasutab eelpool mainitud lahendust sarnase looma leidmiseks ning koondab eri varjupaikades olevate loomade info.

Loodud lahenduse ja veebirakenduse kood on üleval Githubi repositooriumis[1].

[1] https://github.com/pluhila/animal-shelter-cbir-app

2 Analüüs

Selles peatükis kirjeldatakse loodava veebirakenduse funktsionaalsed ja mittefunktsionaalsed nõuded, tuuakse välja sarnased lahendused ning selgitatakse töö planeerimist.

2.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded jagunevad antud töö puhul kaheks – neid vaadeldakse nii tavakasutaja kui varjupaiga seisukohast.

Kõik kasutajad saavad end registreerida ning ühtlasi kasutajanime ja parooli abil lehele siseneda. Sellest tulenevalt on kasutajatel võimalus ka välja logida.

Tavakasutaja saab vaadelda aktiivseid loomi ning lisada neid endale meeldivaks. Igal loomal on profiil, kus saab näha kõiki tema kohta käivaid andmeid ning pilte. Aktiivseid loomi on võimalik otsida erinevate parameetrite alusel – liik, sugu, vanus, teostatud protseduurid, asukoht, varjupaika sattumise aeg. Lisaks sellele saab kasutaja otsida ka sarnaseid loomi. Selle jaoks on tal võimalik ette anda pilt, mille peal olevale loomale sarnaseid loomi otsitakse.

Varjupaiga kasutajal on üks või mitu varjupaika. Varjupaik saab lisaks kõigile tavakasutajale võimaldatud tegevustele teostada järgnevaid toiminguid:

- looma lisamine sisestades looma nime, liigi, soo, vanuse, kirjelduse, pildid, tehtud protseduurid ning varjupaiga;
- lisatud piltide seast loomale profiilipildi valimine;
- varjupaigas oleva looma andmete muutmine;

- varjupaigas oleva looma seisundi muutmine: aktiivne, mitteaktiivne, adopteeritud.

2.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsete nõuete alla käivad eestikeelne kasutajaliides, inglisekeelne lähtekood ning nõue, et rakendus töötab brauserite Google Chrome ja Mozilla Firefox uusimate versioonidega.

2.3 Sarnased lahendused

Nii välismaal kui Eestis on rakendusi, mis muudavad looma varjupaigast valimise lihtsamaks. Sellist lahendust, mis võimaldaks ka pildi alusel loomi otsida, Eestis ei ole. Võrdluse eesmärgil on sellesse peatükki kokku koondatud mõned paremad näited välismaalt ning ühtlasi Eesti esimene loomaportaal Loom24.

Finding Rover[1] kasutab näotuvastust kadunud koera leidmiseks. Keskkond koondab enda alla mitmeid USA loomaarste, varjupaiku kui ka tavalisi inimesi. Kasutamiseks tuleb kõigepealt üles laadida pilt oma koera näost otsevaates, märkida ära silmade ja nina asukoht ning seejärel eraldatakse pildilt iseloomulikud omadused. Kui keegi leiab tänavalt looma, siis läbitakse samad tegevused ning kadunud loomade omadusi võrreldakse leitud loomadega. Kui need langevad piisavalt kokku, teavitatakse kadunud looma omanikku. Finding Rover on saadaval nii Androidi, iOS kui veebirakendusena.

PiP[2] rakendus töötab sarnasel põhimõttel nagu Finding Rover, kuid on mõeldud kõigile loomadele, mitte ainult koertele. Kasutajal tuleb ise määrata looma silmade ja suu asukoht. Looma kadumise korral saadetakse teade varjupaikadesse ning jagatakse infot sotsiaalmeedias. Samas on rakenduse hinnang Google Play poes kõigest 2.5 ning kasutajad kurdavad, et rakendus ei hakka isegi tööle piltide üles laadimisest rääkimata. PiP on saadaval Androidi ja iOS rakendustena.

1 http://www.findingrover.com/

2 http://www.petrecognition.com/

AllPaws[1] on USA-põhine rakendus, mis koondab kokku üle 140 000 looma üle terve riigi. Platvorm annab võimaluse otsida loomi mitmete parameetrite alusel, vaadata nende profiile, lisada meeldivaks ning jagada Facebookis. Ühtlasi saab loomade piltidest genereerida meeme[2]. Lisaks veebilehele on AllPaws-il ka iOS rakendus.

Zeppee[3] on Austraalias saadaval iOS või Androidi rakendusena. Tegemist on Tinderilaadse rakendusega, mis võimaldab vaadata erinevaid loomi, lisada neid meeldivaks ning juhul, kui on soov loom endale võtta, saab läbi rakenduse ühendust võtta varjupaiga või omanikuga. Läbi äpi saab ka enda loomadele uut kodu leida.

Loom24[4] on ennast nimetanud Eesti esimeseks terviklikuks loomaportaaliks. Ühe osana on seal esindatud ka mitmed erinevad varjupaigad üle Eesti. Loomi saab otsida liigi või märksõna alusel. Lisaks sellele on Loom24 portaalis ka loomadega seotud uudised, kirbukas, üritused ning isegi põllumajandusloomad.

Varjupaikadega suheldes (vt Lisa 1) selgus, et antud portaali pole ei neil ega loomaotsijatel kõige mugavam kasutada ning loomadele kodu leimisel ei ole Loom24 olulist osa mänginud.

2.4 Töö planeerimine

Antud töö käigus autori poolt loodud rakendus vajas ühe osana sisupõhist pildiotsingut. Seetõttu oligi esimeseks eesmärgiks luua töötav liides, mis teisendab etteantava pildi omadusvektoriks ning suudab leida sellele sarnaseid pilte. Töö teine eesmärk oli luua veebirakendus, mis kasutab seda liidest. Rakenduse loomise etapid jagunesid vastavalt:

- looma lisamine;
- loomade listivaade;
- looma profiilivaade;
- loomade otsing pildi alusel;

1 https://www.allpaws.com/
2 https://et.wikipedia.org/wiki/Meem
3 http://www.zeppee.com.au/
4 http://loom24.ee/beta/

- loomade otsing asukoha, liigi, soo ja vanuse alusel;
- looma muutmine;
- kasutaja autentimine;
- loomade meeldivaks lisamine.

3 Sisupõhine pildiotsing

Tänapäeval on piltide kasutamine väga laialt levinud. Enamus pilte ei sisalda mingisugust annotatsiooni või märksõnu, et võimaldada tüüpilist tekstipõhist otsingut. Sisupõhine pildiotsing (CBIR) ei vaja mingisuguseid metaandmeid, vaid eraldab pildilt ise vajalikud omadused ning kasutab etteantud sarnasuse mõõtu piltide kollektsioonist otsimiseks [3] . Tüüpiliselt eraldaiakse pildilt värvi, tekstuuri ja kuju omadusi, mille kirjeldamiseks võib kasutada mitmeid erinevaid viise, millest siin peatükis tuuakse välja mõned.

3.1 Värvi omadused

Värvi alusel piltide võrdlemine on üks levinumaid lahendusi. Need omadused ei sõltu pildi suurusest ega rotatsioonist [4] . Üks lihtsamaid viise värvi kirjeldamiseks, on värvi histogramm.

Värvi histogramm näitab eri värvide sagedust mingis värviruumis. Histogrammi eeliseks on värvide globaalse jaotumise kirjeldamine piltidel. Puuduseks on aga värvide jaotuse paiknemise info puudumine. See tähendab, et värvi histogramm ei suuda kirjeldada konkreetseid objekte pildil. [4]

Kahel erineva sisuga pildil võivad olla väga sarnased histogrammid, näiteks kass punase seina taustal ja peotäis punaseid õunu (vt Joonis 1[1]).

Histogrammi arvutamiseks tuleb kõigepealt valida värviruum. Iga värviruum esitleb värvide kogumit, mis tuleneb valitud värvimudeli põhikomponentide kombinatsioonidest. Värviruum määratleb – enamasti 3-mõõtmelise – koordinaatide süsteemi oma komponentide baasil ning selles ruumis iga punkt vastab ühele spetsiifilisele värvile. [6]

[1] Pildid pärit https://satish-sharma.blogspot.com.ee/ ja http://www.prevention.com/food/apple-varieties-and-recipes (Jason Koski/Cornell University Photography)

Joonis 1: Kaks täiesti erineva sisuga pilti, millel on sarnased värvihistogrammid.

Värviruumi valimisel tuleb arvestada sellega, kuidas inimene ise värve tajub. Üks levinumaid värviruume, RGB ei vasta nendele nõuetele. RGB-värvimudeli abil võime määratleda spetsiifilise värvi ainult relatiivse suhtega teistesse värvidesse [6] . Selle asemel on mõistlik kasutada näiteks hoopis HSV-värvimudelit. [4]

Histogrammi arvutamiseks, tuleb värviruum jaotada väiksemateks osadeks. Histogrammi arvutamiseks tuleb ära lugeda, mitu pikslit igasse intervalli satub. [4]

Üks võimalus histogrammi arvutamiseks HSV ruumi jaoks:

1. piltide konverteerimine RGB ruumist HSV ruumi;
2. HSV ruum jaotatakse intervallideks;
3. iga piksli puhul vaadatakse, millisesse intervalli see langeb;
4. loetakse kokku, mitu pikslit igasse intervalli sattus;
5. sarnasuse arvutamiseks leitakse histogrammide vaheline kaugus (vt 3.5).

Selleks, et anda värviomadustega kaasa ka teatav asukoha info, võib pildi jagada nn plokkideks ning arvutada igaühe kohta histogramm [4] . Selleks, et rõhutada mingi osa tähtsust, võib mõne ploki histogrammi ka mingisuguse koefitsendiga läbi korrutada.

Värvi kirjeldamiseks võib kasutada ka värvimomente. Nende leidmiseks tuleb RGB pilt jagada punasteks, rohelisteks ja sinisteks komponentideks. Seejärel arvutatakse iga

komponendi keskmine väärtus ning standardhälve. Värvimomendid genereerivad kuuemõõtmelise vektori. [5]

3.2 Tekstuuri omadused

Tekstuuri omadused kirjeldavad pikslite piirkondi. Nagu värvigi puhul, on tekstuuri kirjeldamiseks mitmeid erinevaid viise.

Lokaalsed binaarmustrid (*Local Binary Patterns*, LBP) on lihtne, kuid samas efektiivne viis tekstuuri analüüsimiseks. See tehnika kirjeldab iga piksli ümbrust ning moodustab nendest kirjeldustest histogrammid. [3]

Olgu B ühe halltoonides pildi piksli kirjeldus, kus keskmine element väljendab selle piksli väärtust ning ülejäänud elemedid teda ümbritsevate pikslite väärtuseid.

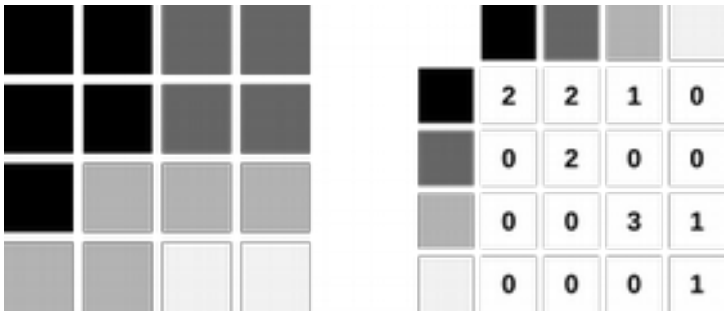
$$B=\begin{pmatrix} g_{(-1,-1)} & g_{(-1,0)} & g_{(-1,1)} \\ g_{(0,-1)} & g_{(0,0)} & g_{(0,1)} \\ g_{(1,-1)} & g_{(1,0)} & g_{(1,1)} \end{pmatrix} \quad (1)$$

Esimese sammuna lahutatakse igast elemendist keskmise piksli väärtus.

$$LBP_1=\begin{pmatrix} g_{(-1,-1)}-g_{(0,0)} & g_{(-1,0)}-g_{(0,0)} & g_{(-1,1)}-g_{(0,0)} \\ g_{(0,-1)}-g_{(0,0)} & 0 & g_{(0,1)}-g_{(0,0)} \\ g_{(1,-1)}-g_{(0,0)} & g_{(1,0)}-g_{(0,0)} & g_{(1,1)}-g_{(0,0)} \end{pmatrix} \quad (2)$$

Siis määratakse igale ümbritsevale pikslile uus väärtus. Kui väärtus on nullist väiksem, on uueks väärtuseks 0, vastasel juhul 1. Nii ehitatakse 256 intervalliga LBP histogramm, mis kirjeldab pilti. [3]

Halli taseme kaasesinemise maatriks (*Grey Level Co-occurrence Matrix*, GLCM) esitab tabeli kujul info selle kohta, kui tihti piksli ereduse väärtuste erinevad kombinatsioonid pildil esinevad (vt Joonis 2). [8]



Joonis 2: Halli taseme kaasesinemise maatriks. Vasakul pildi pikslid ning paremal vastav halli taseme kaasesinemise maatriks. (Autori joonis)

3.3 Kuju omadused

Rääkides kujust, mõeldakse pildil olevate objektide kuju, mitte pildi dimensioone. Kuju kirjeldamiseks läheb enamasti vaja segmenteerimist või äärtetuvastust, mis võimaldab keskenduda vaid nendele kontuuridele, mida päriselt kirjeldada tahetakse [9] .

Servahistogrammi (*Edge histogram*) ideeks on tuvastada pildil asuvad objekti piirjooned. Selleks võib kasutada näiteks Canny servatuvastajat[1], mis kõigepealt eemaldab pildilt müra, siis ühtlustab pikslite väärtused ning kasutab gradiente, et tuvastada servade asukohad [10] .

Orienteeritud gradientide histogramme (*Histogram of Oriented Gradients*, HOG) kasutatakse tihti masinõppes objektide tuvastamiseks. Samas saab HOG abil kirjeldada ka objektide kuju ja tekstuuri. Selleks tuleb jaotada pilt väiksemateks plokkideks, arvutada igaühe jaoks horisontaalsed ja vertikaalsed gradiendid ning leida nende magnituudid ja suunad [11] . HOG vektorid võivad kasvada aga väga suureks, mis tähendab, et nende kasutamine võib osutuda kulukaks – pikad omadusvektorid võtavad rohkem ruumi ning nende omavaheline võrdlemine rohkem aega.

Kuju kirjeldamiseks võib kasutada ka Hu momente (vt täpsemalt 4.4.3). Tavaliselt eraldatakse Hu momendid pildil asetseva objekti siluetist või piirjoontest. Selleks võib

[1] http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

kasutada segmenteerimist (nt. muudetakse tagaplaanil olevad pikslid mustaks ning esiplaani omad valgeks) või lävendi määramist.

Kõige lihtsamal juhul tähendab lävendi määramine seda, et võetakse konkreetne piksli väärtus, millest suurema väärtusega pikslid muudetakse mustaks ja väiksemaga valgeks. Keerulisematel juhtudel võetakse arvesse ka seda, et pilt ei pruugi olla ühtlaselt valgustatud ning erinevatele pildi osadele määratakse erinevad lävendid. [13]

3.4 Ühendatud omadusvektorid

Selleks, et saavutada sisupõhises pildiotsingus paremaid tulemusi, kasutatakse tihti mitmeid omadusi koos.

Värvi ja tekstuuri omaduste koondamiseks kasutatakse näiteks värvi sidususe vektoreid (c*olour coherence vectors*) ja värvi korrelogramme (c*olour correlograms*),

Värvi sidususe vektorid koosnevad kahest osast: üks histogramm sidusatest pikslitest ja teine mittesidusatest. Piksleid peetakse sidusateks, kui nad on osa pidevast sarnase värvitooniga alast ning selle ala suurus ületab etteantud lävendi, milleks on tavaliselt 1% kogu pildi suurusest. [3]

Värvi korrelogramm väljendab, kuidas värvipaaride ruumiline seos muutub nendevahelise kaugusega. Korrelogramm on nagu värvipaaridega indekseeritud tabel, kus *d*-s lahter (reas *i*, veerus *j*) tähistab tõenäosust, et *j* värvi piksel on *i* värvi pikslist kaugusel *d*. [7]

Teine variant ühendatud omadusvektoriteks on lihtsalt panna omavahel kokku kaks erinevat vektorit. Näiteks servahistogramm ja LBP või värvi korrelogramm ja HOG.

3.5 Sarnasuse arvutamine

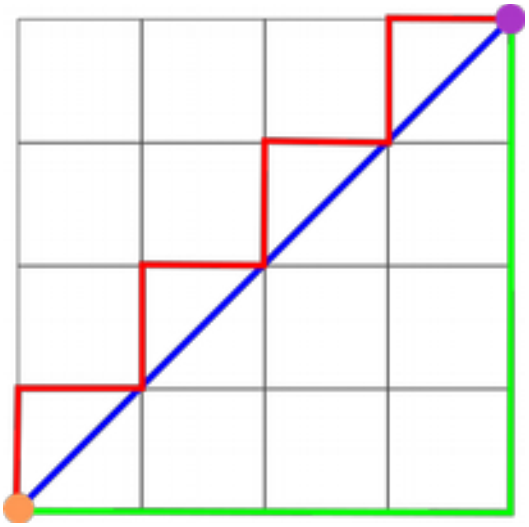
Selleks, et leida, millised pildid on omavahel sarnased, tuleb leida nende omadusvektorite omavaheline kaugus. Mida väiksem on kaugus, seda sarnasemad on pildid.

Eukleidiline kaugus (*Euclidean distance*) on tänu oma tõhususele üks kasutatavamaid sarnasuse mõõte pildiotsingus [5] . Kahe vektori omavahelise kauguse leidmiseks võetakse ruutjuur elementide vahe absoluutväärtuse ruutude summast (Valem 1). Eukleidilist kaugust kirjeldatakse ka kui kaugust linnulennul, mis tähendab, et leitakse lühim tee kahe punkti vahel [12] .

$$D_{Euclidean} = \sqrt{\sum_{i=1}^{n} (|u_i - v_i|)^2} \quad (1)$$

Valem 1: Eukleidiline kaugus [12]

Manhattani kaugust (Valem 2) nimetatakse ka „kvartali" kauguseks. Erinevalt Eukleidilisest kaugusest ei leita lühimat teed kahe punkti vahel. Selle asemel võib ette kujutada ruudustikku ning kauguse määramisel liigutakse ainult mööda ruutude piirjooni (vt Joonis 3). [12]



Joonis 3: Eukleidliline kaugus (sinine joon) ning Manhattani kaugus (punane ja roheline joon) (Autori joonis)

$$D_{Manhattan} = \sum_{i=1}^{n} |u_i - v_i| \quad (2)$$

Valem 2: Manhattani kaugus [12]

Chebyshevi kaugus (Valem 3) leiab aga maksimaalse kauguse mistahes kahe vektori komponendi vahel. [12]

$$D_{Chebyshev} = \max_i |u_i - v_i| \quad (3)$$

Valem 3: Chebyshevi kaugus [12]

Hammingu kaugus võrdleb vektorite mittekokkulangevate komponentide suhet vektorite pikkustesse. Kui kahe vektori pikkuseks on 4 ning nende seas on kaks komponenti, mis ei lange kokku, siis on Hammingu kauguseks *2 / 4 = 0.5*. [12]

Chi-ruudus kaugus on sarnane Eukleidilise kaugusega, kuid selle puhul arvutatakse komponentide jaoks kaalud, mis on pöördvõrdelised nende esinemise sagedusega (Valem 4). Kui andmete suhe koguarvuga on varasemalt leitud, saab arvutamist veelgi lihtsustada. [14]

$$D_{Chi-Squared} = \sum_{i=1}^{n} \frac{(u_i^2 - v_i^2)^2}{(u_i + v_i)} \quad (4)$$

Valem 4: Chi-ruudus kaugus [14]

4 Sisupõhise pildiotsingu realisatsioon

Antud jaotises käsitletakse sisupõhise pildiotsingu realisatsiooni Pythonis.

4.1 Tehnoloogia valik

Antud ülesande lahendamiseks valis autor Pythoni[1] programmeerimiskeele peamiselt isikliku eelistuse tõttu. Python võimaldab kasutada ka pilditöötluseks vajalikke teeke nagu OpenCV[2] ning Scikit-Image[3]. OpenCV teek sisaldab sadu raalnägemise (*Computer Vision*) algoritme, mida sobiva lahenduse leidmisel kasutada. Scikit-Image koondab enda alla kollektsiooni pilditöötluseks vajalikke algoritme, mis on kõigile tasuta kasutamiseks.

4.2 Sarnaste loomade leidmise probleemide analüüs

Loomade sarnasuse hindamiseks saab võrrelda nende värve, mustreid, karvapikkust, suurust, kõrvakuju jne. Käesoleva töö puhul kasutatakse neist mõnda, mida vaadeldakse täpsemalt peatükis 4.3.

Üks lihtsamaid võimalusi on võrrelda loomi värvi alusel. Kuna piltidel on lisaks loomale ka taust, mis võib olla väga erinev, siis ei saa pilte aga ainult värvi histogrammide alusel võrrelda. Järgnevalt vaadeldaksegi, kuidas seda probleemi lahendada.

1 https://www.python.org/
2 http://opencv.org/
3 http://scikit-image.org/

4.2.1 Looma asukoha tuvastamine pildil

Üheks lahenduseks eelnevalt nimetatud probleemile on tuvastada looma asukoht pildil. Kui tuvastada looma nägu ning leida vaid sellest osast omadusvektor, muutuks tausta osakaal nii väikseks, et ei omaks sarnasuse arvutamisel tähtsust.

OpenCV repositooriumis[1] on mitmed juba treenitud Haar klassifitseerijad, mille alusel saab tuvastada nägusid, silmi, naeratusi ning ühtlasi ka kasse.

Haar klassifitseerija tööpõhimõte seisneb paljude klassifitseerijate koondamises. Üksinda ei suudaks need klassifitseerijad vajalikku objekti pildil tuvastada, aga koos osutuvad nad väga võimsaks. Tegemist on masinõppel põhineva lähenemisega, kus tuvastaja treenitakse paljude positiivsete (otsitav objekt on pildil) ning negatiivsete (otsitavat objekti ei ole) näidetega. Piltidelt leitakse objekti iseloomustavad omadused, mille tuvastajate koondamisel valmib Haar klassifitseerija. [16]

Kasside nägude tuvastamiseks on OpenCV repositooriumis koguni kaks klassifitseerijat. Esimene neist eeldab, et kass on täiesti otsevaates ning olukordades, kus loom näiteks lamab, nägu ei tuvastata. Teine klassifitseerija on eelmise edasiarendus, kuid sellegipoolest jäävad ka sellel lamavad kassid tuvastamata.

Katsetamise eesmärgil otsustas autor ise proovida klassifitseerija treenimist. Aluseks võeti “The Oxford-IIIT Pet Dataset” andmekogu[2], kus on iga 37 kassi- ja koeraliigi kohta umbes 200 pilti. Piltidel varieeruvad valgus, poosid ja suurused. Iga pildi kohta on ka välja toodud pea asukoht ning ühtlasi segmenteeritud pilt, mis eristab tausta ja esiplaani.

Tuvastaja treenimiseks koostati positiivne andmehulk kasside nägudest, negatiivne andmehulk taustadest[3] ning failid, mis kirjeldavad neid andmehulki. OpenCV on sisse ehitanud meetodi Haar tuvastaja treenimiseks[4] (Joonis 4).

1 https://github.com/opencv/opencv/tree/master/data/haarcascades
2 http://www.robots.ox.ac.uk/~vgg/data/pets/
3 https://github.com/sonots/tutorial-haartraining
4 http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html

```
opencv_createsamples -info cats.info -vec cats.vec -w 24 -h 24 -num
1188

opencv_traincascade -data data -vec cats.vec -bg bg.txt -w 24 -h 24
-numPos 1000 -numNeg 3019 -precalcValBufSize 1024 -precalcIdxBufSize
1024 -featureType HAAR -numStages 12
```

Joonis 4: Klassifitseerija treenimiseks vajalikud käsud

Esimene käsk loob kõigist cats.info failis kirjeldatud positiivsetest näidetest ühe faili *cats.vec*, kus kõik 1188 pilti on suuruses 24x24.

Teine käsk treenib tuvastaja. Lisaks positiivsete näidete failile *cats.vec* antakse ette ka negatiivseid pilte kirjeldav *bg.txt*, treenimiseks kasutavate positiivsete ja negatiivsete piltide hulk, suurus, treenitava klassifitseerija tüüp ning etappide arv.

OpenCV poolt pakutavate ja isetreenitud klassifitseerijate võrdlemiseks kasutati 102 pilti varjupaigas olevatest kassidest. Võrdluse aluseks olid:

- tuvastatud kasside arv;
- õigesti tuvastatud kasside hulk;
- piltide hulk, millel ei tuvastatud ühtegi kassi.

Kõiki testimisel saadud tulemusi saab näha tabelis 1. Kuna ükski kolmest katsest ei näidanud häid tulemusi, prooviti ka kontuuride abil leidmist, segmenteerimist ning võtmepunkte, kuid ka nende puhul jätsid tulemused soovida (vt Lisa 4).

Seega otsustas autor pöörduda tagasi treenitud tuvastajate juurde. Õnnestus leida üks LBP klassifitseerija, mis on treenitud koguni 9500 positiivse ja ligi miljardi negatiivse pildiga[1]. LBP klassifitseerija puhul eraldatakse pildilt tekstuuri kirjeldavad lokaalsed binaarmustrid, mille alusel määratakse, millised omadused kirjeldavad otsitavat objekti ja millised mitte.

Leitud klassifitseerijaga saadud tulemused olid eelmistest tunduvalt paremad (Tabel 1) ning seeetõttu otsustas autor jätkata antud lahendusega.

[1] http://www.vision-ary.net/2015/03/largest-boosted-cascades-opencv-lbp-haar-hog/

Tabel 1: Kassituvastajate võrdlus

Klassifitseerija	Tuvastas	Tuvastas õigesti	Ei tuvastanud
OpenCV esimene kassituvastaja	68	52	34
OpenCV teine kassituvastaja	62	53	40
Isetreenitud klassifitseerija	62	13	40
LBP klassifitseerija	94	80	8
Tagavaravariandiga tuvastaja	102	81	0

4.2.2 Lahenduse optimiseerimine

Kuigi väljavalitud lahendus tuvastas kassi õigesti kaheksakümnel juhul 102-st, tundus siiski, et ehk on võimalik rohkem saavutada.

Osadel juhtudel tuvastati lisaks kassi näole ka mõni muu ala, nagu käpp, keha, saba või üldsegi tekk kassi all. Selle parandamiseks otsustas autor kasutada OpenCV funktsiooni *detectMultiScale3*, mis lisaks tuvastatud nägudele annab ka väärtuse *levelWeights*, mis näitab usaldusväärsuse astet, et tegemist on tõepoolest kassi näoga (Joonis 5). Kasutades neid väärtuseid, valiti igal pildil välja üks ristkülik, mis sisaldab kõige tõenäolisemalt kassi nägu.

```
rects,rejectLevels,levelWeights = detector.detectMultiScale3
(gray, scaleFactor=1.05, minNeighbors=5, minSize=(30,30),
outputRejectLevels=True)
```

Joonis 5: Kasside nägude tuvastamine OpenCV detectMultiScale3 funktsiooni abil

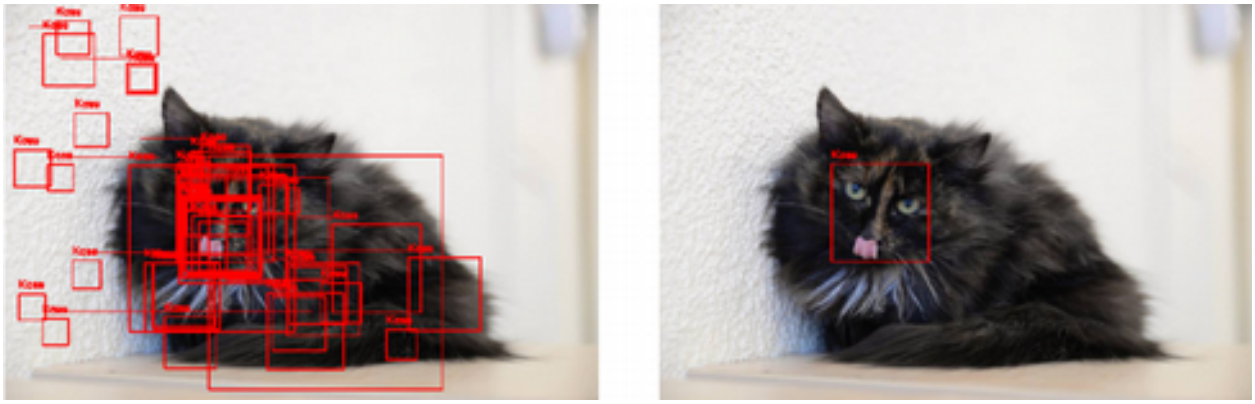
DetectMultiScale3 funktsioon läbib nägude tuvastamiseks mitu etappi. Iga etapi puhul muudetakse pilt veidi väiksemaks, et tuvastada eri suurustega näod.

- *rects* – tuvastatud näod;
- *rejectLevels* – mitmendal etapil ristkülik välistati, varasemalt tagastas kõikide viimasel neljal etapil välistatud ristküliku jaoks väärtused, nüüd vaid viimasel
- *levelWeights* – suurus, kui tõenäoliselt on tegemist näoga
- *gray* – halltoonides pilt, millelt nägu otsitakse
- *scaleFactor* – suurus, mitu korda igal etapil pildi suurust vähendatakse;

- *minNeighbors* – mitu naabrit igal ristkülikul peab olema, et seda näoks võiks pidada, aitab välistada sama näo mitmekordset tuvastamist;
- *minSize* – minimaalne ristküliku suurus, mis võib sisaldada nägu; väiksemaid ristkülikuid ignoreeritakse;
- *outputRejectLevels* – muutuja, mis sätestab, kas *rejectLevels* tagastatakse

Siiski jäid problemaatiliseks olukorrad, kus ühtegi kassi ei tuvastatud. Analüüsides pilte, millel kassi ei leitud või leiti valesti, selgusid korduvad omadused:

- kass ei vaata otse - on külgvaates, pea viltu, alaspidi;
- kassi pea ei ole täielikult näha.



Joonis 6: Kassi näo tuvastamine. Vasakul kõik leitud „näod”, paremal kõige tõenäolisem.

Kuna mittetäieliku kassi tuvastamiseks oleks vaja eraldi klassifitseerijat, mis on treenitud leidma ka osalisi nägusid, otsustati see põhjus arvestamata jätta. Kuid teiste juhtude puhul aitab kaasa teadmine, et kass on alati pildil ning teadmata on vaid tema asukoht. Seetõttu võiks juhul, kui kassi pildilt ei leita, sooritada uue otsingu madalamate kriteeriumitega. Uue otsingu tulemusel leitakse väga palju võimalikke nägusid (Joonis 6). Valides välja neist kõige tõenäolisema, ei jäänud enam ühelgi pildil kass tuvastamata (Tabel 1). Nendest piltidest, mille puhul nägu täielikult ei leitud, tuvastati kolmeteistkümnel osa näost, viiel mingi muu osa kassist ning ainult kolmel juhul pandi “kass” täiesti valesse kohta.

Kui kasside leidmiseks leidus juba olemasolev treenitud klassifitseerija, siis koerte jaoks autoril sellist lahendust leida ei õnnestunud. Kuna lisaks koertele on võimalik, et varjupaika satuvad ka jänesed, rotid, papagoid jne, siis oleks teoorias vaja iga võimaliku kodulooma liigi jaoks eraldi klassifitseerijat. Seetõttu eeldatakse antud töös, et koerte ja muude loomade puhul moodustab looma nägu profiilipildil piisavalt suure osa, et tausta osatähtsus ei mängi rolli.

4.3 Omadusvektori valimine sarnaste piltide leidmiseks

Kui kasside nägude leidmiseks vajalik lahendus leitud, asuti sarnaste piltide leidmiseks valima sobilikku omadusvektorit. Omadusvektorite omavahelisel võrdlemisel on võimalik tuvastada, kui sarnaste piltidega tegemist on. Omaduste eraldamine on sisupõhise pildiotsingu kõige olulisem samm, kuna see määrab kogu süsteemi efektiivsuse. Enne pildilt omaduste eraldamist, ühtlustatakse pildi histogramm – see tähendab, et pimedate piltide puhul tehakse need heledamaks ning ülevalgustatud piltide puhul hoopis pimedamaks. See aitab ühtlasi tuua pildil olevaid värve rohkem esile.

Joonis 7: Vasakul pilt enne, paremal pärast histogrammi ühtlustamist

Võimalike omadusvektorite efektiivsuse võrdlemise eesmärgil koostati kaks erinevat pildikogumit:

- Nelikümmend kassipilti, kus on hallid, punased, mustad ja valged kassid - iga värvi kohta 10 pilti. Kõikide värvide kohta on valitud päringupildid, millele otsitakse sarnaseid (Joonis 8). Lisaks sellele on pildid konkreetsetest kassidest, kelle puhul oodatakse vastuseks pilte samast kassist.

- Nelikümmend koerapilti, kus on hundikoerataolised, taksitaolised, suured lühikarvalised, väiksed pikakarvalised koerad jne. Otsitakse konkreetsele koerale vastavaid pilte.

Joonis 8: Omadusvektori valimisel kasutatud kasside päringupildid

4.3.1 Värvi omadused

Värvi omaduste võrdlemisel saab mitmeid meetodeid rakendada erinevates värviruumides nagu RGB, HSV, LAB, YUV jne. Selleks, et anda kaasa ka ruumiline info värvide jagunemise kohta, jagatakse pilt kuueteistkümneks ühesuuruseks plokiks. Iga ploki kohta leitakse värvi omadused, mis koondatakse lõplikku omadusvekorisse.

Peatükis 3.1 kirjeldatud värvi omaduste kirjeldamise meetoditest võrreldi värvimomente ning mitme eri värviruumi histogramme. Kasside puhul saadi igale päringupildile vastuseks kümme pilti ning lugedes kokku, mitu neist kuulusid õigesse värvikategooriasse, arvutati keskmine õigsuse protsent.

Tabel 2: Värvi omaduste kirjeldamise meetodite võrdlus

Meetod	Hall	Punane	Must	Valge	Keskmine protsent
RGB histogramm	7	9	10	7	82.5%
HSV histogramm	6	9	5	7	67.5%
Värvimomendid	8	8	9	9	85%
LAB histogramm	6	8	9	6	72.5%
YUV histogramm	6	9	9	6	75%
YcbCr histogramm	5	8	10	6	72.5%

Kuigi HSV värviruum peaks võrreldes RGB-ga olema sarnasem sellele, kuidas inimene värve tajub, olid HSV histogrammi tulemused kõige nõrgemad (Tabel 2). Nende tulemuste puhul sõltub palju valitud piltidest ja nende kvaliteedist. HSV histogrammi

tulemust võib mõjutada ka kauguse arvutamiseks kasutatava valemi valik. Siiski esineb mitmetes uurimustes anomaaliaid, kus HSV ei töötagi nii hästi ning ühest vastust ei teata. [17]

Värvimomendid ja RGB histogramm näitasid seevastu väga kõrget taset. Näiteks, kui valgete kasside päringuga tuvastati 9 valget kassi, siis see üks, kes ei olnud valge, oli helehall ning seega päris sarnane hoolimata teise kategooriasse kuulumisest. Katsete tulemusi saab näha Lisas 5.

Loodava rakenduse üheks ideeks on lihtsustada kiibistamata loomade leidmist. Seetõttu oli oluline, et kui anda päringupildina sisse pilt konkreetsest kassist või koerast, on esimese kümne vaste seas ka otsitava looma pilt. Selle testimiseks lisati juba olemasolevatele kassipiltidele veel 8 pilti kahest erinevast kassist, mis on tehtud eri keskkondades. Eesmärgiks oli saada päringu vastuseks tuleva 10 pildi hulka võimalikult palju pilte samast loomast.

Kasside piltide puhul piisas sama looma leidmiseks vaid värvi omaduste võrdlemisest. Kasutades eelmises testis parimaid tulemusi andnud värvimomente, olid kõik neli pilti otsitavast musta-valgekirjust kassist ka 10 sarnaseima pildi hulgas (Joonis 9).

Joonis 9: Sama kassi otsingu tulemused

4.3.2 Tekstuuri ja kuju omadused

Kui kasside puhul piisas ainuüksi värviomaduste võrdlemisest, siis koertel on rohkem erinevusi. Väikest taksitüüpi koera ning suurt hundikoera ei saa pidada väga sarnaseks isegi, kui nad on sama värvi. Siinkohal saab arvestada kasutajapoolset infot looma suuruse või karvkatte pikkuse kohta. Mingil määral parandab tulemust aga ka tekstuuri ja kuju omaduste kaasamine võrdlusesse.

Testpiltideks valiti suurt kasvu lühikarvaline hundikoer ja kirju pikakarvalise kasukaga väikest kasvu segavereline koer. Ka siinkohal toimisid värvimomendid üllatavalt hästi. Lisades omadusvektorile juurde aga tekstuuri kirjeldamiseks halli taseme kaasesinemise maatriksi (vt 3.2) ning kuju jaoks Hu momendid (vt 3.3), muutus tulemus veidi täpsemaks (Joonis 10).

Piltide omavahelisel võrdlemisel mõjutab tulemust pildi kvaliteet, valgus ning nurk, mille all foto tehtud on. Kõige paremaid resultaate saab hästi valgustatud piltide puhul, millel loom on otsevaates. Muudel juhtudel hakkab sarnasust mõjutama näiteks looma poos.

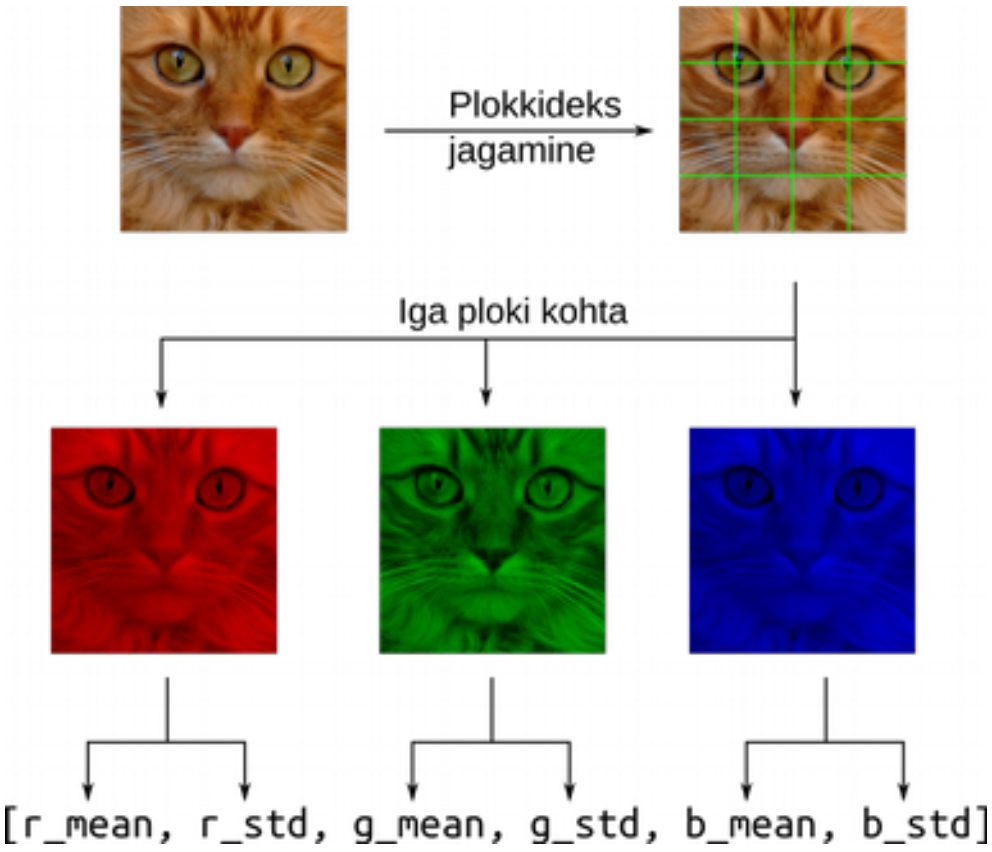
Joonis 10: Koerte otsingu tulemus hundikoera näitel

4.4 Valitud omadusvektori tööpõhimõte

Tehtud katsetuste tulemusena otsustas autor kasutada loomade piltide võrdlemiseks omadusvektorit, mis koosneb kolmest osas – värvimomendid, GLCM ja Hu momendid. Antud peatükis kirjeldatakse täpsemalt, kuidas need komponendid pildi alusel leitakse.

4.4.1 Värvimomendid

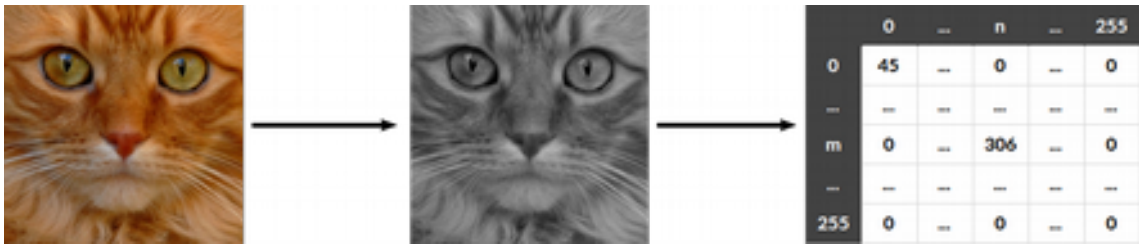
Enne värvimomentide leidmist jaotatakse pilt kuueteistkümneks võrdseks plokiks. Seejärel eraldatakse igast plokist punased, rohelised ja sinised komponendid. Iga komponendi kohta arvutatakse kaks momenti. Esimeseks momendiks on komponendi keskmine väärtus, millest võib järeldada pildi keskmist värvi. Teine moment on standardhälve, mis kirjeldab värvi jaotumist. Liites kõikide plokkide vektorid, moodustub 96-elemendiline vektor. Joonisel 11 on kujutatud värvimomentide leidmist graafiliselt.



Joonis 11: Värvimomentide leidmine

4.4.2 Halli taseme kaasesinemise maatriks

Halli taseme kaasesinemise maatriksi ehk GLCM-i arvutamiseks konverteeritakse pilt kõigepealt halltoonidesse ning siis hakatakse võrdlema ükshaaval kõrvuti asetsevaid pikslipaare (*m*, *n*). Kuna erinevaid halle toone on 256, siis moodustub maatriks suurusega 256x256, kus leitakse selliste kombinatsioonide arvud, kus *m* värvi piksli kõrval on *n* värvi piksel (Joonis 12).



Joonis 12: GLCM leidmine

Leitud maatriksi alusel on võimalik leida mitmeid erinevaid tekstuuri näitajaid nagu kontrast, erisugusus, homogeensus, nurga teine moment (*Angular Second Moment*, ASM), ühetaolisus jne. Leitavad näitajad jagatakse arvutamise põhimõtte järgi kolme gruppi.

Esimesse gruppi kuuluvate näitajate puhul korrutatakse maatriksi elemendid asukohast sõltuvalt läbi erinevate kaaludega. Kontrasti arvutamisel on maatriksi diagonaalil selleks kaaluks 0 ning liikudes servade poole kasvavad kaalud eksponentsiaalselt. Läbikorrutatud elementide summa iseloomustab pildi kontrastsust. Ka erisugususe ja homogeensuse arvutamisel korrutatakse maatriksi elemendid koefitsentidega läbi. Erisugususe puhul kasvavad kaalud aga lineaarselt ning homogeenususe puhul kahanevad eksponentsiaalselt. [8]

Teise grupi puhul võetakse arvesse seda, kui regulaarselt pikslite väärtused pildil esinevad. ASM ning ühetaolisuse väärtuste arvutamisel võetakse kaaludeks iga kombinatsiooni esinemise sagedus. [8]

Kolmas grupp koosneb erinevatest statistilistest näitajatest, mis maatriksi alusel arvutada võimalik – keskmine väärtus, standardhälve, korrelatsioon [8] .

Töös kasutatav omadusvektor sisaldab tekstuuri kirjeldamiseks halli taseme kaasesinemise maatriksi abil leitud näitajaid: kontrast, erisugusus, homogeensus, ASM

ning ühetaolisus. Nende väärtuste leidmiseks on scikit-image algoritmide kollektsioonis vajalikud funktsioonid (Joonis 13).

```
glcm = greycomatrix(grey, [1], [0], 256)
contrast = greycoprops(glcm, 'contrast')[0][0]
dissimilarity = greycoprops(glcm, 'dissimilarity')[0][0]
homogeneity = greycoprops(glcm, 'homogeneity')[0][0]
asm = greycoprops(glcm, 'ASM')[0][0]
energy = greycoprops(glcm, 'energy')[0][0]
```

Joonis 13: Halli taseme kaasesinemise maatriksist vajalike suuruste leidmine

4.4.3 Hu momendid

Kuju määramiseks kasutatakse Hu momente. Hu momente kirjeldati juba 1962. aastal valminud uurimuses kui kuju iseloomustavaid suurusi, mille väärtused ei sõltu pildi suurusest, rotatsioonist ega moonutamisest [18] . Nende suuruste leidmiseks tuleb taaskord konverteerida pilt halltoonidesse. Seejärel rakendatakse lävendi määramist. Käesolevas töös määrati lävendiks 127, mis tähendab, et kõik pikslid, mille RGB väärtus on suurem kui (127, 127, 127) muudetakse valgeks ning ülejäänud mustaks (Joonis 14).

Joonis 14: Lävendi määramine (*thresholding*)

OpenCV teegis leiduva *HuMoments* meetodi kasutamisel (Joonis 15) on tulemuseks seitsmest elemendist koosnev vektor. Kuna nende elementide väärtused on sõltumatud pildi suurusest ning looma näo asendist, siis sobivad need hästi kuju kirjeldamiseks antud töö puhul.

```
ret, th = cv2.threshold(grey, 127, 255, cv2.THRESH_BINARY)
moments = cv2.HuMoments(cv2.moments(th)).flatten()
```

Joonis 15: Hu momentide leidmine

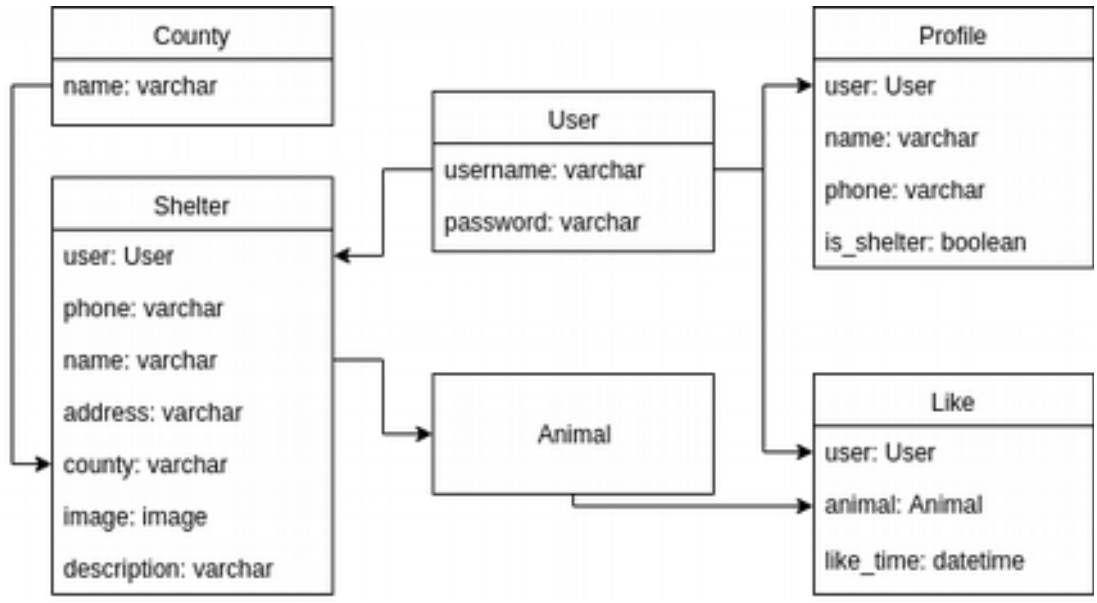
5 Rakenduse realisatsioon

Eelnevate peatükkide põhjal loodud sisupõhise pildiotsingu liides on aluseks veebirakendusele E-varjupaik, mille loomist käesolevas peatükis kirjeldatakse. Loodav veebirakendus koosneb kahest osast:

- kasutajapoolne rakendus - Vue.js[1] raamistik, EcmaScript 6[2] ja Webpack[3];
- serverirakendus – järgib REST põhimõtteid; Python, Django [4]ja Django Rest Framework[5]

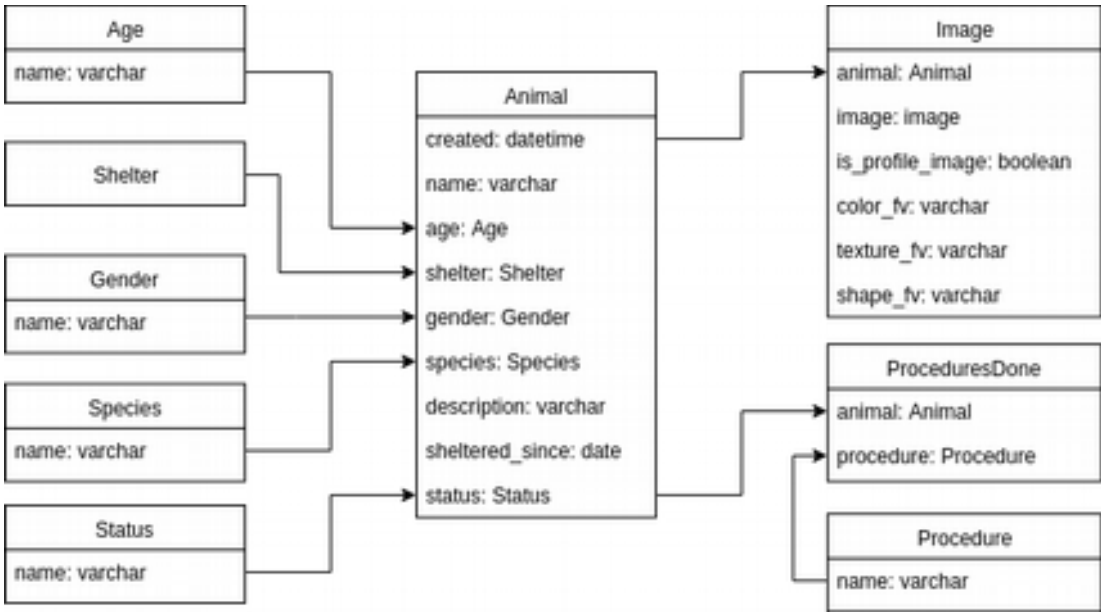
5.1 Andmebaasi disain

Andmebaasi füüsilist disaini kirjeldavad Joonis 16 ja Joonis 17.



Joonis 16: Andmebaasi disain I

[1] https://vuejs.org/
[2] http://es6-features.org
[3] https://webpack.github.io/
[4] https://www.djangoproject.com/
[5] http://www.django-rest-framework.org/



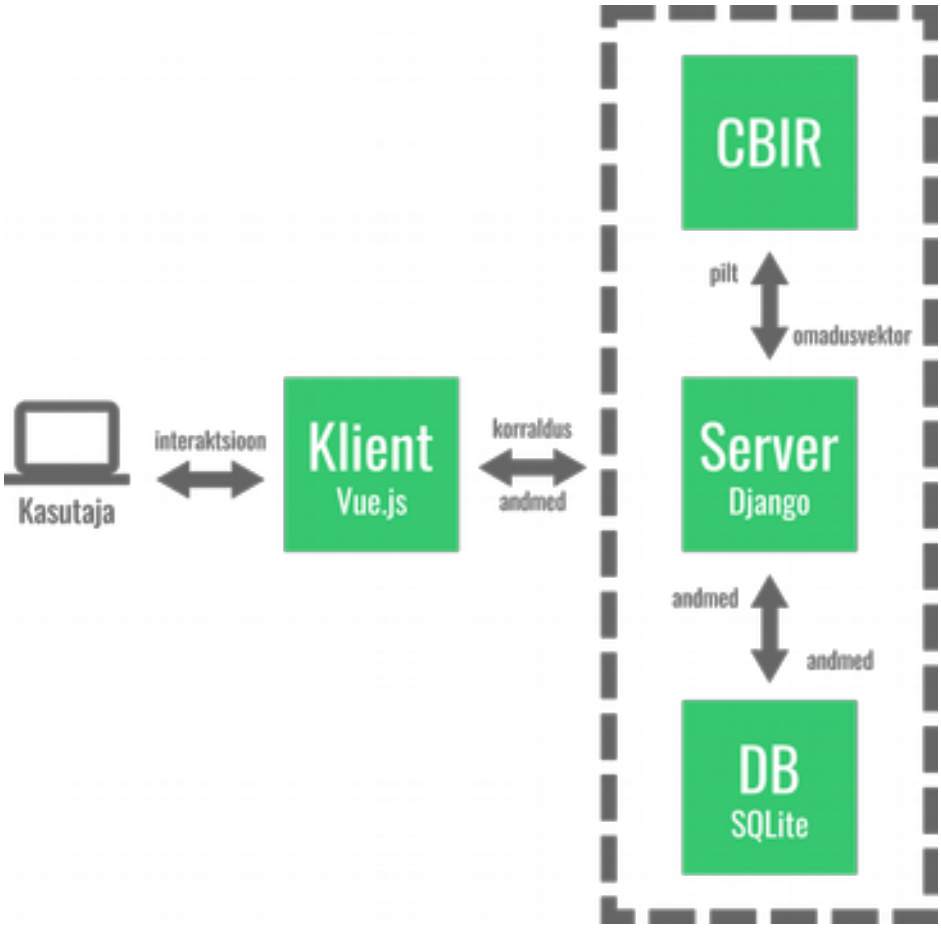
Joonis 17: Andmebaasi disain II

5.2 Rakenduse ülesehitus

Rakenduse ülesehitus põhineb klient-server arhitektuuril. Kui kliendil on vaja kas andmeid andmebaasi kirjutada või andmebaasist midagi lugeda, saadab ta serverile korralduse, mille viimane ka täidab.

Kui kasutaja lisab näiteks uue looma või sooritab pildi alusel otsingut, saadab kliendipoolne rakendus andmed serverile. Kui tekstilised andmed saab server kohe andmebaasi salvestada, siis piltide puhul tuleb läbida üks vahesamm. Loodud CBIR lahenduse abil teisendatakse pildid omadusvektoriteks. Iga omadusvektor koosneb omakorda kolmest komponendist, milleks on värvi, tekstuuri ja kuju kirjeldused. Kui tegemist on pildiga kassist, rakendadatakse enne omadusvektori eraldamist ka näotuvastus ning omadusvektor leitakse vaid näo osast. Saadud omadusvektorid salvestakse siis koos looma andmetega andmebaasi. Pildi alusel otsingut sooritades saadab server ühtlasi kliendile vastusena kollektsiooni sarnastest loomadest.

Süsteemi arhitektuur on kujutatud joonisel 18.



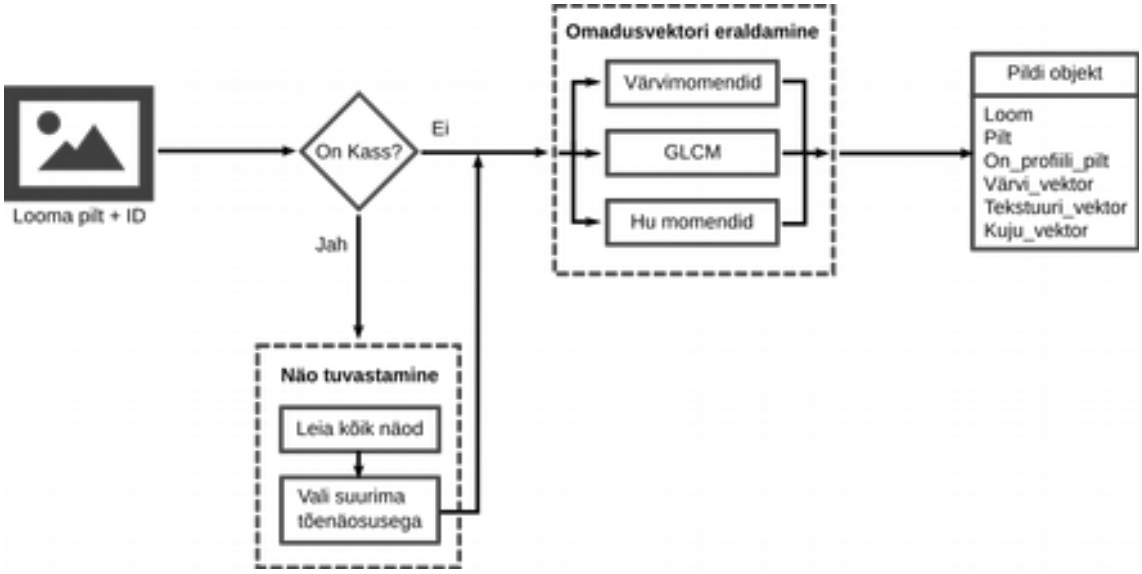
Joonis 18: Süsteemi arhitektuur (Autori joonis)

5.3 Realisatsioon

Kui rakenduse ülesehitus paigas, saab alustada realiseerimist. Käesolev jaotis kirjeldab rakenduse loomist. Rakenduses on võimaldatud kõik peatükis 2.1 välja toodud funktsionaalsed nõuded. Keskkonna kasutajad jagunevad tavakasutajateks ning varjupaiga kasutajateks. Igal varjupaiga kasutajal võib olla mitu varjupaika.

Rakenduses on kaks põhiobjekti: loom (*Animal*) ja pilt (*Image*). Mõlema andmeid saab täpsemalt vaadata andmebaasi disaini joonistelt 16 ja 17. Teise joonise puhul on oluline välja tuua, et omadusvektori jaoks on kolm erinevat välja: värvi, tekstuuri ja kuju omadusvektor. Kuna tekstuuri omadusvektori elementide suurusjärgud on teiste omadest tunduvalt väiksemad, tuleb enne piltidevahelise kogukauguse arvutamist korrutada tekstuuri kaugus piisavalt suure koefitsendiga.

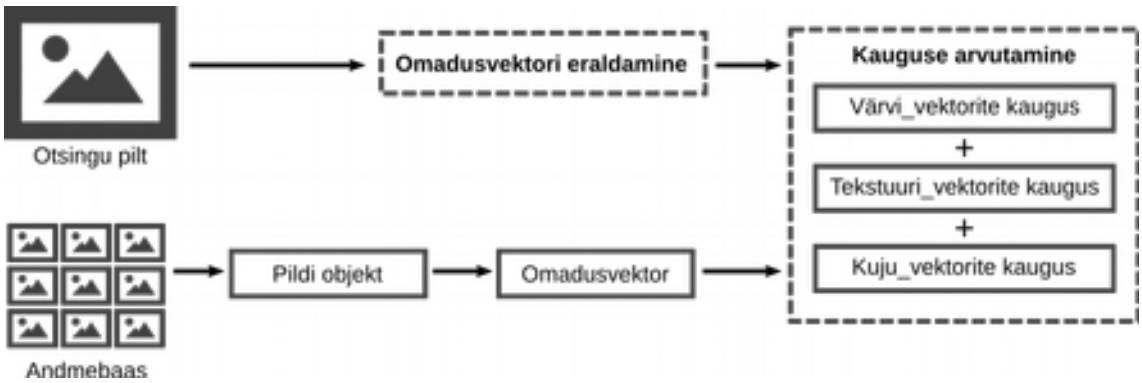
Kõige olulisemateks osadeks rakenduses on looma lisamine ning pildi alusel otsingu sooritamine. Looma lisamisel oodatakse kasutajalt kindlaid sisendeid: looma nimi või illustreeriv pealkiri, liik, sugu, vanus, varjupaik, varjupaigas alates, tehtud protseduurid, kirjeldus ja pildid. Kõigepealt lisatakse loom koos andmetega ning seejärel eraldatakse igast pildist omadused (Joonis 19).



Joonis 19: Omadusvektori eraldamine pildi üleslaadimisel

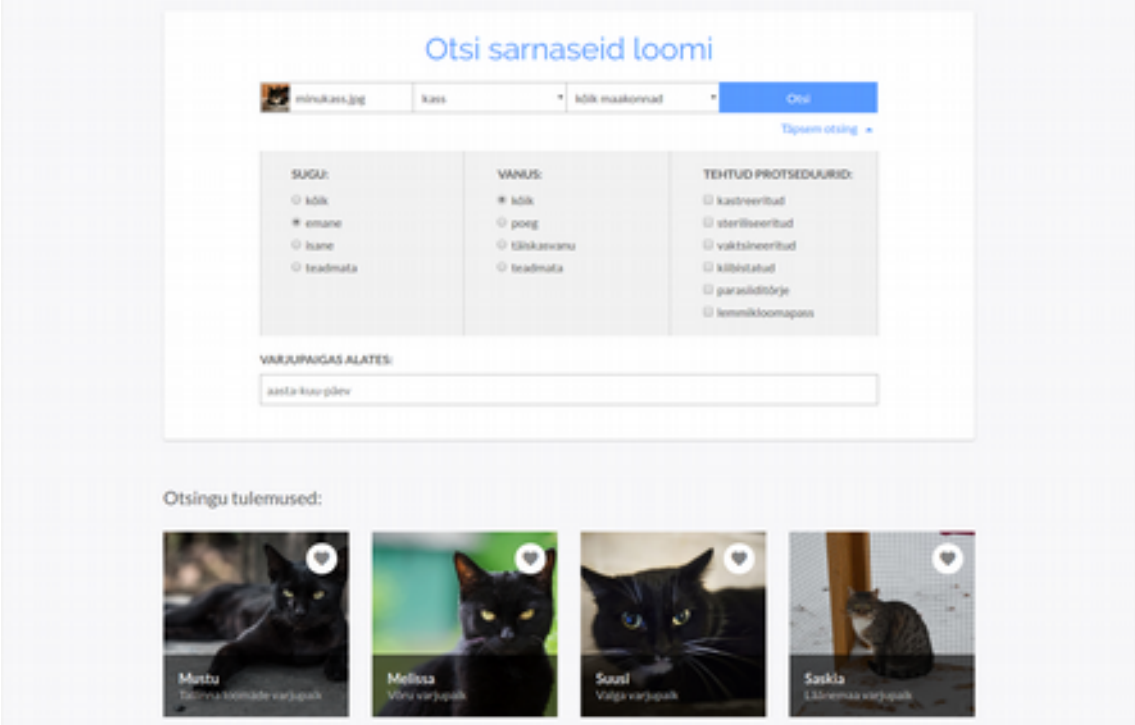
Eraldatud omaduste alusel toimub ka pildi alusel otsingu sooritamine. Selleks, et tagada võimalikult täpsed otsingutulemused, on võimalik anda kaasa mitmed parameetrid nagu liik, asukoht, sugu, vanus jne. Tänu nendele ei hakata omavahel võrdlema eri liikidest loomi.

Pildiotsingu sooritamisel eraldatakse kõigepealt otsingupildist omadusvektor. Selleks, et tulemus oleks võimalikult täpne, peaks otsingupilt olema hea kvaliteediga ning hästi valgustatud. Lisaks sellele peaks loom olema otsevaates ning pilt võiks sisaldada peamiselt looma nägu, sest kuigi kasside puhul on olemas ka näotuvastus, siis teiste loomade puhul seda ei ole.



Joonis 20: Piltidevahelise kauguse arvutamine

Pärast omadusvektori eraldamist filtreeritakse kõigist loomadest välja need, kes vastavad sisestatud kriteeriumitele. Seejärel võrreldakse ükshaaval kõikide filtreeritud loomade profiilipiltide omadusvektoreid ning arvutatakse nendevahelised kaugused. Antud töös kasutatakse arvutamiseks peatükis 3.5 kirjeldatud Chi-ruudus kauguse valemit. Saadud tulemuste alusel järjestatakse loomad – mida väiksem kaugus, seda sarnasem loom (Joonis 21).



Joonis 21: Sarnaste loomade otsing rakenduses

Lisaks kahele peamisele tegevusele realiseeriti ka teised funktsionaalsed nõuded. Varjupaiga kasutajad saavad lisatud loomi muuta, märkida aktiivseks või mitteaktiivseks ning hallata oma varjupaikade andmeid. Tavakasutaja saab vaadata aktiivseid loomi, otsida neid erinevate parameetrite alusel ning märkida loomi meeldivaks. Rakenduse erinevate vaadete ekraanipildid on nähtavad lisa 6 all.

5.4 Edasise arengu võimalused

Antud rakenduse loomisel oli ka mitmeid ideid, mille abil edaspidi rakendust paremaks muuta. Sarnaste koerte ja muude loomade otsimise optimeerimiseks on mitu võimalust. Ühe ideena võiks treenida näotuvastuse ka teiste loomade jaoks. Nii ei peaks lisama nende puhul profiilipildiks sellist pilti, millel moodustab valdava osa nägu. Teine võimalus, mis muudaks näo asukoha kindlaks, oleks anda kasutajale võimalus parandada vajadusel leitud näo asukohta. Pärast profiilipildi valimist ning looma andmete sisestamist võiks kuvada kasutajale profiilipildi koos ristkülikuga, mis kujutab leitud näo asukohta. Kasutaja saaks muuta ristküliku asukohta ja suurust. Parandusi saaks omakorda kasutada näotuvastuse paremaks muutmisel.

E-varjupaiga kui sotsiaalse platvormi jaoks võiks lisada võimaluse loomade profiile sotsiaalmeedias jagada. See lihtsustaks ühtlasi varjupaikade vaeva, kes praegu käsitsi postitusi trükivad. Inimestel, kes on huvitatud looma adopteerimisest, võiks olla võimalus võtta varjupaigaga ühendust otse läbi keskkonna. Antud töö raames lisati juba lehele adopteerimise nupp, millel hetkel funktsionaalsus puudub.

6 Kokkuvõte

Antud töö eesmärkideks oli realiseerida lahendus, mis võimaldab otsida pildil olevale loomale sarnast ning luua rakendus, mis koondab infot eri varjupaikades olevate loomade kohta ning kasutab eelpool mainitud CBIR lahendust sarnase looma leidmiseks.

Esimese eesmärgi saavutamiseks uuriti sisupõhise pildituvastuse olemust ning erinevaid võimalusi pildist omaduste eraldamiseks. Sobiva võimaluse valimiseks koostati pildikogum varjupaikades viibivatest loomadest ning võrreldi erinevate viiside tulemusi. Lahendamaks probleemi, et tausta osakaal pildil mõjutab loomade võrdlemise tulemust negatiivselt, rakendati treenitud kasside nägude tuvastajat. Teiste loomade puhul eeldatakse, et profiilipildil moodustab piisavalt suure osa looma nägu. Tehtud katsetuste tulemusena valiti välja kolmest erinevast osast koosnev omadusvektor, mis koondab endasse nii pildi värvi, tekstuuri kui kuju kirjeldavad omadused. Piltide sarnasuse hindamiseks leitakse nende omadusvektorite vahelised kaugused, mille arvutamiseks kasutatakse Chi-ruudus valemit.

Teise eesmärgi saavutamiseks kirjutati veebirakendus, mis põhineb klient-server arhitektuuril ning kasutab väljatöötatud sarnaste loomade otsimise lahendust. Loodud rakenduses implementeeriti nii tavakasutajate kui varjupaikade osa, muutes keskkonna kasutuskogemus terviklikuks. Varjupaigad saavad luua enda loomadele profiile ning tavakasutajad saavad lisaks loomade profiilide vaatamisele, lisada neid ka meeldivaks. Rakenduse kõige olulisem osa, mille poolest Eesti konkurendist ka erinetakse, on sarnaste loomade otsing. Kasutaja saab otsingusse lisada pildi, millel olevale loomale sarnast otsitakse. Lisaks pildile on hulgaliselt parameetreid, mis muudavad otsingu veelgi täpsemaks. Andes kaasa looma liigi, vanuse, soo, kadumise kuupäeva jne, saab muuta võrreldavate loomade hulga tunduvalt väiksemaks ning suurendada võimalust leida otsitav loom.

Kasutatud kirjandus

[1] Varjupaikade MTÜ aastal 2016
[WWW] http://www.varjupaik.ee/loomade-statistika (27.04.2017)

[2] Hulkuvate loomadega seotud probleemistik Eesti kohalikes omavalitsustes
[WWW] http://www.loomakaitse.ee/wp-content/uploads/2016/05/Hulkuvate-loomadega-seotud-probleemistik-Eesti-kohalikes-omavalitsustes_2015.pdf (27.04.2017)

[3] Shaefer, G.
Content-Based Image Retrieval - Some Basics
Advances in Intelligent and Soft Computing, Volume 103, 2011

[4] Yuea, J., Lib, Z., Liub, L., Fub, Z.
Content-based image retrieval using color and texture fused features
Mathematical and Computer Modelling, Volume 54, Issues 3–4, August 2011, Pages 1121–1127

[5] Mistry, Y., Ingole, D.T., Ingole, M.D.
Content based image retrieval using hybrid features and various distance metric
Journal of Electrical Systems and Information Technology, In Press, Corrected Proof, January 2017

[6] Tammert, M. Värviruum
[WWW] http://opiobjektid.tptlive.ee/Varviop/VT_Varvikorrastus_Varviruum.htm (27.04.2017)

[7] Huang, J., Zabih, R. Combining Color and Spatial Information for Content-based Image Retrieval
[WWW] http://www.cs.cornell.edu/rdz/Papers/ecdl2/spatial.htm (27.04.2017)

[8] The Grey Level Co-occurrence Matrix
[WWW] http://www.fp.ucalgary.ca/mhallbey/the_glcm.htm (27.04.2017)

[9] Rosebrock, A. Building an Image Search Engine: Defining Your Image Descriptor (Step 1 of 4)
[WWW] http://www.pyimagesearch.com/2014/02/03/building-an-image-search-engine-defining-your-image-descriptor-step-1-of-4/ (21.03.2017)

[10] Canny Edge Detection
[WWW] http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html (27.04.2017)

[11] Mallick, S. Histogram of Oriented Gradients
[WWW] http://www.learnopencv.com/histogram-of-oriented-gradients/ (27.04.2017)

[12] Rosebrock, A. Building an Image Search Engine: Defining Your Similarity Metric (Step 3 of 4)
[WWW] http://www.pyimagesearch.com/2014/02/17/building-an-image-search-engine-defining-your-similarity-metric-step-3-of-4/ (28.04.2017)

[13] Image Thresholding
[WWW] http://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html (08.05.2017)

[14] Distance Measures
[WWW]
http://www.umass.edu/landeco/teaching/multivariate/readings/McCune.and.Grace.2002.chapter6.pdf (08.05.2017)

[15] E-Teatmik
[WWW] http://www.vallaste.ee/ (10.05.2017)

[16] Face Detection using Haar Cascades
[WWW] http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html (17.05.2017)

[17] Ljubovic, V., Supic, H.
Comparative Study of Color Histograms as Global Feature for Image Retrieval Information Communication and Automation Technologies (ICAT) 2013 XXIV International Symposium on, pp. 1-5, 2013.

[18] Hu. M
Visual Pattern Recognition by Moment Invariants, IRE Transactions on Information Theory, 8:2, pp. 179-187, 1962.

Lisa 1 – Varjupaikade vastused

Kokkuvõte varjupaikade vastustest seoses loomade varjupaiku koondava veebirakenduse E-varjupaik loomise ideega.

Varjupaik: **Varjupaikade MTÜ** (koondab enda alla 7 varjupaika üle Eesti)

Huvitatud kasutamisest?	Jah
Kanalid, mida praegu kasutavad	Loom24.ee – ei ole kõige mugavam lahendus ei töötajatele ega „klientidele”
Kommentaarid	See on suurepärane idee ja loomulikult oleme nõus kasutama! Sooviks oma kodulehega suhtlust võimalikult lihtsaks teha.

Varjupaik: **Tartu Koduta Loomade Varjupaik**

Huvitatud kasutamisest?	Jah
Kanalid, mida praegu kasutavad	Koduleht Facebook Tartu Postimees
Kommentaarid	Võiks kodulehelt ise info üles korjata.

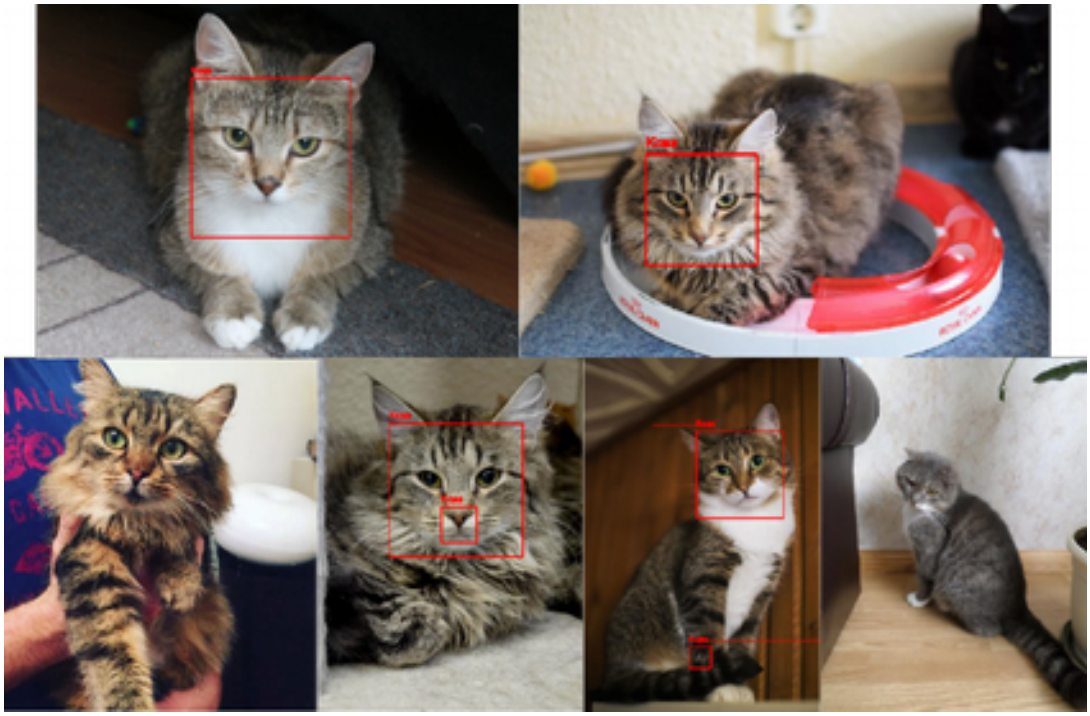
Varjupaik: **MTÜ Saaremaa Lemmikloomade Turvakodu**

Huvitatud kasutamisest?	Jah
Kanalid, mida praegu kasutavad	Facebook – enamus loomi leiab sealt kodu Koduleht Loom24 – Meie loomadele kodu leidmisel ei ole nimetatud portaal kahjuks olulist osa mänginud, kuigi liitusime portaali loomisel. AnimalLife Kohalikud lehed
Kommentaarid	Rakenduse kasutamine sõltuks paljuski kuulutuse üles panemiseks ja haldamiseks kuluvast ajalisest ressursist

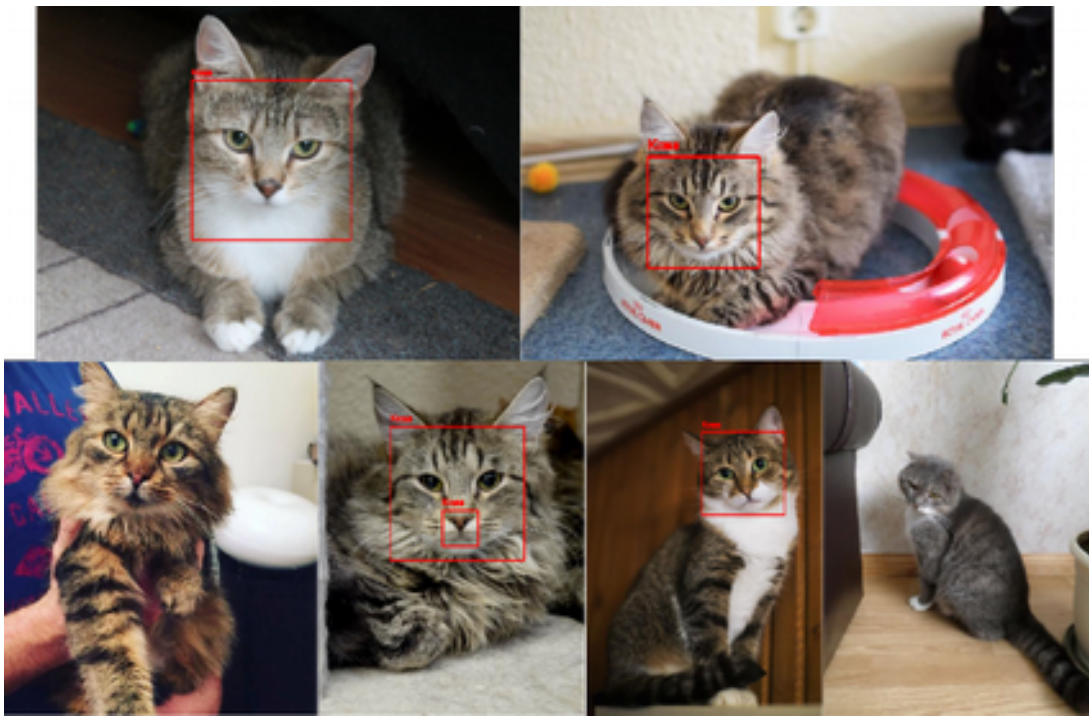
Lisa 2 – Kasside tuvastamine OpenCV klassifitseerijatega

OpenCV repositoorium sisaldab kahte kasside nägude tuvastajat, mille tulemused on näha allpool.

Mõlema klassifitseerija tulemused jäid aga reaalsete varjupaikade kasside piltide puhul kesiseks. Otsevaates kasside nägusid ei suudetud tuvastada, rääkimata kassidest, kes ei vaadanud otse kaamerasse. Kuigi üks tuvastajatest oli teise edasiarendus, ei olnud selle täpsus siiski eelmisest parem ning paljudel piltidel ei leitudki nägu üles.



Esimese OpenCV klassifitseerija tuvastatud näod

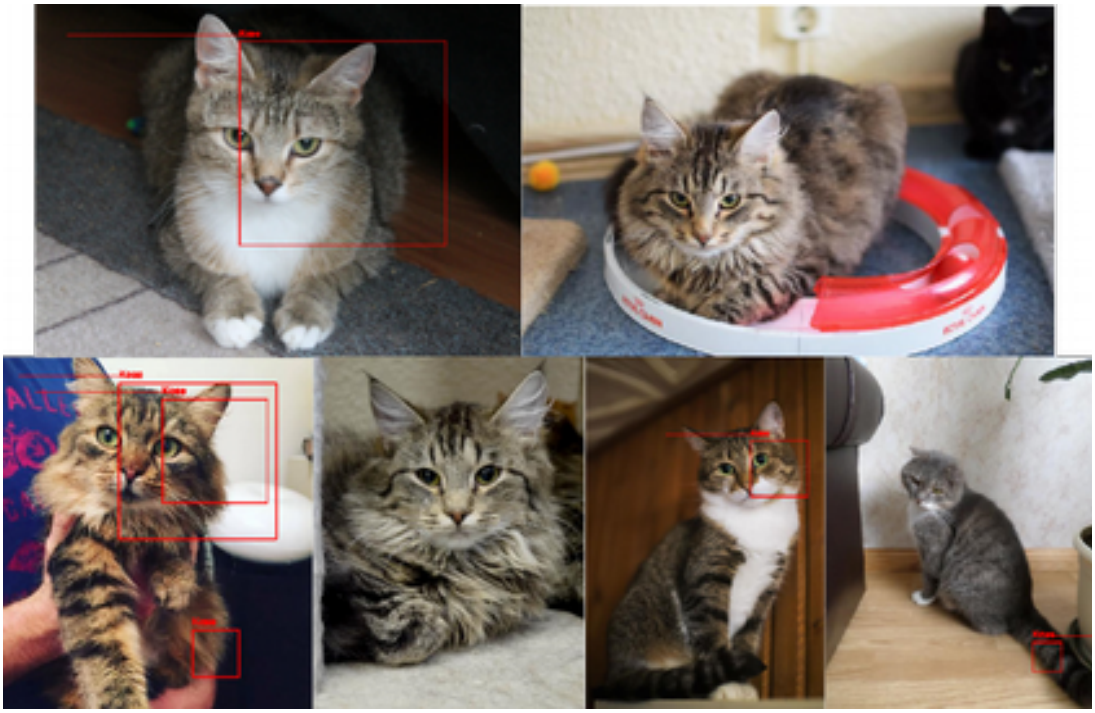


Teise OpenCV klassifitseerija poolt tuvastatud kasside näod

Lisa 3 – Kasside tuvastamine isetreenitud klassifitseerijaga

Kuna OpenCV poolt pakutavate kasside nägude tuvastajate tulemused ei olnud kõige paremad, proovis autor kätt ka ise klassifitseerija treenimisega. Täpsemalt saab sellest lugeda peatükis 4.2.1.

Kahjuks olid isetreenitud tuvastaja tulemused veelgi halvemad ning tuvastatud „näod" asusid enamasti vales kohas, näiteks saba, käpa või hoopiski seina peal.



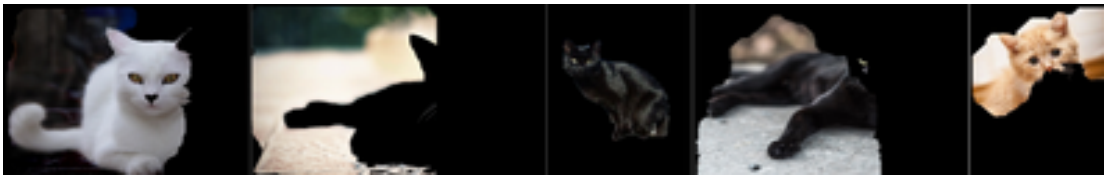
Isetreenitud klassifitseerija tuvastatud näod

Lisa 4 – Kasside tuvastamine teiste meetoditega

Eesmärgiga leida korrelatsioone kasside asukohtades piltidel, katsetati mitmeid erinevaid meetodeid. Esimesena prooviti kontuuride leidmist, et eraldada kass taustast. Olenevalt pildi kvaliteedist, taustast ning kassist endast, olid piirjoonte leidmise tulemused erinevad. Kui just ei olnud tegemist väga kontrastse pildiga, näiteks üleni must kass täiesti valgel taustal, siis kahjuks kontuuride abil kassi taustast eraldada ei olnud võimalik.

Kontuuride leidmine

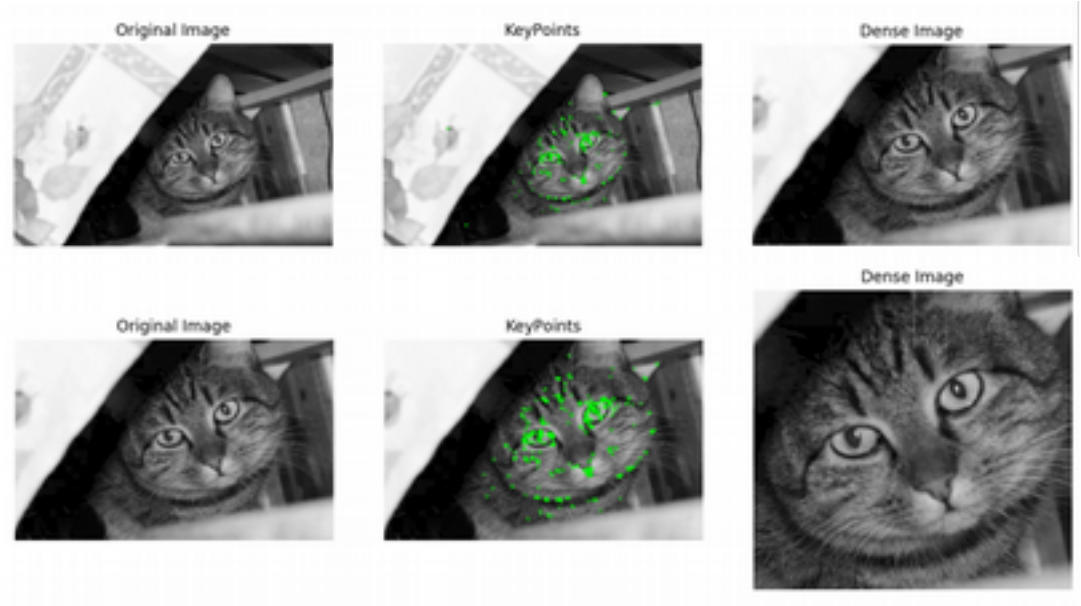
Teine katsetus hõlmas segmenteerimist, mille eesmärgiks oli sarnaselt kontuuride leidmisele, eraldada terve kass taustast. Kahjuks ebaõnnestus ka see variant täpselt samadel põhjustel. Nagu allpool näha, siis ühe pildi puhul töötas segmenteerimine suurepäraselt, teiste puhul aga jättis soovida.



Segmenteerimine

Kolmas test hõlmas võtmepunktide leidmist. Võtmepunktide all peetakse silmas selliseid koht pildil, mis on suure kontrastsusega, tähistavad nurki või servasid. Võtmepunktide leidmisel selgus, et suurem osa võtmepunktidest asub kassi näos –

silmad, nina jne. Nagu eelmistegi meetodite puhul, olid osade piltidega tulemused suurepärased. Teiste puhul, kus taustal oli näiteks kiri või lihtsalt palju asju, muutus näo leidmine ebatäpseks ning keeruliseks.



Võtmepunktide kasutamine

Lisa 5 – Omadusvektori valimisel tehtud võrdluste tulemused

RGB histogramm

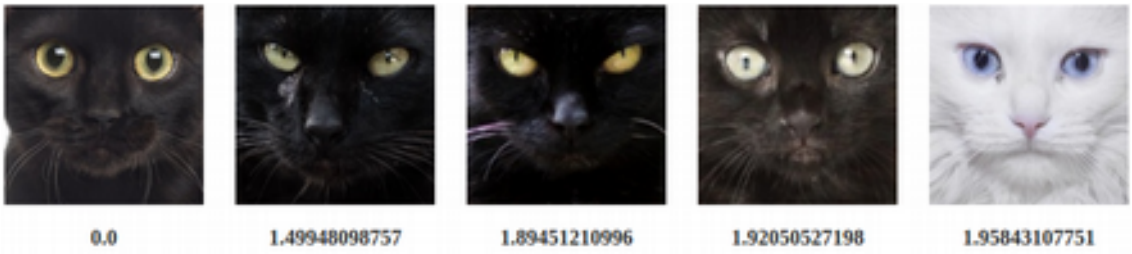
0.0 0.412792600017 0.671834011846 0.674265216897 0.72754871079

0.780742736032 0.820440962134 0.853893626554 0.9446444097 1.06111046054

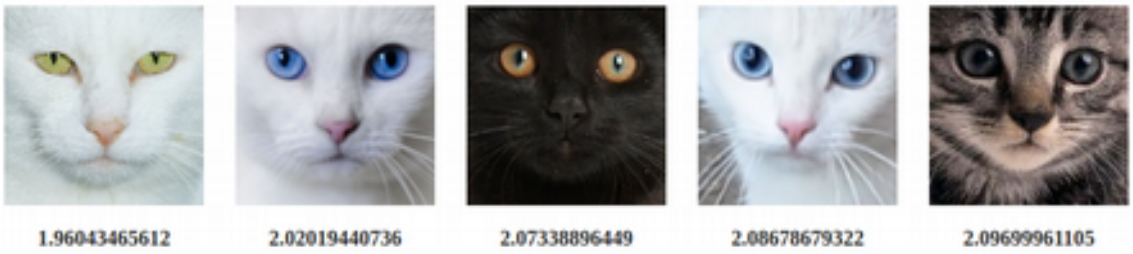
0.0 0.364770341964 0.51207816739 0.796439845794 0.898927574439

0.911488432872 0.945895725313 0.972165580001 1.0030369856 1.04758151827

HSV histogramm



0.0 1.49948098757 1.89451210996 1.92050527198 1.95843107751

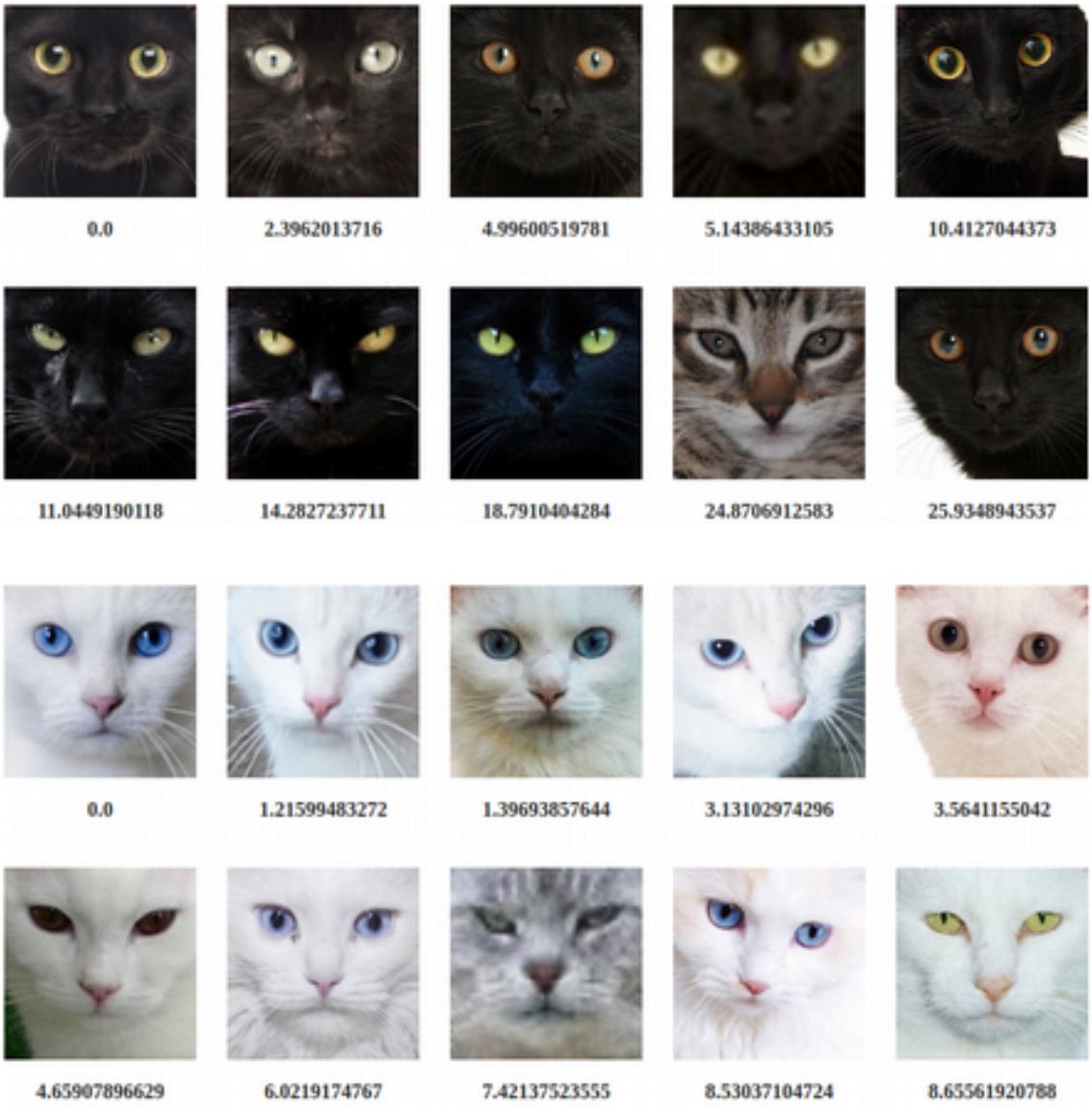


1.96043465612 2.02019440736 2.07338896449 2.08678679322 2.09699961105

0.0 0.161757255748 0.341121559554 0.80694041633 0.875174011882

0.900864266231 1.11639674371 1.12644898829 1.4276653916 1.56854212119

Värvimomendid


0.0
2.3962013716
4.99600519781
5.14386433105
10.4127044373
11.0449190118
14.2827237711
18.7910404284
24.8706912583
25.9348943537
0.0
1.21599483272
1.39693857644
3.13102974296
3.5641155042
4.65907896629
6.0219174767
7.42137523555
8.53037104724
8.65561920788

LAB histogramm


0.0
0.734329158349
0.752749839975
0.865313807749
0.933144543421
0.944872020456
1.01117723244
1.06334908441
1.16626965794
1.25555379725


0.0
0.240470594305
0.711460208032
0.78753832977
0.793678247146
0.84099819582
0.95710691654
1.01416345414
1.12427430816
1.40704570873

YUV histogramm

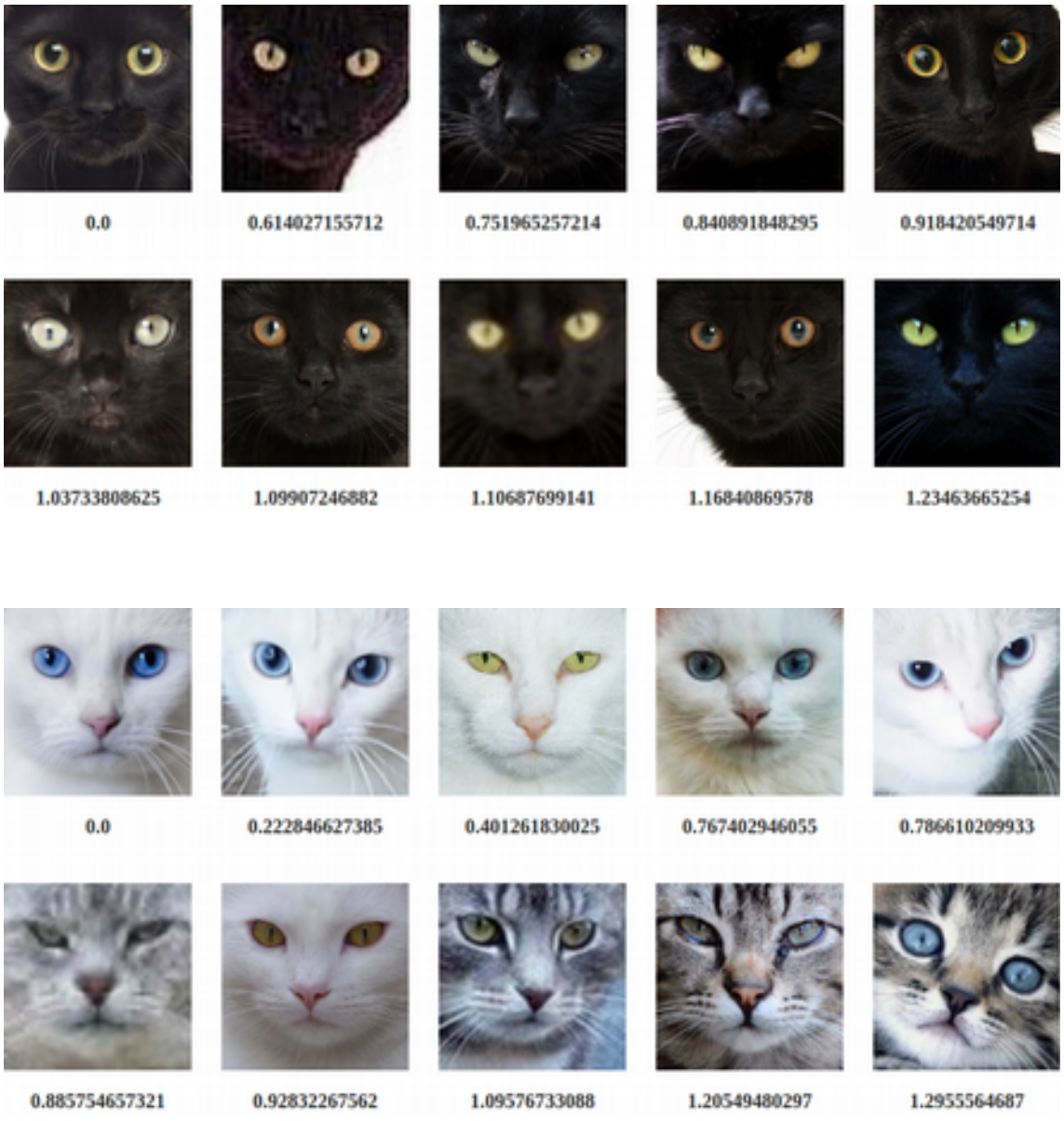
0.0 0.607954035331 0.751129106791 0.833505118066 0.919706913018

1.03269817899 1.09648675513 1.10121312095 1.16795104976 1.22747953883

0.0 0.215110999238 0.397298030192 0.757972201078 0.772895309603

0.86116678528 0.925766894941 1.06750826567 1.19067229177 1.27126923988

YcbCr histogramm

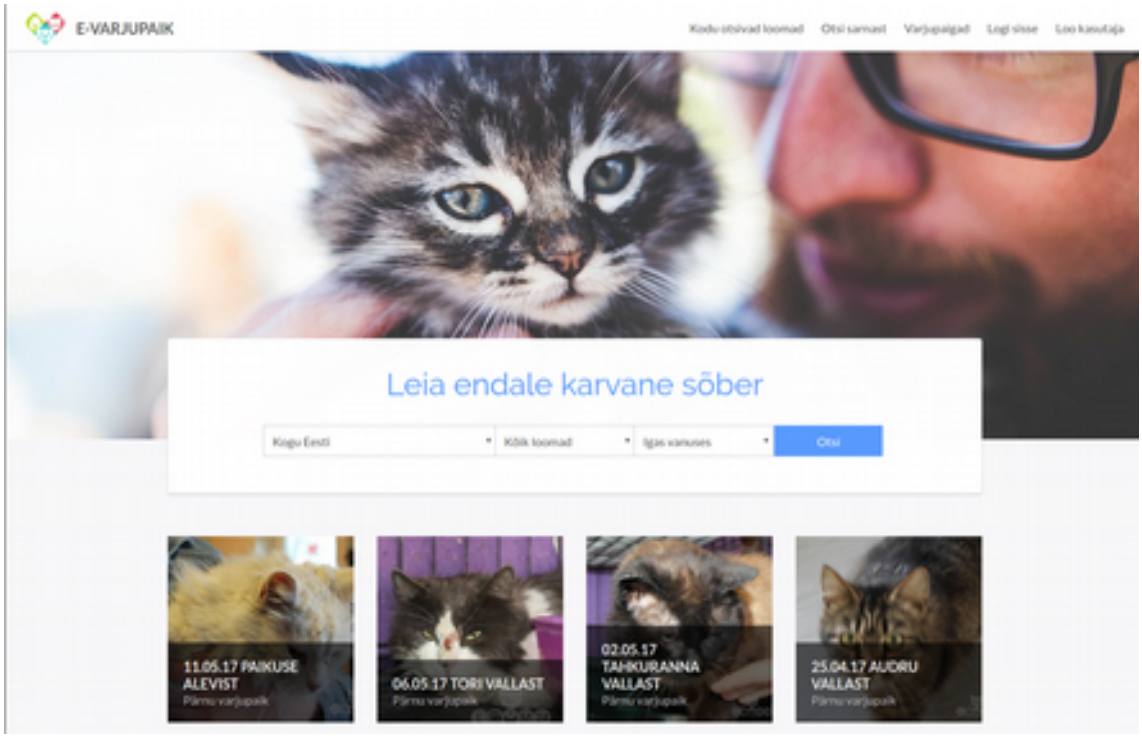

0.0
0.614027155712
0.751965257214
0.840891848295
0.918420549714
1.03733808625
1.09907246882
1.10687699141
1.16840869578
1.23463665254
0.0
0.222846627385
0.401261830025
0.767402946055
0.786610209933
0.885754657321
0.92832267562
1.09576733088
1.20549480297
1.2955564687

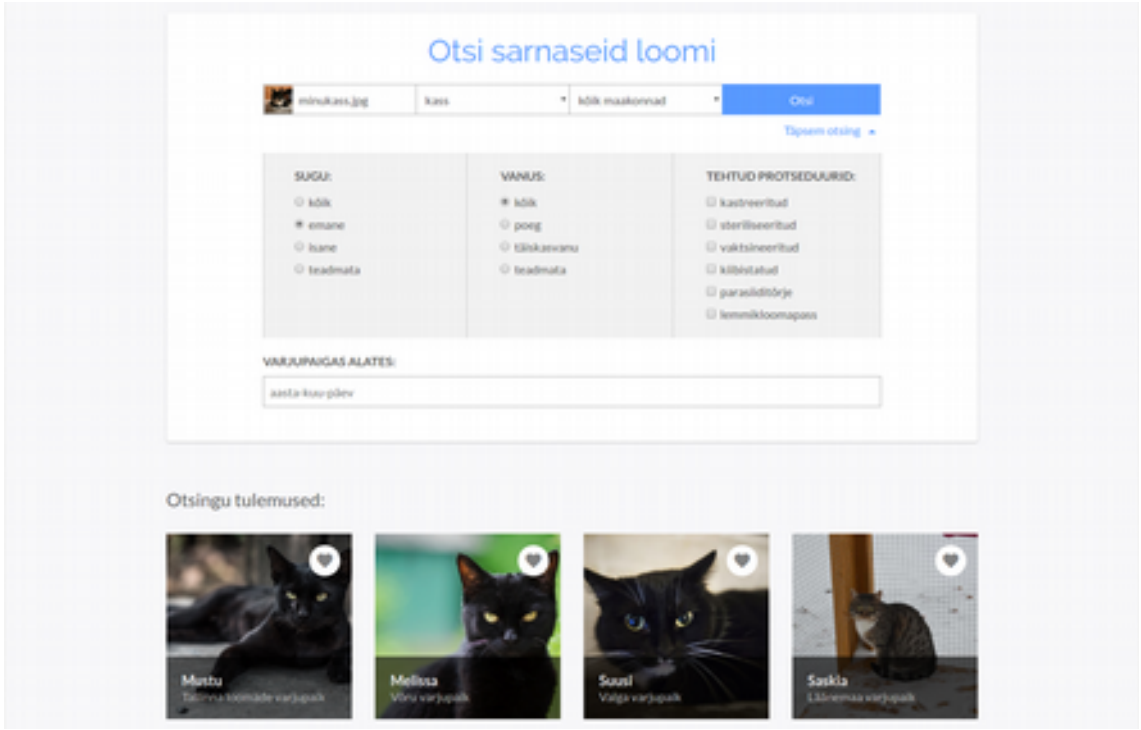
Kolmest komponendist koosnev omadusvektor

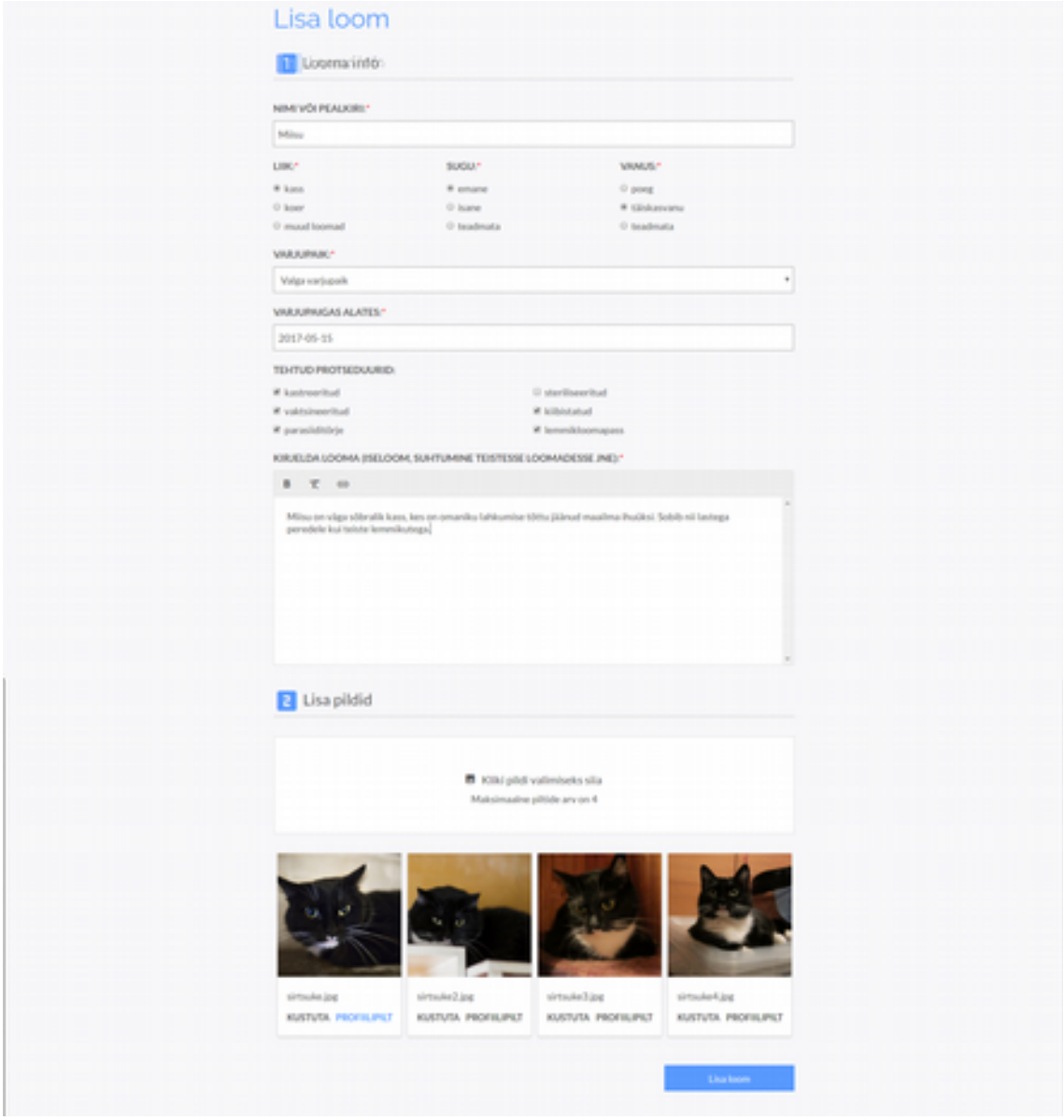
Väike pikakarvaline kirju koer

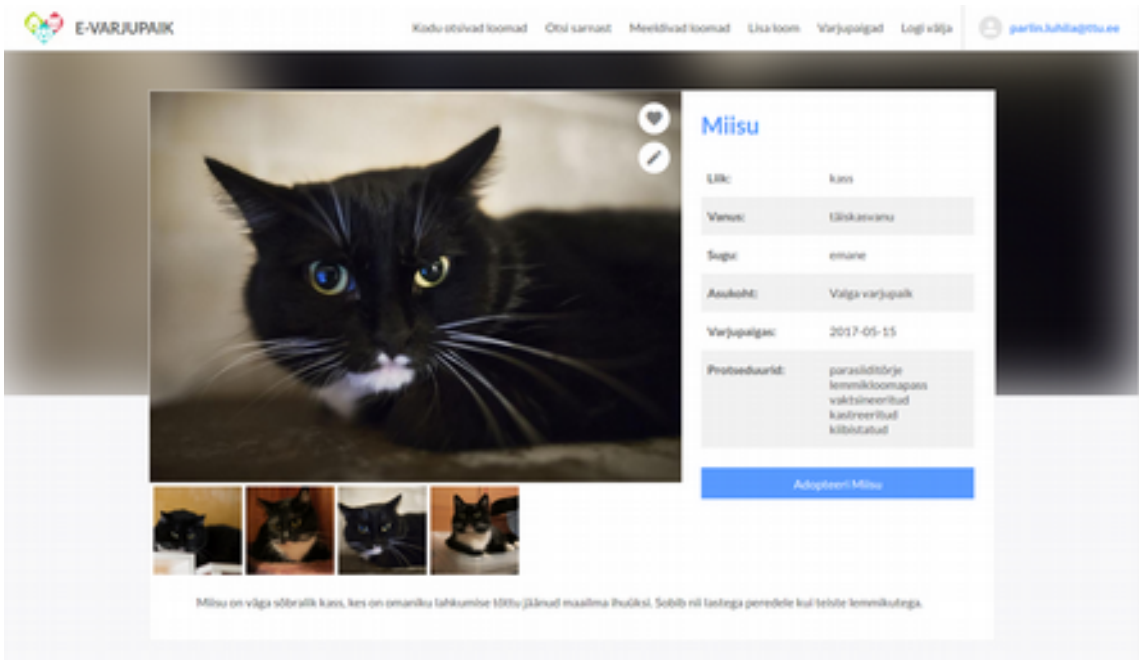
Hundikoera tüüpi koer

Lisa 6 – Rakenduse E-varjupaik ekraanipildid






Lisa loom
1 Looma info
NIMI VÕI PEALKIRI:*
Miisu
LIIK:*
kass
koer
muud loomad
SUGU:*
emane
isane
teadmata
VANUS:*
poeg
täiskasvanu
teadmata
VAKSIINIGAS ALATES:*
2017-05-15
TEHTUD PROTSEDUURID:
kastreeritud
steriliseeritud
vaktsineeritud
kiibistatud
parasiidtõrje
lemmikloomapass
KIRJELDA LOOMA (ISELOOM, SUHTUMINE TEISTESSE LOOMADESSE JNE):*
Miisu on väga sõbralik kass, kes on omaniku lahkumise tõttu jäänud maalima Puukõsi. Sobib nii lastega peredele kui teiste lemmikutega.
2 Lisa pildid
Klõki pildi valimiseks siia
Maksimaalne piltide arv on 4
siirtuke.jpg
KUSTUTA PROFIILIPILT
siirtuke2.jpg
KUSTUTA PROFIILIPILT
siirtuke3.jpg
KUSTUTA PROFIILIPILT
siirtuke4.jpg
KUSTUTA PROFIILIPILT
Lisa loom


E-VARJUPAIK
Kodu otsivad loomad
Otsi sarnast
Meeldivad loomad
Lisa loom
Varjupaigad
Logi välja
Miisu
Liik:
kass
Vanus:
täiskasvanu
Sugu:
emane
Asukoht:
Valga varjupaik
Varjupaigas:
2017-05-15
Protseduurid:
parasiiditõrje
lemmikloomapass
vaktsineerilud
kastreerilud
kiibistatud
Adopteeri Miisu
Miisu on väga sõbralik kass, kes on omaniku lahkumise tõttu jäänud maailma ihuüksi. Sobib nii lastega peredele kui teiste lemmikutega.