TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Karoliina Koppel 143044IAPB

PILDITÖÖTLUS TUNNUSVEKTORI KOOSTAMINE SENSORI ANDMETEST

Bakalaureusetöö

Juhendaja: Martin Rebane
MSc

Tallinn 2018

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: Karoliina Koppel

23.05.2018

Annotatsioon

Eesmärgiks oli luua programm kasutades laserskanneri abil tehtud 3D skaneeringuid, et tuvastada mobiiltelefonide korpusel leiduvaid karakteristikuid, milleks võivad olla kriim, täke või kaamera. Karakteristikute leidmiseks kasutati erinevaid pilditöötlusmeetodeid nagu Gaussi hägustamine, lävisegmenteerimine ja Canny ääretuvastus.

Töö käigus valmis programm, mille sisendiks on laserskanneri 3D andmed mobiiltelefonist, väljundiks korpuse pinnalt tuvastatud karakteristikud, mida kasutati Haar klassifitseerija treenimisel. Treenitud klassifitseerija abil tuvastati korpusel asuvaid kriime, täkkeid ja mõrasid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 17 leheküljel, 4 peatükki, 13 joonist.

Abstract

Image processing feature extraction from sensor data

The purpose of this thesis is to create a program using 3D scans from a laser scanner, that are made of mobile phones, to extract features from phone case. The features may be a scratch, a nick or a camera. Different image processing methods are used to extract the features from a phone's surface, like Gaussian blur, thresholding, and Canny edge detection.

This thesis sets the target to create a program, which has 3D laser scanner data as input. Outputs are features, that are used to train Haar classifier. Trained classifier is used to detect scratches, nicks and cracks.

Thesis is written in Estonian and contains 17 pages of text, 4 chapters, 13 figures.

Lühendite ja mõistete sõnastik

Karakteristik	*Feature*. Suvaline pilti või pildi piirkonda kirjeldav tunnusomadus.
Lävisegmenteerimine	*Thresholding.* Pildi segmentimise protsess, mis põhineb pildi teatud atribuudi läve kasutamisel binaarpildi loomiseks.
Mask	Märgi muster, mille abil juhitakse teise märgimustri osade allesjättu või kõrvaldamist.
OpenCV	*Open Source Computer Vision Library* on avatud lähtekoodiga raalnägemise algoritme sisaldav teek, mis on kasutatav C++, Python ja Java liidestes[1]
PCD	*Point Cloud Data.* Punktpilve kirjeldav failiformaat[2]
PCL	*Point Cloud Library*. Punktpilv teek on 2D/3D pildi ja punktpilve töötluse teek, mis sisaldab endas erinevaid raalnägemise algoritme[3]
Pilditöötlus	*Image processing*. Andmetöötlussüsteemi kasutamine piltide loomiseks, skaneerimiseks, analüüsimiseks, täiustamiseks, tõlgendamiseks või kuvamiseks.
Raalnägemine	*Computer Vision.* Funktsionaalüksuse võime hõivata, töödelda ja interpreteerida visuaal andmeid.
RGB	Liitvärvimudel, milles erinevaid värvitoone saadakse kolme põhivärvuse – punane, roheline ja sinine – liitumisel.
Tunnusvektor	*Feature vector*. Karakteristikute n-järjend, mida saab kasutada pildil olevate alade või objektide kirjeldamiseks

[1] https://opencv.org/

[2] http://pointclouds.org/documentation/tutorials/pcd_file_format.php

[3] http://pointclouds.org/about/

[4] http://www.eki.ee/dict/its

Sisukord

Jooniste loetelu

1 Sissejuhatus

Mobiiltelefonide tööstuses on olulisel kohal ka kasutatud seadmete edasimüük. Selle jaoks on oluline hinnata eelnevalt seadme korrasolekut. Üheks oluliseks aspektiks edasimüügil on mobiili välimus. Kasutatud mobiililt leiab enamasti kulumisest tingitud täkkeid ja kriime, mis vähendavad seadme edasimüügi väärtust. Kasutatud mobiiltelefonide edasimüüjal on vaja tuvastada, kas telefoni korpusel on kriime ja täkkeid ning need klassifitseerida sügavuse/suuruse järgi.

Käesolev töö valmis koostöös Teleplan Estonia OÜ-ga. Teleplan Estonia arendab tarkvara FocalSpec laserskannerile, mille abil hakatakse tuvastama ja klassifitseerima kriime mobiiltelefoni ekraanil. Kõik algandmed, mis on kasutatud antud töös, pärinevad FocalSpec sensoriga skaneeritud mobiiltelefonidest.

Antud töö esmärgiks on luua programm kasutades laserskanneri abil tehtud 3D skaneeringuid, et tuvastada mobiiltelefonide korpusel leiduvaid vead. Skannerist saadud andmed on vaja töödelda, et need oleks kasutatavad pilditöötluses. Eesmärgini saamiseks planeeritakse esmalt tuvastada korras telefonile iseloomulikud karakteristikud. Selleks kasutatakse erinevaid pilditöötlusalgoritme ning ääretuvastusmeetodeid, mis toovad mobiilikorpustel olevad karakteristikuid muust taustast esile. Saadud andmeid on võimalik kasutada Haar klassifitseerijate treenimisel. Saadud klassifitseerijaid saab kasutada karakteristikute tuvastamiseks. Erinevad karakteristikud võivad olla nii kriimud, täkked kui ka kaamera või kõlar. Töö käigus on vaja luua programm, mis eristab mobiili korpusel olevad vead kaamerast ja kõlarist ning kuvab saadud tulemuse.

Loodud lahenduse kood on üleval Githubi repositooriumis[1].

[1] https://github.com/KaroliinaKoppel/pcd_feature_extraction

2 Karakteristikute tuvastamine

Karakteristikud raalnägemises (*Computer Vision*) ja pilditöötluses (*Image Processing*) on informatsiooni kogumid, mis kirjeldavad pildil asuvaid objekte või alasid. Karakteristikuteks võivad mobiiltelefonil olla kaamera, nupud või kriimud. Need eristuvad telefoni korpusest märgatavalt. Karakteristikute tuvastamiseks on palju erinevaid tehnikaid. Raalnägemise abil saame tuvastada pildil või punktpilves (*Point Cloud Data*) välja paistvate omaduste piire, nurki jms.

2.1 Pilditöötluse alused

Selleks, et alustada pildilt tunnusvektorite tuvastamist, oleks vaja eelnevalt pildil müra vähendada. Pildi heleduse kirjeldamiseks saab infot histogrammi abil. Punktoperatsioonide abil saab pildi iga piksli ümber arvutada ümbritsevate pikslite väärtuste järgi. Grupioperatsioonidega saab arvutada näiteks pildi keskmist väärtust või moodi. [1]

2.1.1 Histogramm

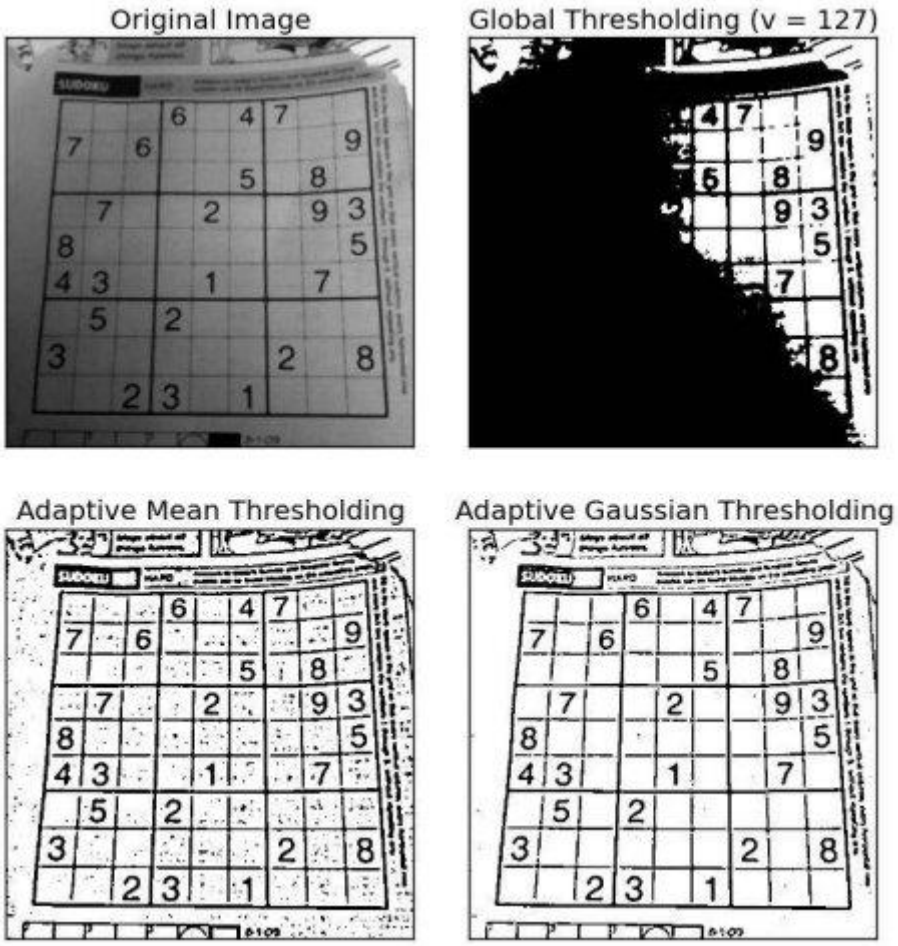
Histogramm näitab, kuidas erinevad helduse astmed asuvad pildil. Pildi kontrast on mõõdetud heleduse vahemiku järgi. Histogrammilt näeb ära, kui ei ole ära kasutatud iga olemasolevat hallitaset. Kui algse pildi histogramm on olemas, saab kasutada histogrammi normaliseerimise algoritmi. See on vajalik selleks, et pilti muuta selgemaks ja kontrastsemaks. [1]

2.1.2 Punktoperatsioonid

Põhilised pilditöötlusoperatsioonid kasutavad algoritme, mis asendavad iga vana piksli väärtuse uuega. Näiteks pildi heleduse muutmiseks on vaja pildi maatriksit korrutada skalaariga. [1]

Lävisegmentimine (*thresholding*) selekteerib pikslid, millel on kindel väärtus või väärtuste vahemik. Seda saab kasutada, kui on teada, millise heledusega on pildil otsitavad objektid. Optimaalsel lävisegmentimisel kasutatakse normaliseeritud

histogrammi, et leida kõige optimaalsem lävi, millega on eristatavad taust ja objektid (vt Joonis 1). [1]



Joonis 1: Lävisegmenteerimine[1]

2.1.3 Grupioperatsioonid

Grupioperatsioonide abil arvutatakse pikslile uus väärtus seda ümbritsevate pikslite järgi. See tähendab seda, et piltide ääri ei saa ümber arvutada, kuna äärmiseid piksleid ei ümbritse igast suunast pikslid. Seetõttu on ümberarvutatud pilt väiksem, kui originaal. [1]

Keskmistamise operaator (*averaging operator*) näiteks korrutab kõik pikslid 3x3 ruudus läbi 1/9. Saadud väärtused liidetakse ja saadakse uus piksli väärtus. Korrutus toimub 1/9, sest üheksa valge piksli keskmine väärtus ei saa olla rohkem, kui valge väärtus. Antud

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

operaatorit kasutatakse selleks, et pildil vähendada ebavajalikke detaile ja tuua paremini esile suuremaid objekte. [1]

Operaatori tuum võib olla ka suurem, kui 3x3. Kuna need on tavaliselt keskendunud huvipunkti ümber, et arvutada uut väärtust uuritavale punktile, siis on tuuma suurus tavaliselt paaritu arv. Samas ei saa ka tuuma suurus olla liiga suur, sest vastasel juhul kasvab arvutamisele kuuluv aeg. Näiteks 9x9 suurusega operaatori puhul peab tegema 9 korda rohkem arvutusi, kui 3x3 puhul. [1]

Gaussi keskmistamise operaator on optimaalne pildi silumisoperaator. Gaussi operaatori tuum omab väärtuseid, mis on sätitud Gaussi suhete poolt. Gaussi kõver määrab ära, et keskmise piksli väärtuse arvutamisel on keskel asuvate pikslite kaal suurem, kui ääres asuvate pikslite kaal (vt Joonis 2). Selle meetodiga välditakse keskpunktist kaugel asuva radikaalselt erineva piksli mõju arvutatavale uuele pikslile. [1]

0.002	0.013	0.022	0.013	0.002
0.013	0.060	0.098	0.060	0.013
0.022	0.098	0.162	0.098	0.022
0.013	0.060	0.098	0.060	0.013
0.002	0.013	0.022	0.013	0.002

Joonis 2: Gaussi keskmistamise operaatori 5x5 tuum ($\sigma = 10$) [1]

2.2 Ääretuvastus

Ääretuvastus võimaldab keskenduda vaid nendele piirjoontele, mida päriselt leida tahetakse. Ääretuvastuse tulemiks on objektide kontuurid. Ääretuvastus leiab pildil piirkonnad, kus pildil asub kontrast. Objekti äär pildil on sisuliselt piirkond, kus on astmeline intensiivsuse taseme muutus. [1]

2.2.1 Sobeli ääretuvastus

Sobeli ääretuvastus on gradiendipõhine meetod. See arvutab esimese tuletise eraldi x- ja y-telgedest. Meetodit, kus mõlema telje tuletised arvesse võetakse, nimetatakse Laplace'i

[1] Allikas: Feature Extraction and Image Processing for Computer Vision (lk 106)

meetodiks. Tuletised on ainult ligikaudsed väärtused, kuna pildid ei ole pidevad mööda telge. Antud meetodi puhul kasutatakse kahte tuuma ahendamiseks (vt Joonis 3). [3]

-1	0	1
-2	0	2
-1	0	1

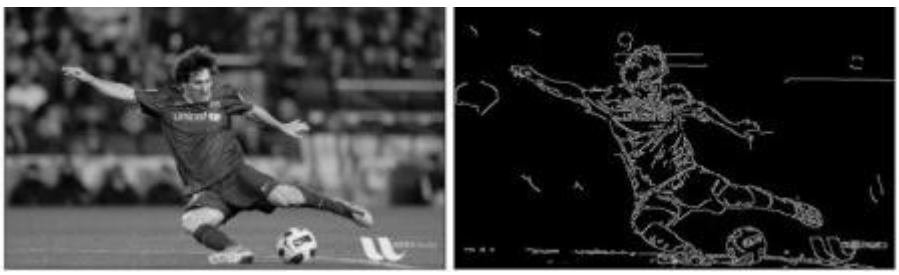
-1	-2	-1
0	0	0
-1	-2	-1

Joonis 3: Horisontaal- ja vertikaaltuumad Sobeli ääretuvastusoperatsioonis[1]

2.2.2 Canny ääretuvastus

Piksel asub äärel, kui selle intensiivsus muutub järsult võrreldes oma naabritega. Äär on ise lineaarne kuju, mida mööda muutus on maksimaalne. Canny tuvastusega leiab peenikesed, selged servad pildilt (vt Joonis 4). [2]

Canny ääretuvastuse eel on vajalik pildi hägustamine müra eemaldamiseks, milleks kasutatakse Gaussi hägustamist. Seejärel kasutatakse pildi töötlemiseks Sobeli ääretuvastust, kus olime arvutanud ligikaudse esimese tuletise nii x-, kui ka y-teljele. Nende andmete põhjal saab leida ääre gradiendi ja suuna igale pikslile. Seejärel eemaldatakse pikslid, mis ei kuulu serva juurde. Ebavajaliku info eemaldamiseks kasutatakse ka lävisegmentimist. [4]



Joonis 4: Canny ääretuvastus[2]

2.3 Nurgatuvastus

Ääretuvastusega säilivad omadused, mis on inimsilmaga tuvastatavad, kuid piisav ka kõveruste tuvastamisest. Kõverus on ääre suuna muutumise kiirus. Punktid, kus ääre

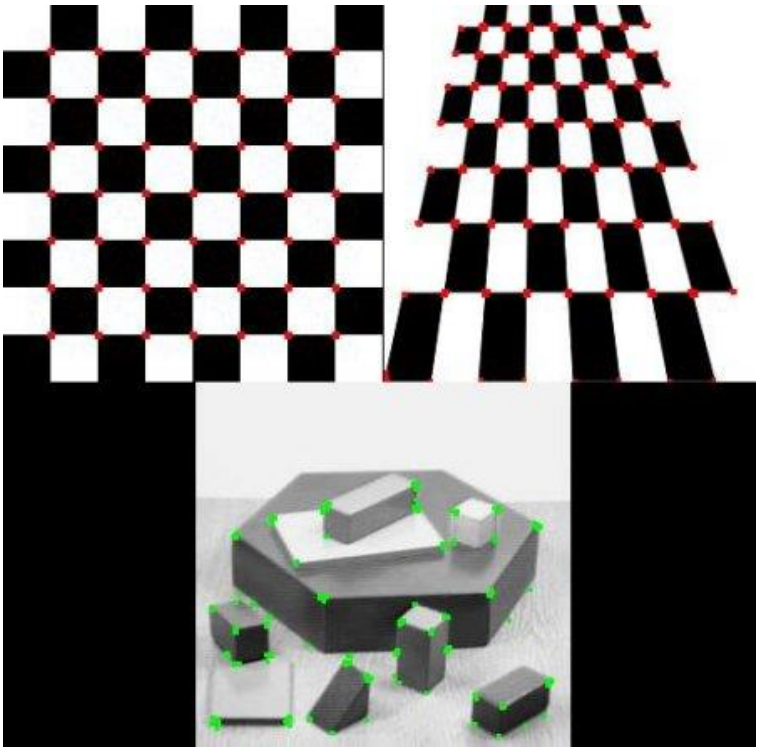
[1] http://aishack.in/tutorials/sobel-laplacian-edge-detectors/

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

kõverus muutub järsult, on nurgad. Nurgad on väga kasulikud omaduse kirjeldamisel ja omavahel sobitamisel, kuna need esindavad tähtsat informatsiooni vähendatud kujul. [1]

2.3.1 Harrise nurgatuvastus

Harrise nurgatuvastus põhineb Moravec'i nurgatuvastusel. See operaator arvutab keskmise muutuse kujutise intensiivsuses, kui akent nihutatakse erinevas suunas (vt Joonis 5). Kõveruse muutus on miinimum väärtus, mis saadakse, kui pilti liigutatakse neljas erinevas suunas. Näiteks, kui vaadeldakse serva, siis servaga pikuti on kõveruse muutus minimaalne ning servaga risti on kõveruse muutus maksimaalne. Kuid kui vaadeldakse nurka, siis peaks olema kõveruse muutus igas suunas suur. [1]



Joonis 5: Harrise nurgatuvastus[1]

[1] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html

2.4 Karakteristikute tuvastus

Karakteristikute põhjal objekti tuvastuse korral kasutatakse malli järgi sobitamist. Selleks, et tuvastada sobivust, leitakse esiteks huvipunktid, mis asuvad nurkades ja muudes sellistes kohtades. Huvipunkti tuvastaja kõige väärtuslikum omadus on selle korratavus, kuna sõltumata vaatenurgast erinevates tingimustes, tuvastab see samad huvipunktid. Iga huvipunkti ümbrus on esindatud omadusvektorina. Antud omadusvektor peab olema eristuv ja robustne, et eristuda hoolimata mürast ja geomeetrilistest deformatsioonidest. Saadud omadusvektoreid võrreldakse andmekogus asuvate piltide vahel. Väikesemõõduliste vektoritega on sobitamine kiire, kuid kaotatakse täpsuses. Täpsema tuvastuse jaoks on vaja kasutada suuremõõtmelisi omadusvektoreid. [7]

2.4.1 Haari teisendus

Haari teisendust ja Haari laineteisendust kasutatakse Haari kaskaadis klassifitseeria treenimiseks. Haari teisendust kasutatakse andmete kokkupressimiseks, eraldades ebaolulise ja säilitades olulise. See on madalpääsfilter, kuna see ühtlustab väärtusi maatriksis. Selle vastandiks on Haari laineteisendusfilter, mis on kõrgpääsfilter. Kui kaks väärtust on sarnased, siis antud filter tagastab 0, kuid kui väärtused on erinevad, siis tagastatakse ühe numbri kaalutud väärtus. Kahemõõtmeline Haari laineteisendus sobib suurepäraselt pilditihenduseks (vt Joonis 6). [8]

Joonis 6: Kahemõõtmeline Haari laineteisendus[1]

[1] http://www.whydomath.org/node/wavlets/blowup/haarwt.html

3 Karakteristikute tuvastamise realisatsioon

Antud jaotises käsitletakse karakteristikute tuvastamise realisatsiooni keeltes C++ ja Python.

3.1 Tehnoloogia valik

Antud ülesande lahendamisel valis autor Pythoni[1] programmeerimiskeele selle lihtsuse tõttu. C++[2] programmeerimiskeele kasutamine on vajalik, et kasutada Point Cloud Library (PCL)[3] teeki, kuna antud sensorist saadakse andmed Point Cloud Data (PCD)[4] formaadis. Pythonis on võimalik kasutada pilditöötluseks ja raalnägemise implementeerimiseks OpenCV[5] teeki, mis sisaldab sadu raalnägemise algoritme. Pilditöötluseks on võimalik Pythonis kasutada ka Scikit-Image[6] teeki. Masinõppe realiseerimiseks on kasutatud OpenCV teegi juure kuuluvat Haar klassifitseerijat.

3.2 Programmi ülesehitus

Antud lahendus on jaotatud kaheks. Esimese osa ülesandeks on PCD failist eraldada tuvastatud karakteristikud. Alternatiivseks sisendiks antud osal on ka BMP fail juhul, kui andmed on saadaval ka pildi kujul. Esimese osa väljundeid kasutatakse Haar klassifitseerija õpetamiseks. Teine osa kasutab õpetatud klassifitseerijaid, et eristada sisendiks antud mobiiltelefoni korpuse pildil sellele kuuluvad karakteristikud kriimudest, täketest ja mõradest.

[1] https://www.python.org/

[2] http://www.cplusplus.com/

[3] http://pointclouds.org/

[4] http://pointclouds.org/documentation/tutorials/pcd_file_format.php

[5] https://opencv.org/

[6] http://scikit-image.org/

Esimene programm käivitab järjest kolm erinevat programmi, mida saab ka eraldiseisvalt käivitada. Esimene neist on C++ programmeerimiskeeles kirjutatud Windows *executable*, mille ülesandeks on sensorilt saadud binaarandmed töödelda loetavaks tekstifailiks. Järgnevad kaks programmi on Python skripti kujul, millest esimene teisendab teksti kujul andmed pildifailiks ja teine tuvastab pildilt leitavad karakteristikud, mis salvestatakse pildi kujul. Põhiprogramm, mis käivitab need kolm programmi järjest, on kirjutatud Pythonis. Käivitamiseks on programmile ette vaja anda PCD faili asukoht argumendina või pildi olemasolu korral BMP faili asukoht.

Teine osa on kirjutatud Python programmeerimiskeeles. See kasutab OpenCV teegi juurde kuuluvat Haar klassifitseerijat, et tuvastada eraldada kaamera, kõlar või sõrmejäljelugeja leitud karakteristikute seast.

3.3 Sensorandmete töötlus

Sensorist saadud PCD fail sisaldab endas punktide kirjeldust, defineerides punkti X, Y ja Z koordinaadid ning RGB värvikoodi. Kuna failis on andmed binaarkujul, siis on vajalik nende andmete lugemiseks kasutada PCL teeki. Kui andmed oleksid ASCII formaadis, saaks andmeid lugeda ka tavalise tekstifailina.

Selleks, et karakteristikuid tuvastada, viiakse need andmed pildi kujule. Pildi kujul andmete töötlemine on palju kiirem, kui PCD failis asuva 3D andmete töötlemine. Sensorandmetest pildi saamiseks viiakse PCD fail tekstifaili kujule, kus iga rida kirjeldab ühe punkti XYZ koordinaate. RGB andmed ei ole antud probleemi puhul olulised ja saab andmete vähendamise eesmärgil välja jätta, kuna kriimude, täkete jms korral on mobiili pinna sügavus selles kohas muutunud.

Realiseerides seda ülesannet C++ programmeerimiskeeles, tekkis probleem andmete suurusega. Algandmete failide suurused on 200 MB-st kuni 1 GB-ni. Esialgne programm valmis 32-bitisena. 32-bitise programmiga aga ei saanud töödelda andmeid, mis olid suuremad kui 700 MB. Programmile eraldatud mälu sai faili lugemisel otsa ning programmi töö katkes. Antud probleemi korral tuli viia kogu programm üle 64-bitiseks. Selleks tuli ka installeerida kasutatava PCL teegi 64-bitine versioon. 64-bitise programmiga lahenes mälu probleem.

Lisaks mälukasutuse probleemile tekkis ka probleem, et PCD faili lugemine ja töötlemine võttis aega minuteid, mitte sekundeid, nagu oli algne eesmärk. Seda probleemi antud ülesandes lahendada ei saa, kuna PCL teegil puudub faili vooglugemise võimalus binaarkujul faili korral. Antud probleem on aga lahendatav juhul, kui sensorandmete töötlemine pildiks toimub juba algandmete saamisel sensorist.

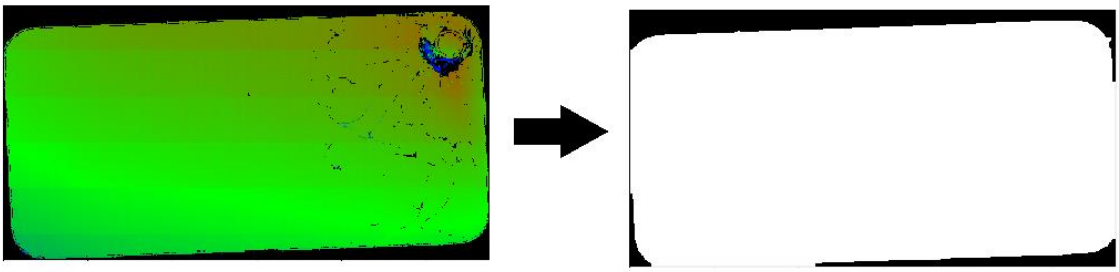
3.4 Koordinaatandmete teisendamine pildiks

Selleks, et kasutada andmete töötlemisel OpenCV teeki, tuleb kõigepealt tekstikujul olevad andmed teisendada pildiks. Esimeses etapis tuvastatakse andmete põhjal loodava pildifaili suurus X ja Y väärtuste põhjal. Leitakse ka Z väärtuse miinimum ja maksimum, et vastavalt nendele määrata Z koordinaadile RGB väärtus.

Edasi toimub pildi koostamine kasutades Pillow[1] teeki, mis on Pythoni pildindusteek. Antud teek võimaldab luua pildi, määrates igale pikslile tema vastava värvi. Piksli asukohana saab kasutada X ja Y koordinaate otse failist. Nende koordinaatide asukohale määran piksli värvi RGB kujul, Z koordinaadi ning esimeses punktis leitud Z koordinaadi miinimumi ja maksimumi väärtuse järgi.

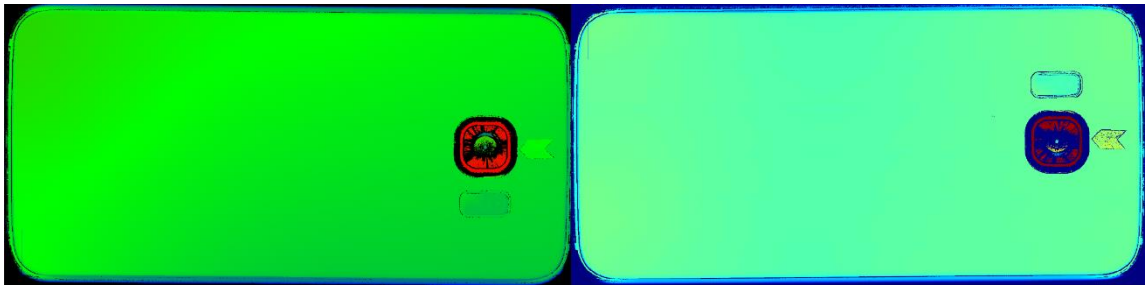
Peale pildi koostamist vähendatakse selle suurust, eemaldades ebavajalik taust. Objekti piiride leidmiseks kasutatakse esiteks Gaussi hägustamist, et eemaldada ebavajalikud detailid. Morfoloogilist sulgemisalgoritmi koos lävisegmenteerimisega kasutatakse, et luua objektist mask, mille abil saab juhtida pildi osade säilitamist või kõrvaldamist (vt Joonis 4). Saadud mask piiratakse minimaalse ristkülikuga, mille koordinaate kasutatakse objekti välja lõikamiseks pildilt. Tulemuseks on minimaalse suurusega pilt mobiilist, mille värvid viitavad sügavusele.

[1] https://pillow.readthedocs.io/en/5.0.0/



Joonis 4: Mobiili objektist saadud mask

Antud programmid, mis töötlesid PCD failist pildi, töötavad väga aeglaselt. Selle probleemi lahendamiseks sai palutud ka sensorandmeid kohesel pildi kujul. Seejärel võimaldati ligipääs andmetele, mis olid nii PCD kui ka BMP (*bitmap image file*) formaadis. Piltide õigsuse testimiseks kasutati pildiga kaasas olevat PCD faili ning kasutati eelnevalt loodud programmi, et genereerida see pildi kujule. Tulemused olid rahuldavad ning sai järeldada, et mõlemal juhul on genereeritud pildid peaaegu identsed (vt Joonis 7).



Joonis 7: PCD failist genereeritud pilt ja eelnevalt genereeritud pilt

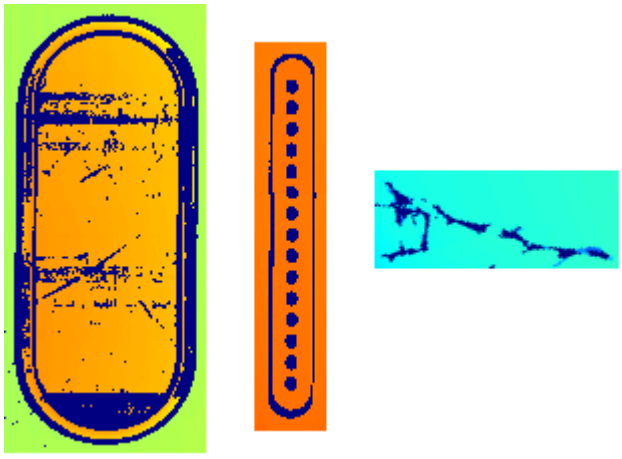
3.5 Karakteristikute tuvastamine pildilt

Karakteristikute tuvastamine pildil algab ääretuvastusest. Ääretuvastusega leian pildil jooned, kus pildi värv muutub hüppeliselt. Antud programmis kasutatakse selleks Canny ääretuvastusoperatsiooni (vt Joonis 8).

Joonis 8: Canny ääretuvastus

Selleks, et ääre kontuurid oleksid omavahel paremini seotud, kasutatakse erinevaid kombinatsioone morfoloogilistest transformatsioonidest. Saadud tulemusele rakendatakse OpenCV funktsiooni *findContours*, et leida kõik kontuurid. Leitud kontuuridest jäetakse välja alad, mille külje pikkus on väiksem kui 5 pikslit, et vähendada müra. Kontuurid piiratakse minimaalse ristkülikuga ning pilti pööratakse, et x-koordinaadid oleks horisontaalil ning y-koordinaadid asuksid vertikaalil.

Esialgsest pildist lõigatakse välja leitud kontuure piiravad ristkülikud. Saadud pildid kujutavad tuvastatud karakteristikuid. Nendeks võivad olla kaamera, kõlar, sõrmejäljelugeja või ka näiteks täke, kriim ja mõra (vt Joonis 9).

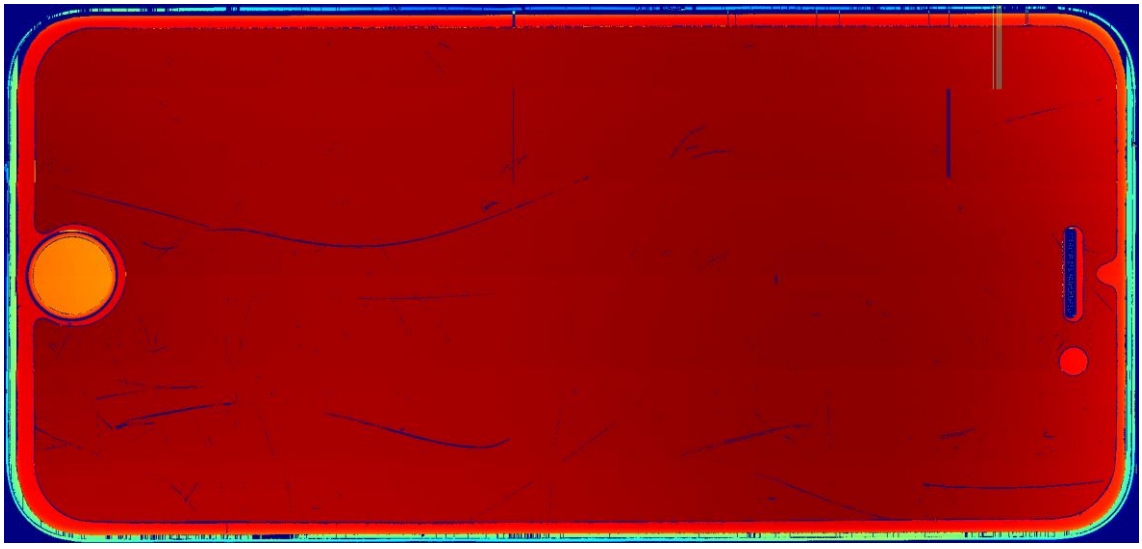


Joonis 9: Tuvastatud sõrmejäljelugeja, kõlar ja mõra

3.6 Vigade tuvastamine telefonil Haar klassifitseerija abil

Vigade tuvastamiseks haar klassifitseerija abil tuleb esmalt koostada vajalikud andmehulgad. Positiivsetes andmehulkades on nii kaamera, kõlar kui ka sõrmejäljelugeja. Negatiivses andmehulgas asuvad kriimustused, mõrad jms, mis ei tohiks asuda terve mobiiltelefoni korpusel.

Antud andmeid kasutatakse, et treenida uusi Haar klassifitseerijaid. Antud juhul on tegemist karakteristikutega, mille kohta ei eksisteeri olemasolevaid klassifitseerijaid. Selle ülesande teeb raskeks väike andmete hulk. Välja valitud karakteristikud pärinevad kõik sarnaste mobiiltelefonide korpustelt, kuna andmeid on vähe ning eesmärk on treenida klassifitseerijat võimalikult täpselt.



Joonis 10: Mobiili ekraan kriibitud kaitsekilega

Olemasolevate andmete hulgast eemaldatakse testandmed, mis annavad vääri tulemusi. Nendeks on mobiilikorpused, mille peal on kaitsekiled, kleepsud ja sõrmejäljed (vt Joonis 10). Enne mobiiltelefoni skaneerimist on vajalik korpuselt eemaldada kõik segavad faktorid, et skaneerimine oleks võimalikult täpne. Kindlasti on vajalik mobiiltelefoni puhastamine lapiga, et eemaldada sõrmejäljed ning tolm. Andmehulkade sisse valisin ainult korras olekuga kaamera, kõlari ja sõrmejäljelugeja näited (vt Joonis 11). Tausta andmehulgana kasutan tuvastatud kriime ja mõrasid.

```
bg/
    00a1113f-9554-4b73-b6ca-029e178d6587.bmp
    00b8a24b-ee69-4c5f-90a7-eff9abc5a810.bmp
    ...
camera/
    0b6745f7-4352-4a80-8774-258d332dbe46.bmp
    208181fc-c8a7-4169-95ab-785e556f1dca.bmp
    ..
fingerprint_sensor/
    0c2830f4-cb77-4c08-a5b1-84b865405202.bmp
    6e23b877-1511-42d2-ab36-a95e9cb26ba2.bmp
    ..
speaker/
    1c63e986-7cdb-497a-8591-0eb36c599743.bmp
    4aa0d900-9e43-4130-bfa0-5af45a00a315.bmp
    ..
```

Joonis 11: Andmehulkade kaustad

Antud andmehulkade failide asukohad koondan kokku vastavalt *bg.dat*, *camera.dat*, *fingerprint_sensor.dat* ja *speaker.dat* tekstifailidesse. Edaspidi kasutan OpenCV teegis olevaid Haar tuvastaja treenimise meetodeid (vt Joonis 12).

```
opencv_createsamples -info camera.dat -vec camera.vec -w 220 -h 220
-num 27

opencv_traincascade -data data -vec camera.vec -bg bg.dat -w 220 -h
220 -numPos 20 -numNeg 1000 -featureType HAAR -numStages 12
```

Joonis 12: Käsud klassifitseerija treenimiseks

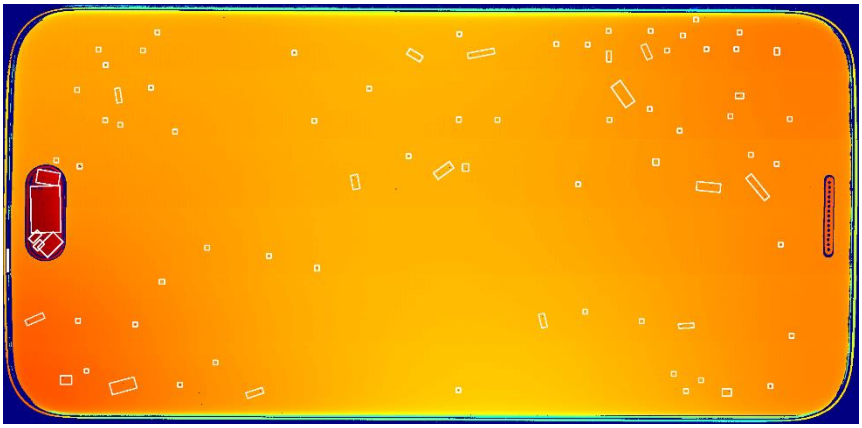
Esimene käsk loob igast *camera.dat* failis kirjeldatud pildist ühe *camera.vec* faili, kus kõik 27 kaamera pilti on ära kirjeldatud suuruses 220x220.

Teise käsu ülesandeks on klassifitseerija treenimine. Selleks antakse ette negatiivseid pilte kirjeldav *bg.txt*, positiivsete näidetega *camera.vec*, treenimiseks kasutatavate piltide hulk ja suurus ning klassifitseerija tüüp ja etappide arv.

Antud käsud viiakse läbi iga positiivse andmekogu peal, et saada vastavalt kaamera, kõlari ja sõrmejäljelugeja klassifitseerijad.

Saadud klassifitseerijaid kasutatakse järgnevas programmis, et eraldada mobiili korpusel olevad kaamera, kõlar ja sõrmejäljelugeja kriimudest ja mõradest. Selleks eraldatakse uuesti mobiilikorpusel asuvad karakteristikud ning kasutatakse klassifitseerijaid, et tuvastada, kas tegemist on kaamera, kõlari või sõrmejäljelugejaga. Kui antud karakteristik

on tuvastatud, siis see eemaldatakse karakteristikute loetelust. Järele jäänud karakteristikud peaksid olema kriimud, mõrad ja täkked (vt Joonis 13).



Joonis 13: Mobiili korpuselt tuvastatud täkked ja kriimud

Tulemuse täpsuse kontrollimiseks prooviti antud programmi 6 mobiiltelefoni pildi korpuse peal (vt Lisa 1). Antud tulemustest saab järeldada, et mõnel juhul töötab klassifitseerija abil kaamera, sõrmejäljelugeja ning kõlari eemaldamine vigade hulgast, kuid on ka juhte, kus ei suudeta antud karakteristikuid tuvastada. See olukord võib olla põhjustatud väikestest positiivsete andmehulkadest. Vaadatud tulemustest võib järeldada, et ka mobiili korpusele võiks luua treenitud klassifitseerija või eraldada tuvastatud korpused vigade hulgast mingil muul viisil.

4 Kokkuvõte

Mobiiltelefonide turul on oluline koht ka nende järelmüügil. Selleks, et hinnata seadme väärtust järelturul, on eelnevalt vaja hinnata seadme seisukorda. Seadme seisukorra juurde kuulub ka korpuselt vigade tuvastamine. Käesoleva töö eesmärgiks oli luua programm, mis tuvastab 3D FocalSpec sensoriga skaneeritud mobiiltelefonide korpustel ja ekraanil asuvaid kriime, täkkeid ja mõrasid.

Eesmärgi saavutamiseks uuriti pilditöötluse aluseid ja erinevaid algoritme karakteristikute välja toomiseks pildil. Uuriti lävisegmenteerimist, Gaussi hägustamist, Canny ääretuvastust ja mitmeid teisi algoritme.

Probleemi lahendus algas sensorandmete töötlusest. Sensorandmed olid defineeritud punktpilve kujul PCD (*Point Cloud Data*) formaadis. Antud andmete töötluseks loodi programm C++ programmeerimiskeeles. Selle programmi sisendiks olid sensorandmed ning väljundiks tekstifail, mis kirjeldas sügavusandmeid igal koordinaadil. Saadud andmed töödeldi Python skripti abil pildi kujule, kus erinevad värvid kirjeldasid erinevat sügavust. Antud programmi loomine oli keeruline mälukasutuse aspektist. Iga PCD fail on ligi 1 GB suurune ning kahjuks ei leitud käesoleva töö raames lahendust programmi töökiiruse vähendamiseks alla 15 minuti. Küll aga osutus võimalikuks andmete konverteerimine pIldi kujule juba sensorist andmete lugemisel, mobiiltelefonide skaneerimise ajal.

Selleks et tuvastada vigu telefoni korpusel, oli oluline kasutada erinevaid pilditöötluse võtteid, et korpuse punktide sügavuste erinevused paremini esile tuua. Esile kerkinud karakteristikud salvestati pildikujul ning jaotati erinevatesse kategooriatesse, et neid kasutada Haar klassifitseerijate treenimisel.

Haar klassifitseerijate treenimiseks valisin kaamera, kõlari ja sõrmejäljelugeja andmehulgad. Antud karakteristikute kohta oli treenimiseks sobivaid näiteid alla saja. Paremaks klassifitseerija treenimiseks oleks vaja neid andmeid tuhandetes. Treenitud klassifitseerijaid kasutasin, et eristada tuvastatud karakteristikutest eelpool mainitud

karakteristikuid. Saadud tulemus sisaldas tuvastatud karakteristikuid, milleks olid kriimud, mõrad või täkked.

Töö esialgne eesmärk oli lisaks kriimude ja täkete tuvastamisele need ka klassifitseerida suuruse/sügavuse järgi. Antud eesmärgist loobuti, kuna selleks oleks olnud vaja vahet teha kriimul, täkkel või mõral. Haar klassifitseerija treenimisel oli oluline kasutada väga kindla kujuga karakteristikuid, et klassifitseerija tuvastaks neid võimalikult täpselt. Kriimude, täkete ning mõrade puhul on tegemist karakteristikutega, millel puudub kindel kuju.

Käesoleva töö peamiseks eesmärgiks oli eristada korpusel tüüpiliselt esinevaid karakteristikuid vigadest, mis on tekkinud mobiiltelefoni korpusele kasutamise tagajärjel. Antud eesmärk sai täidetud ning programm on võimeline tuvastama vead ning eristama neid kaamerast, kõlarist või sõrmejälje lugejast.

Kasutatud kirjandus

[1] Nixon, Mark S., Aguado, Alberto S. Feature Extraction & Image Processing for Computer Vision, Third edition. London: Accademic Press, 2012. [*Online*] Ebook Central (03.12.2017)

[2] Aichert, A. Feature extraction techniques. – *Camo Medical Seminar*, vol. 13, 352-333. [*Online*] Semantic Scholar (06.12.2017)

[3] The Sobel and Laplacian Edge Detectors.
[WWW] http://aishack.in/tutorials/sobel-laplacian-edge-detectors/ (08.12.2017)

[4] Canny Edge Detection.
[WWW] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html (09.12.2017)

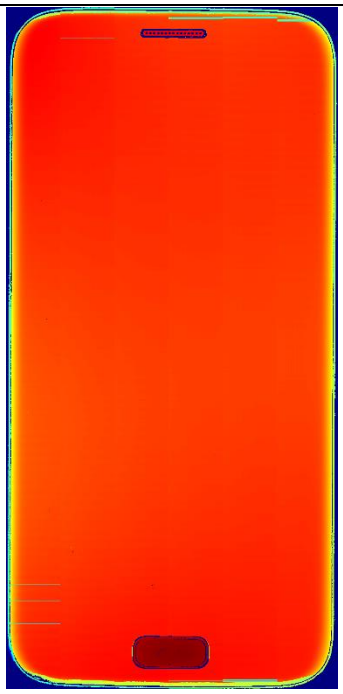
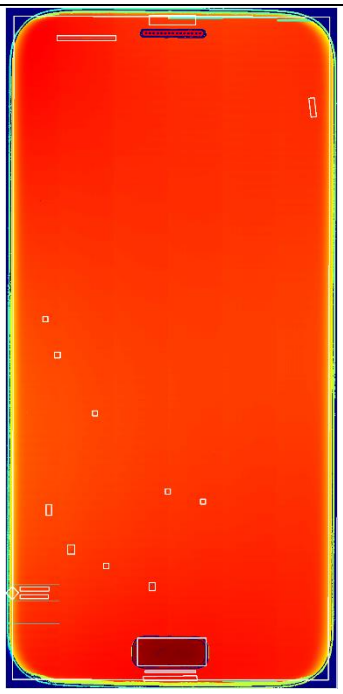
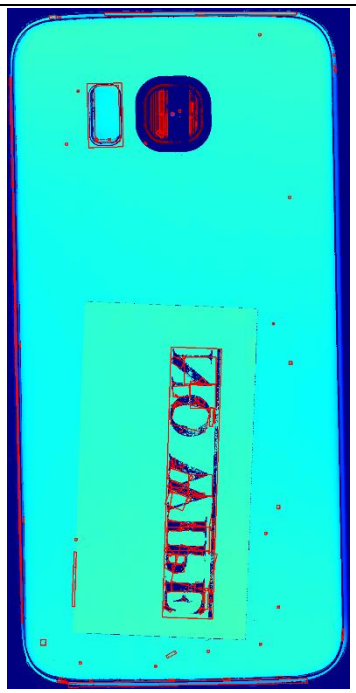
[5] Harris Corner Detection. [WWW] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html (11.12.2017)

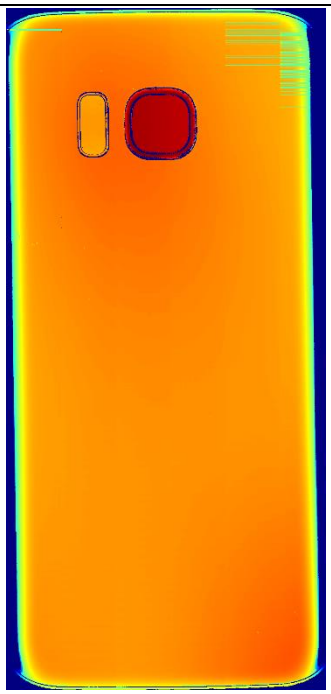
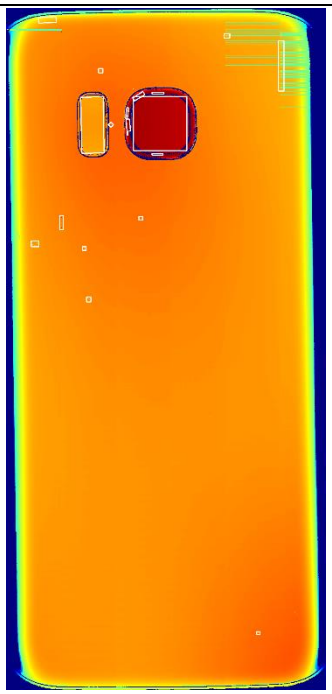
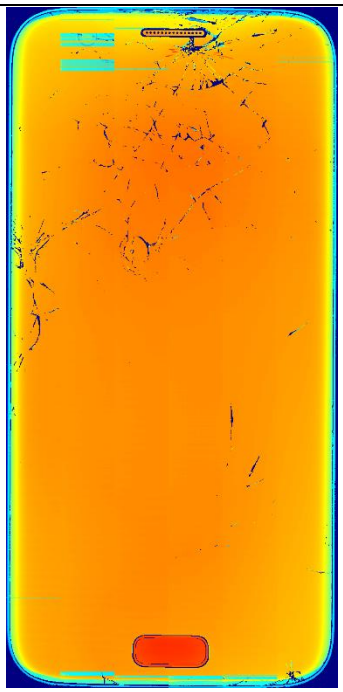
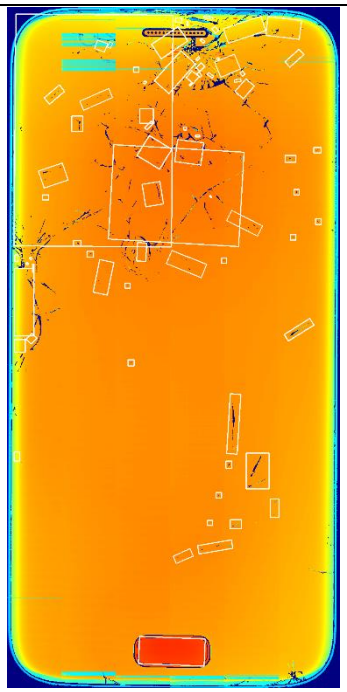
[6] He, X.C., Yung, N.H.C., Corner detector based on global and local curvature properties. – *Optical Engineering*, 2008, 47(5), 057008. [*Online*] Spie Digital Library (11.12.2017)

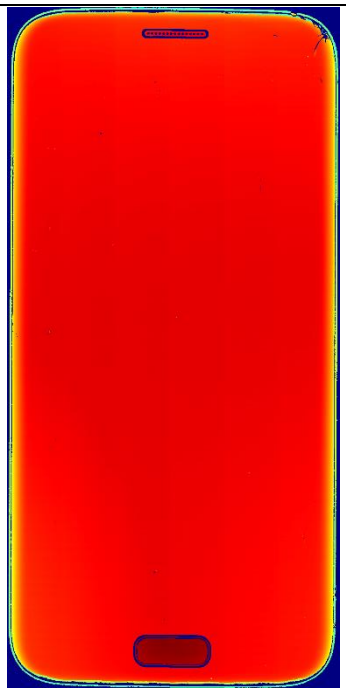
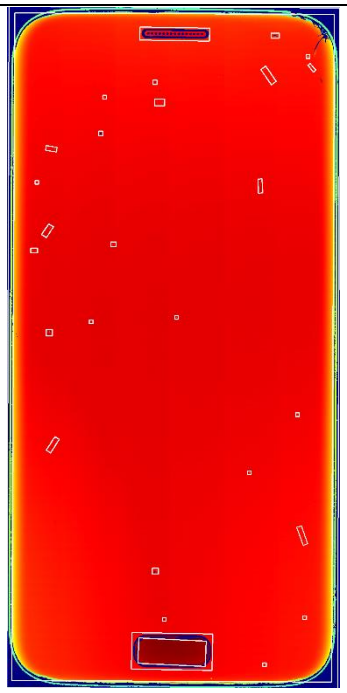
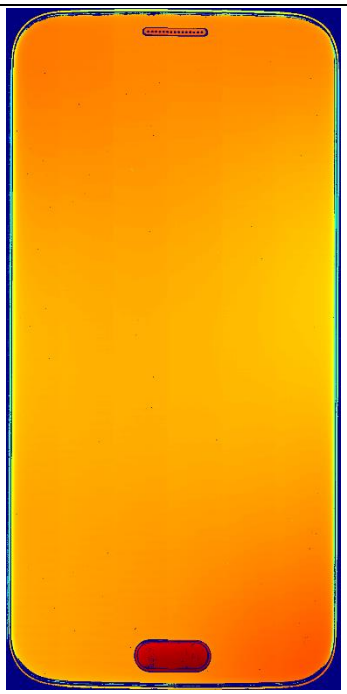
[7] Bay, H., Ess, A., Tuytelaars, T., Gool, V.L., Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding.* 2008, 110(3), 346-359. [*Online*] ScienceDirect (20.05.2018)

[8] Haar Wavelet Transformation. [WWW]
http://www.whydomath.org/node/wavlets/hwt.html (23.05.2018)

Lisa 1 – Vigade tuvastuste tulemus ning analüüs

Esialgne pilt	Tuvastatud vead	Tähelepanekud
		Tuvastati enamus kriime ja täkkeid. Ära on märgitud ka kogu korpus, millest võib järeldada, et on vaja õpetada ka korpuse ära tundmiseks klassifitseerija.
		Programm on tuvastanud ära teksti ja täkked, kuid ka sõrmejäljelugeja. Klassifitseerijale õpetati ainult ilma kleepsude ja kriimudeta sõrmejäljelugejat ning antud mobiili puhul fikseeriti see veana.

		Tuvastatud on täkked ja kriimud, kuid samuti on tuvastatud ka kaamera ja sõrmejäljelugeja sisemine osa. Järelikult on vaja õpetada klassifitseerijale eristama ka nende komponentide sisemist osa.
		Antud mobiiltelefoni ekraan on nii katki, et tihedate mõrade seast on keeruline tuvastada iga üksikut mõra.

		Tuvastatud vigade hulgas on ka sõrmejäljelugeja ning kõlar, millel pole midagi viga. Antud juhul ei ole klassifitseerija suutnud tuvastada, et tegemist on kõlari ja sõrmejäljelugejaga. Selline ebatäpsus võib tuleneda vähestest positiivsete näidete hulgast.
		Antud juhul on vigade tuvastamine õnnestunud kõige paremini. Kõik väiksed täkked ja kriimud on välja toodud. Samuti on näha, et klassifitseerija abil on korrektsel ära tuvastatud kõlar ja sõrmejäljelugeja ning need vigade seast välja jäätud.