TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jaanus Keller
155243IAPB

TARKVARA LOOMINE MAA KAUGSEIREKS TTÜ100 SATELLIIDIPROGRAMMIS

bakalaureusetöö

Juhendaja: Evelin Halling, 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: Jaanus Keller

20. mai 2018

Annotatsioon

Töö eesmärk on teha testkeskkonnas pilte TTÜ100 satelliidi kaamerasse mineva sensoriga ning luua missioonijuhtimistarkvara täiendus, mis korraldab kaugseire andmetega ümberkäimist.

Esile on toodud käsud ja tarkvara, millega saab testkeskkonnas pilte teha. Veel on välja toodud meta andmed, mida on vaja, et toores pilt konverteerida vaadatavasse formaati.

Missioonijuhtimistarkvara täienduse osas on realiseeritud kaugseire andmete salvestamine. Välja on toodud ka piltide *georeferencing*'i lahendusi ning selle automatiseerimisel tekkinud probleemid. Kaugseire andmete kasutajale näitamiseks on välja toodud nõuded.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 6 peatükki, 25 joonist, 3 tabelit.

Abstract

Remote sensing software development for TTÜ100 satellite program

The goal of the thesis is to create software that handles remote sensing data. The main goal consists of two subgoals. The first one is capturing images using the same sensor that will be used for the TTÜ100 satellite. The second subgoal is to develop mission control system software extension that handles remote sensing data.

The thesis highlights the commands and software required to take pictures using the development environment as well as the metadata that can be used to convert raw image to a more usable format.

Remote sensing data storing solution has been developed as part of the software extension. Additionally, the thesis explains the georeferencing process and issues that arose when trying to automate the process. Lastly, the thesis introduces requirements for displaying remote sensing data to users.

The thesis is in Estonian and contains 28 pages of text, 6 chapters, 25 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	*application programming interface*
DMAI	*DaVinci multimedia application interface*
DVSDK	*digital video software development kit*
EPSG	*European Petrolium Survey Group*
GIS	*geographic information system*
MVC	*model-view-controller*, mudel-vaade-kontroller
NIR	*Near infrared*, lähi-infrapuna spekter
REST	*representational state transfer*
RGB	*Red green blue*, värvimassiivis punane, roheline ja sinine värv
SDK	*software development kit*
SPA	*singel-page-application*,
TCP	*transmission control protocol*
TI	Texas Instruments
UTM	*Universal Transverse Mercator*
V4L	*video 4 linux*
V4L2	*video 4 linux 2*

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

TTÜ100 satelliidi eesmärk on maa seire läbiviimine. Põhilised vahendid selleks on NIR kaamera lähi-infrapuna spektris pildistamiseks ja RGB kaamera inimsilmale nähtavas valgusspektris piltide tegemiseks. Lisaks seirele viiakse satelliidil läbi katseid arvutustehnika tõrkekindlusest ning optilise side kasutamisest. Maajaamaga suhtlemiseks kasutab satelliit madala sagedusega ühendust või eksperimentaalsemat kõrge sagedusega ühendust. [1]

Töö eesmärk on teha pilte testkaameraga ning välja mõelda ja realiseerida TTÜ satelliidiprojekti kuupsatelliidi TTÜ100 missioonijuhtimistarkvara täiendus, mis tegeleb kaugseire andmete salvestamise, analüüsimise ja kasutajale väljanäitamisega. Seejuures on arvestatud nõuetega missioonijuhtimistarkvarale ja juba kasutusel olevate tehnoloogiatega.

Peatükis 2 esitatakse loodavale süsteemile nõuded, tutvustatakse olemasolevat missioonijuhtimistarkvara ja kirjeldatakse kasutatavaid tarkvaraarenduse metoodikat. Välja on veel toodud kasutatavad programmid, teegid, raamistikud ja loodava süsteemi arhitektuur.

Peatükis 3 on kirjeldatud satelliidi RGB kaamera toorest formaati ning toodud välja vajalikud meta andmed, mida maajaam ootab satelliidilt. Meta andmete vajadus on ka põhjendatud ning võimalusel on välja toodud alternatiivid info jaoks.

Peatükk 4 kirjeldab testkaameraga pildi tegemist. Välja on toodud kasutatav tarkvara ning riistvara ja kirjeldatud nende ülesseadmine. Peatükis on ära mainitud pildistamisel tekkinud probleemid ja probleemide lahendused.

Peatükk 5 analüüsitud maajaama tegevusi piltidega, mis on satelliidilt alla saadetud. Seal hulgas on piltide salvestamine, analüüsimine ning kasutajale näitamine.

2 Nõuded loodavale süsteemile

Töös on käsitletud kaameraga pildi tegemist, kaugseire tulemite (põhiliselt piltide) analüüsi, salvestamist ning kasutajale näitamist. Nõuded on välja toodud tiimiga ning lõppkasutajaga intervjuude käigus.

Nõuded kaameraga pilditegemise kohta:

N1 Kaameraga peab kätte saama pildi Bayer'i mustri.

N2 Kaamera parameetreid peab saama muuta.

N3 Kaamera peab pilte tegema sensori aktiivse osa täisresolutsiooniga (2592x1944).

Nõuded kaugseire andmete analüüsimise ja salvestamise kohta:

N1 Kaugseire tulemitele pääseb ligi vaid selleks piisavaid õiguseid omav kasutaja.

N2 Kaugseire tulemitest on lihtne teha tagavarakoopiaid.

N3 Kaugseire tulemid on salvestatud püsivalt.

N4 Maast tehtud piltidele peavad olema vastavusse seatud pildistatud ala koordinaadid.

N5 Kaugseire toores formaadis tulemid peab konverteerima vaadatavaks formaadiks (näiteks PNG, JPEG või GIF formaat).

N6 Kaugseire tulemite salvestamisel peab olema kiire andmete lugemine.

Nõuded kaugseire andmete kasutajaliidese kohta:

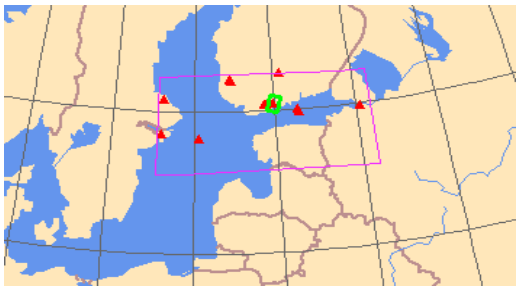
N1 Kaugseire tulemeid saab alla tõmmata toores formaadis (pildid Bayer'i mustrina).

N2 Pilte peab saama alla tõmmata GeoTIFF formaadis.

N3 Haldusrakendus peab võimaldama vaadata vähendatud resolutsiooniga kaugseire tulemeid, et langetada otsus, kas antud tulem on mõttekas alla tõmmata satelliidilt täisresolutsiooniga.

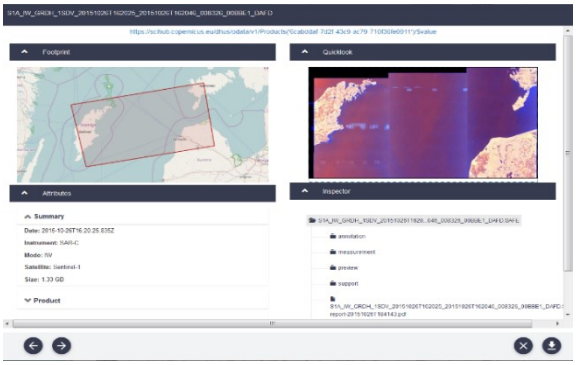
N4 Kaugseire andmete esitamiseks loodud kasutajaliides peab võimaldama kasutajal andmeid otsida ja filtreerida vastavalt oma vajadustele (piirkond, ajavahemik, pilvekate ja muud meta andmed). Kusjuures piltide otsimisel huvipakkuva ala koordinaatide järgi on eeldatud võimalust huvipakkuv ala valida kaardilt.

N5 Kaardil peaks olema võimalus välja tuua olemasolevate piltide keskkohad (Joonis 1).



Joonis 1. EOWEB'i[1] aluskaart, kus on esitatud olemasolevate piltide keskkohad

N6 Kaugseire tulemeid peab olema võimalik näha listis, kus on toodud välja oluline info ning reale pealevajutamisel peab avanema tulemi detailvaade, kus on muuhulgas välja toodud tulemi eelvaade ja asukoht kaardil (Joonis 2).



Joonis 2. scihub'i[2] detailvaade

2.1 Ülevaade satelliidi missioonijuhtimise tarkvarast

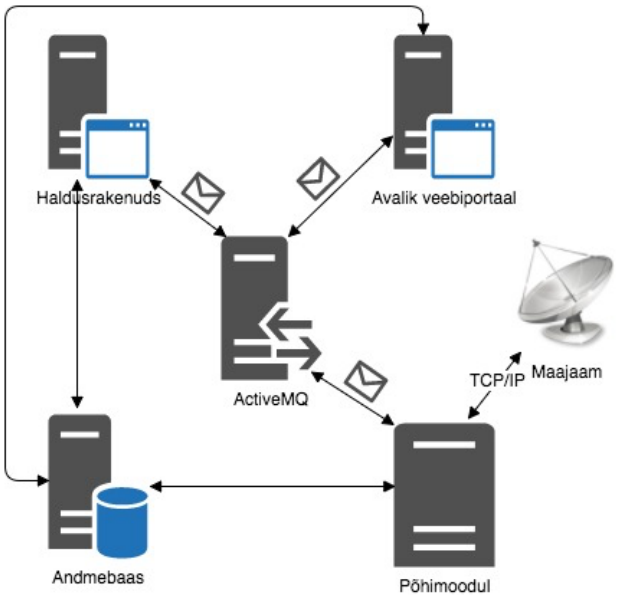
TTÜ satelliidiprojekti missioonijuhtimistarkvara on tudengite poolt loodud tarkvara, sest leiti, et olemasolevad tarkvaralahendused on kesised või ilma aktiivse arenduseta. Tarkvara on arendatud lähtuvalt agiilsetest põhimõtetest, et muutuvatele nõuetele oleks

[1] http://eoweb.dlr.de/

[2] https://scihub.copernicus.eu/

võimalik reageerida. Versioonihalduseks on kasutatud gitlab'i ja Dockerit on kasutatud projekti keskkonna ülesseadmiseks. [2]

Tarkvara on üles ehitatud 5 komponendina: haldusrakendus, avalik veebiportaal, põhiarvutus- ja planeerimismoodul, sõnumivahetus komponent ja andmebaas (Joonis 3). Sõnumivahetuseks kasutatakse programmi ActiveMQ, andmebaasina on kasutusel PostgreSQL. [2]



Joonis 3. Missioonijuhtimise tarkvara üldine arhitektuur [2]

Antud töös on edasi arendatud põhimoodulit ja haldusrakendust lisades sinna võimaluse salvestada kaugseire tulemeid, neid analüüsida ning luua kasutajaliides, et võimaldada kasutajal tulemeid vaadata, alla tõmmata ning langetada satelliidilt tulnud vähendatud piltide põhjal otsust, kas täissuuruses on mõtet pilt alla tõmmata.

Haldusrakendus on ehitatud kasutades Java Hipster vahendit, mis loob põhisüsteemi (*backend*) Spring bootiga, mis omakorda lähtub MVC (*Model-View-Controller*) arendusmustrist ning kasutajaliides on SPA (*Singel-Page-Application*), mis on realiseeritud Angular 4 raamistikuga. Kasutajaliides suhtleb põhisüsteemiga läbi REST'i (*REpresentational State Transfer*) põhimõtetel lähtuva liidese. Turvalisuse tagamiseks kasutatakse Spring security laiendit ning häid tavasid programmeerimisel. [2]

Arenduse käigus lähtutakse olemasolevast arhitektuurist ning kõik muudatused esitatakse selles töös. Uute tarkvaralahenduste lisamisel lähtutakse töös [2] toodud põhimõtetes kasutada võimalusel avatud lähtekoodiga tarkvara ja arvestatakse eelmainitud nõuetega.

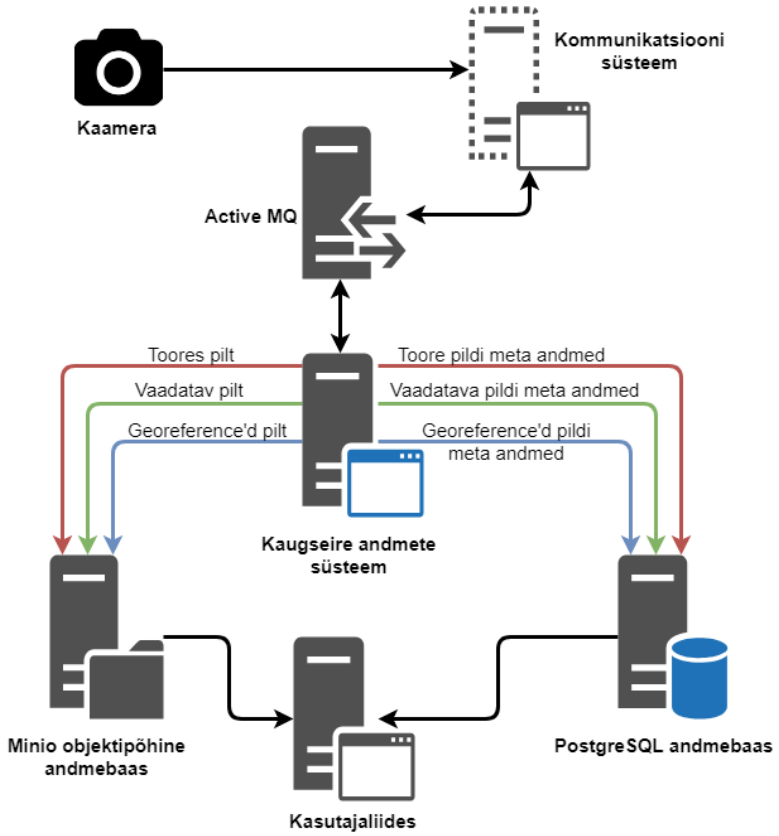
2.2 Antud töös loodava süsteemi arhitektuur

Joonis 4 kirjeldab töös loodud kaugseire tulemite protsessi. Protsess algab kaameraga, mis on satelliidil või testkeskkonna korral maal ja teeb pildi toores formaadis. Tehtud pilt liigub edasi kommunikatsiooni süsteemi.

Antud töös puudub kommunikatsiooni süsteem ning pilt tuleb ise panna Active MQ'sse, kust kaugseire andmetega tegelev moodul (selle töö jooksul loodud) salvestab kõigepealt toorel kujul sissetulnud meta andmed struktureeritult relatsioonilisse andmebaasi ja toore pildi andmed objektipõhisesse andmebaasi.

Järgmisena tegeleb kaugseire andmete süsteem toorete piltide töötlemise ja analüüsimisega ning tekkivad tulemid salvestab objektipõhisesse andmebaasi ning vastavad meta andmed salvestab PostgreSQL andmebaasi.

Viimaks on kasutajaliides, mis REST API (*Application Programming Interface*) abil kasutab PostgreSQL andmebaasi, et leida kasutaja otsingule vastavad pildid ning siis küsib Minio andmebaasist vastavaid pilte.



Joonis 4. Kaugseire tulemite protsess

3 Kaugseire tulemid ja infovajadus

Satelliidil on kaks kaamerat. Üks, mis pildistab nähtavas vahemikus ja teine, mis pildistab NIR (near infrared) vahemikus. Nähtavas sagedus vahemikus pildistamiseks kasutatakse sensorit MT9P031, mis väljastab Bayer'i mustri [3]. Saamaks Bayeri mustrist tagasi pilt on vaja teada ülesvõetud pildi resolutsiooni (pikkust ja laiust), mitu baiti on üks värv maatriksis ja mitu kaadrit pildistati.

Määramaks pildile vastavad ala maa peal on vaja teada satelliidi asukohta orbiidil pildistamise hetkel (piisab ajatemplist kui arvutada satelliidi positsioon maa peal) ning satelliidi nurka maa või muu taevakeha suhtes.

Vajalikud meta andmed on vaja saata maale koos kaamera tulemiga. Võimalikult väikese paketi huvides peaks info olema kodeeritud binaarkujul.

VER	TIME	W	H	BUF	AN	ON	KOOD	ANDMED
1	4	2	2	2	4/8	4/8	1	W*H*BUF*KOOD'ist tulenev suurus

Tabel 1. Meta andmete paketi kodeering, kus esimene rida kirjeldab struktuuri ja teine rida päise osa suurust baitides

AN ja ON tulpade suurus on 4 või 8 baiti olenevalt kas kasutada andmetüüpi *float* või *double*, mis omakorda sõltub seadmete mõõte täpsusest.

Struktuuriosade tähendused:

VER	versioon	Paketi päise versioon. Võimaldab andmete vajaduse muutudes paketti uuendada.
TIME	Pildistamise ajatempel	
W	Pildi laius	
H	Pildi kõrgus	
BUF	Kaadrite arv	Pildi korral on väärtus 1, video korral suurem.
AN	Satelliidi asimuut nurk	Nurk palju pilt on pööratud põhjasuunast päripäeva liikudes.

ON	Satelliidi *off-nadir* nurk	Nurk palju satelliit oli pööratud eemale joonest, kui satelliit oleks olnud suunatud otse maale pildistamise hetkel.
KOOD	Pildi tüübi kood	Siia saab kodeerida pildi/video kaamera (rgb või nir), formaadi (bayer'i muster, jpg, png, mp4) ja ka igale värvile kuluva baitide arvu (1 bait kui värv <= 8 bitti ja 2 baiti kui värvile vaja rohkem bitte)
ANDMED	Pildi andmed vastavalt koodile	Näiteks Bayer'i muster

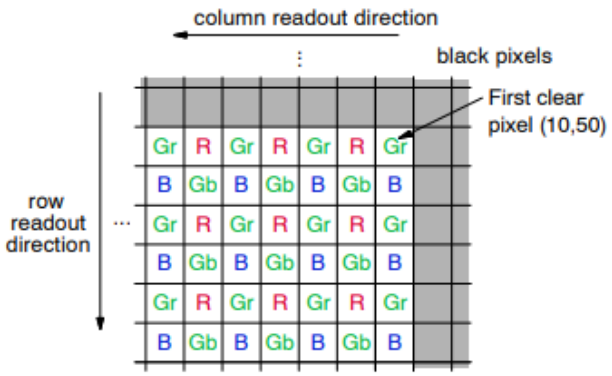
Tabel 2. Meta andmete paketi kodeeringu struktuuri selgitus

Tabel 1 esitatud paketis pole arvestatud kaamera parameetritega. Kui tekib vajadus ka parameetrid saata maale, siis on vaja eraldi paketti või teha paketist versioon 2.

MT9P031 võimaldab kuni 2592x1944 resolutsiooniga pilti, kus iga piksel on 12 bitti. Täisresolutsioonil saab video olla kuni 14 kaadrit sekundis. [3]

3.1 Sensori MT9P031 pildi formaat

Sensori väljund on Bayer'i muster, kus esimeses reas on vaheldumisi rohelisele (Gr) ja punasele (R) toonile reageerivad valgussensorid, teises reas on vaheldumisi sinisele (B) ja rohelisele (Gb) toonile reageerivad valgussensorid. Peale seda kordub esimene ja teine rida ülejäänud ridadel (Joonis 5). Veel väljastab sensor ülevalt 50 musta rida, alt 2 musta rida, vasakult 134 musta veergu ning paremalt 10 musta veergu, mis teeb kogu väljundi maksimaalseks suuruseks 2752 veergu ja 2004 rida, millest värvidega on 2592 veergu ning 1944 rida. Mustasid alasid saab kasutada näiteks pildi musta tooni valimisel. [3]

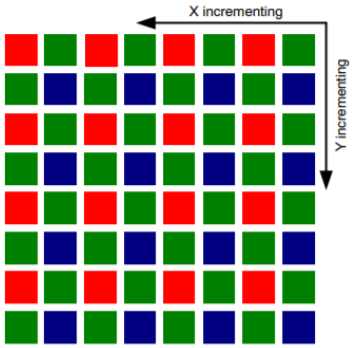


Joonis 5. MT9P031 Bayeri maatriks [3]

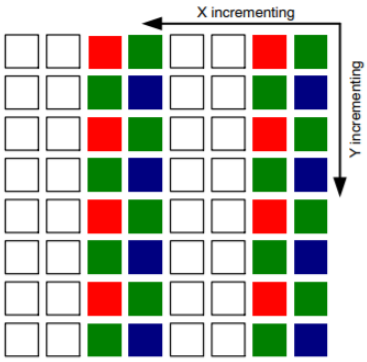
Sensor on küll sama, mis satelliidi kaamerasse pannakse, aga katsetamise eesmärgil on kaamerana kasutatud Leopard Imaging Inc'i kaamerat LI-5M03, mis on ühendatud Leopard Board 368 kiibiga (vt peatükk 4).

Sensori muudetavad parameetrid on: värvide globaalne võimendus (*gain*), säristus aeg (*exposure*), automaatne säristus aeg (*automatic exposure*) ning igal värvil (R, B, Gr, Gb) saab muuta võimendust eraldi [3]. Sensori draiveris on värvide võimendustel teistsugused nimed. Punase (R) ja sinise (B) värvi võimendused on nimetatud tasakaaludeks (vastavalt *red balance* ja *blue balance*), punase värvisensoriga vahelduv roheline (Gr) on nimega heledus (*brightness*) ja sinisega vahelduv rohelise (Gb) värvi võimendus on nimega *autogain*.

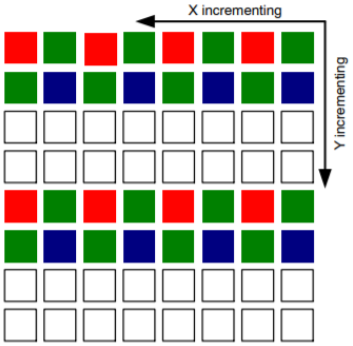
Veel võimaldab sensor lugeda värvi massiivi (*color array*) nii horisontaalselt (*horizontal flip*) kui ka vertikaalselt (*vertical flip*) ümberpööratuna. Sensoril on ka võimalus lahterdamiseks (*binning*) ja vahelejätmiseks (*skipping*) (Vt Joonis 6, Joonis 7, Joonis 8, Joonis 9) [3].



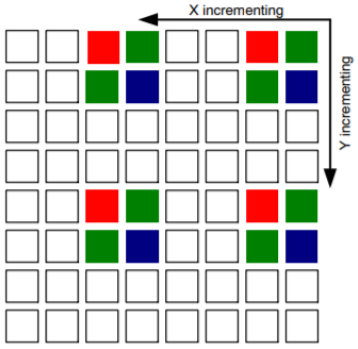
Joonis 6. Pikslite lugemine (ilma vahelejätmiseta) [3]



Joonis 7. Pikslite lugemine (veergude vahelejätmine 2 korda) [3]

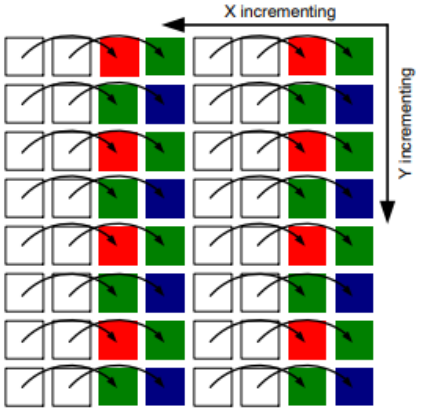


Joonis 8. Pikslite lugemine (ridade vahelejätmine 2 korda) [3]

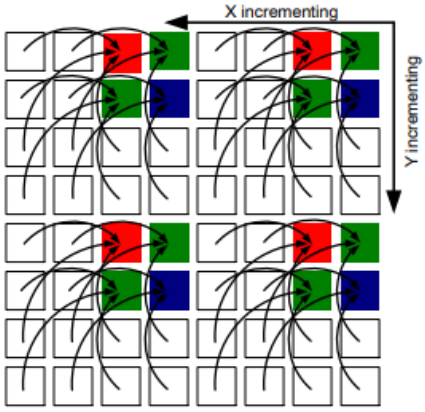


Joonis 9. Pikslite lugemine (veergude ja ridade vahelejätmine 2 korda) [3]

Lahterdamine tähendab kõrvuti asuvate sama värvi pikslite kombineerimine üheks piksliks (Vt Joonis 10, Joonis 11). Kombineerimisel võib liita kõrvuti olevad värvid või võtta nende keskmine. Lahterdamine vähendab pildi resolutsiooni, aga võib pakkuda pilte, millel on vähem artefakte. [3]



Joonis 10. Pikslite lahterdamine ridade kaupa [3]



Joonis 11. Pikslite lahterdamine ridade kaupa ja veergude kaupa [3]

3.2 Pildi toore formaadi vaatamine

Pilt tuleb Bayer'i mustrina, kus iga piksli kohta pole kõiki 3 värvikomponenti. Puuduvad komponendid tuleb arvutada välja interpolatsiooni (*demosaicking*) käigus. Eksisteerib mitmeid algoritme interpolatsiooniks. Antud projektis on kasutatud OpenCV[1] poolt pakutud lahendust (vt Lisa 1).

Pilti tahetakse GeoTIFF formaadis, kus värvikanalid on eraldatud ning pildi pikslitele on vastavusse seatud koordinaadid.

Vähendatud pildid peavad olema veebilehitsejas näidatavad. Veebi sobivad hulk formaate nagu PNG, JPEG, GIF ja teised. Tuleb valida formaat, mida võimalikult palju veebilehitsejaid on võimelised välja näitama.

OpenCV võimaldab tuntud formaatidesse konverteerimist, mis teeb formaadi nõuete täitmise lihtsaks.

[1] https://opencv.org/

4 Kaameraga pildi tegemine

Tekkis vajadus teha pilt kaameraga, millel on sama sensor, mis on satelliidi kaameras. Test kaamera on LI-5M03, mis on ühendatud Leopard Board 368 kiibi külge. Pildi tegemise eesmärk on näha pildi kvaliteeti enne kui satelliidi kaamerad pole valmis või pole tarkvara, mis võimaldaks nendega pilti teha. Veel saab tutvuda sensori parameetritega ning kalibreerida sensorit.

Kiip on üles seatud kasutades RidgeRun Izaru SDK[1]-d (*Software Development Kit*), mis sisaldab pildistamiseks vajalikke programme ja draivereid. Täpsemalt kasutatakse GStreamer[2] utiliiti, mis kasutab V4L[3] (*video 4 linux*) API'l loodud sensori draiverit.

Tegemist on üldkasutuseks mõeldud lahendusega, mis võimaldab näiteks kaamera välja vahetada ilma pildistamise programmi ümber kirjutamata. Negatiivne antud lahenduse puhul on üldises V4L API's sensori spetsiifiliste võimaluste puudumine. Veel tuleb arvestada, et test kaameraga pildistamise tarkvara on erinev satelliidile tulevast tarkvarast, mis võib tekitada erinevusi antud töös tehtud piltide ja satelliidiga tehtud piltide vahel.

4.1 Vajaliku tarkvara seadistamine

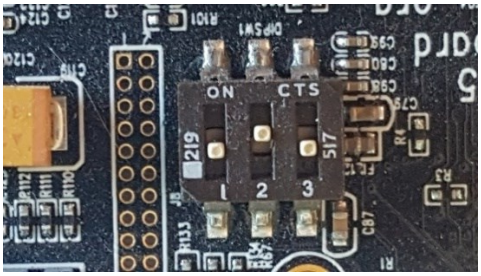
Kiibi kasutamiseks installitakse SD kaardile operatsioonisüsteem ning kiip pannakse SD kaardilt käivitamise režiimi (Joonis 12). Pildi tegemise lihtsustamiseks kasutatav RidgeRun Irazu SDK (tasuta versioon) võimaldab lihtsalt konfigureerida kiibi käivitamiseks vajaliku kettakujutise loomist. RidgeRun SDK tuleb üles seada peaarvutil (soovitatakse Ubuntut). Ülesseadisel on lähtutud õpetustest [4] ja [5]. RidgeRun SDK on

[1] https://www.ridgerun.com/store/Evaluation-SDK-for-DM36x-for-Leopardboard-and-TI-EVM-p76368725

[2] https://gstreamer.freedesktop.org/

[3] https://linuxtv.org/downloads/v4l-dvb-apis/uapi/v4l/v4l2.html

antud lähtekoodina, mis tuleb kompileerida plaadi jaoks. Selleks on kasutusel Code Sourcery ARM toolchain [6].



Joonis 12 Kiibi käivitamise režiimi muutmise nupud. Hetkel olekus SD kaardilt käivitamine.

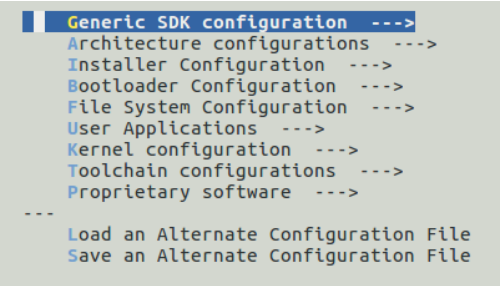
Switch 1 (BTSEL2)	Switch 2 (BTSEL1)	Switch 3 (BTSEL0)	Boot mode
off	off	off	NAND Boot Mode
off	on	off	SD Card Boot Mode
off	on	on	UART Boot Mode
on	off	off	USB Boot Mode
on	on	off	EMAC Boot Mode

Joonis 13 LeopardBoard kiibi käivitamise režiimid [7]

RidgeRun SDK installeerimiseks on veel vajalik DVSDK 4[1] (Texas Instruments' Digital Video Software Development Kit). DVSDK toetab vaid Ubuntu 10.04 LTS 32-bit põhiarvuti operatsioonisüsteemi, aga sellest piirangust saab mööda, kui käivitada installatsioon käsuga " ./dvsdk_dm368-evm_4_02_00_06_setuplinux --forcehost". [5]

DVSDK sisaldab tarkvara, et TI (Texas Instruments) riistvaraga kiipe kiiresti tööle saada. Antud versioonis on muuhulgas näiteks Linux kernel 2.6.32.17, Boot loader ja DMAI (DaVinci Multimedia Application Interface). [8]

Pärast RidgeRun SDK installimist, saab installitud kaustas käsuga "make config" lahti kasutajaliidese (Joonis 14), kus saab konfigureerida parameetreid nagu kerneli mälu algusaadressid, mälujaotused, installeeritavad programmid, kuidas kiibile toimetada loodav kettakujutis ja palju muud. Kõigel on algväärtused ning sulgedes menüü saab käsuga "make" kompileerida tarkvara kokku ning käsuga "make install" luua kettakujutise fail kiibile laadimiseks.

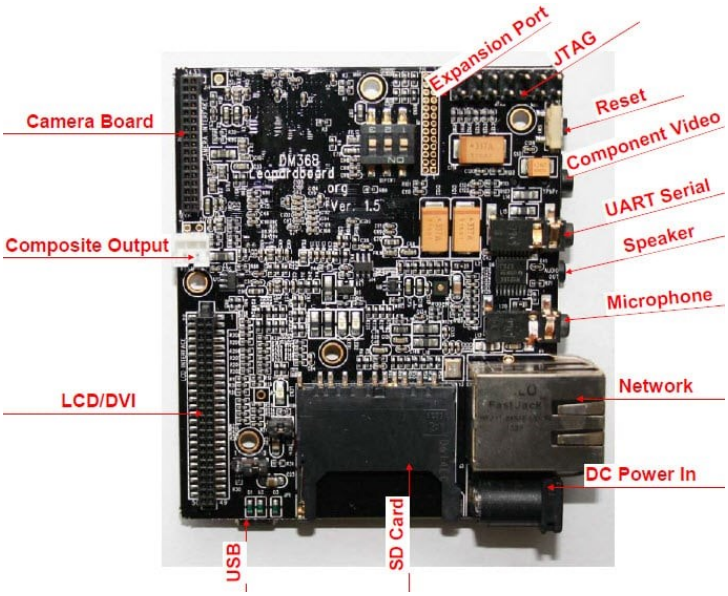


Joonis 14. Käsu make config avatud SDK seadistuse peamenüü

[1] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/dvsdk/DVSDK_4_00/4_02_00_06/index_FDS.html

4.2 Plaadiga suhtlus

Plaadiga suhtlemiseks tuleb kiibi jadaport UART Serial (vt Joonis 15) ühendada arvutiga. Selleks on kaabel, kus kiibiga ühenduses olev ots on 2.5mm Stereo ühendus ja teises otsas on female DB9 ühendus (vt Joonis 16). Peaarvutina kasutatavas arvutis puudub male DB9 ühendus, selleks kasutati teist kaablit, kus ühes otsas on male DB9 ja teises otsas arvutisse ühendamiseks USB ühendus.



Joonis 15. LeopardBoard DM368 kiip [9]

Joonis 16. Kaabel, mille ühes otsas on 2.5mm Stereo ühendus (paremal) ja teises otsas on female DB9 ühendus (vasakul) [9]

Kaablitega ühendus on jadaühendus. Suhtluseks on kasutatud Minicom[1] käsurea utiliiti, kus seadistused suhtluseks on 115200 bitti sekundis, kus 8 bitti on andmebitid ilma paarsusbitita ja 1 *stop* bitiga.

Lisaks kasutatakse ühendust internetti läbi etherneti kaabli failide saatmiseks. Põhiliselt suuremate failide nagu pildi ja video failide saatmiseks kiibilt peaarvutile.

4.3 Pildi tegemise katsed

Piltide tegemiseks kasutatakse GStreamer teeki. GStreamer on teek, mis võimaldab luua graafe multimeediat haldavatest komponentidest. GStreamer toetab suurel hulgal audio ja video esitamise, ülekandmise, kombineerimise ja redigeerimise komponente. GStreameril on kontseptsioon meedia allikast (*source*), mis tuleb ühendada valamusse (*sink*). Allika ja valamu vahel on komponendid, mis täidavad oma ülesannet neid läbivate multimeedia puhvrite peal. Allikaid ja valamuid võib olla mitu. Allikad, komponendid ja valamud kokku moodustavadki graafi.

GStreameriga tuleb kaasa käsureautiliit gst-launch, mis võimaldab kirjeldada allikaid, valamuid ja nende vahel olevaid komponente käsureal. Näiteks käsk "gst-launch-1.0 filesrc location=videofile.mpg ! dvddemux ! mpeg2dec ! xvimagesink" loeb video faili videofile.mpg ja väljastab vaid video osa ilma helita X aknasse[2]. [10]

Pildi tegemiseks on GStreameril komponent v4l2src, mis on V4L API juurdepääs GStreameri jaoks [11]. RidgeRun SDK-ga kaasatulev MT0P031 driver on ehitatud V4L API peale. Probleemiks on olukord, kus GStreameri graafi täitmisel tekib viga, siis üldjuhul on vea sõnum väga halb ning pildi fail luuakse ikkagi, aga selle suuruseks on 0 baiti.

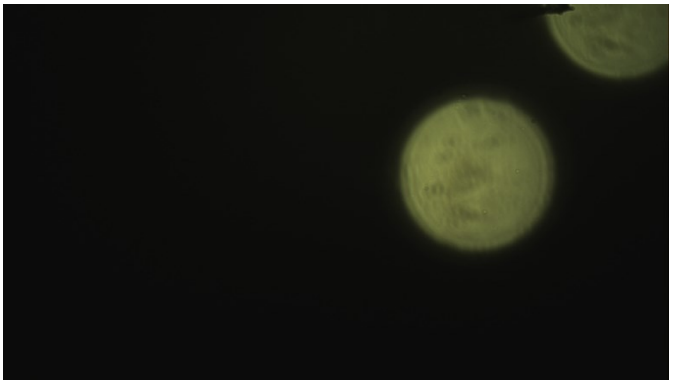
4.3.1 Esimesed katsed

Esimesed õnnestunud pildid olid YUV värviruumis, mis tekitas segadust, sest sensor peaks väljastama Bayer'i mustri. Piltide tegemisel lähtuti käskudest, mis olid kirjeldatud allikas [12]. Näiteks käsuga

[1] https://alioth.debian.org/projects/minicom/
[2] https://www.x.org/wiki/

```
gst-launch -e v4l2src device=/dev/video0 chain-ipipe=true always-copy=false
num-buffers=1 ! capsfilter caps='video/x-raw-
yuv,format=(fourcc)NV12,width=1920,height=1080' ! dmaiaccel ! filesink
location='image_1920x1080.yuv'
```

sai pildi Joonis 17. Üldiselt olid tehtavad pildid üleni mustad või valged olenevalt valgusest. Käsul on näha formaadi kirjeldust video/x-raw-yuv,format=(fourcc)NV12, aga muutes see otse käsuks video/x-raw-bayer,format=bggr, mis on sensori poolt väljastatav formaat, viskas GStreamer veateate, et ei suutnud formaati kaameralt küsida (Joonis 18).



Joonis 17. Pilt laelambist. Toores NV12 pilt on konverteeritud vaadatavaks kasutades http://rawpixels.net/

```
/ # gst-launch -e v4l2src device=/dev/video0 chain-ipipe=true always-copy=false
num-buffers=1 ! capsfilter caps='video/x-raw-bayer,format=bggr,width=1920,height
=1088' ! dmaiaccel ! filesink location='image_1920x1088.yuv'
Setting pipeline to PAUSED ...
vpfe-capture vpfe-capture: IPIPE Chained
vpfe-capture vpfe-capture: Resizer present
vpfe-capture vpfe-capture: standard not supported
ERROR: Pipeline doesn't want to pause.
WARNING: from element /GstPipeline:pipeline0/GstV4l2Src:v4l2src0: Failed to set norm for device '/dev/video0'.
Additional debug info:
../../../src/sys/v4l2/v4l2_calls.c(743): gst_v4l2_set_norm (): /GstPipeline:pipeline0/GstV4l2Src:v4l2src0:
system error: Invalid argument
ERROR: from element /GstPipeline:pipeline0/GstV4l2Src:v4l2src0: Could not negotiate format
Additional debug info:
../../../../src/libs/gst/base/gstbasesrc.c(2778): gst_base_src_start (): /GstPipeline:pipeline0/GstV4l2Src:v4l2src0:
Check your filtered caps, if any
Setting pipeline to NULL ...
Freeing pipeline ...
```

Joonis 18. Ebaõnnestunud käsu näide

```
/ # gst-launch -e v4l2src device=/dev/video0 chain-ipipe=true always-copy=false
num-buffers=1 ! capsfilter caps='video/x-raw-yuv,format=(fourcc)NV12,width=1920,
height=1088' ! dmaiaccel ! filesink location='image_1920x1088.yuv'
Setting pipeline to PAUSED ...
vpfe-capture vpfe-capture: IPIPE Chained
vpfe-capture vpfe-capture: Resizer present
vpfe-capture vpfe-capture: standard not supported
Pipeline is live and does not need PREROLL ...
WARNING: from element /GstPipeline:pipeline0/GstV4l2Src:v4l2src0: Failed to set norm for device '/dev/video0'.
Additional debug info:
../../../src/sys/v4l2/v4l2_calls.c(743): gst_v4l2_set_norm (): /GstPipeline:pipeline0/GstV4l2Src:v4l2src0:
system error: Invalid argument
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Got EOS from element "pipeline0".
Execution ended after 247272837 ns.
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...
```

Joonis 19. Õnnestunud käsu näide

Teine probleem, mis esines tulenes RidgeRun SDK algväärtustest, millega suurim pilt, mida teha sai oli resolutsiooniga 1920x1088, aga sensor võimaldab pilti suurusega 2592x1944. Proovides teha suuremat pilti tuli veateade:

```
vpfe-capture vpfe-capture: dma_alloc_coherent size 7561216 failed
Segmentation fault
```

Pilte ei pidanud ainult toores formaadis küsima. Käsuga

```
gst-launch -e v4l2src device=/dev/video0 chain-ipipe=true always-copy=false
num-buffers=1 ! capsfilter caps='video/x-raw-
yuv,format=(fourcc)NV12,width=1920, height=1088' ! dmaiaccel ! dmaienc_jpeg !
filesink location='image_1920x1088.jpg'
```

sai JPEG formaadis pildi, aga esinesid samad probleemid ja piirangud, mis toores formaadis pildi tegemisel.

4.3.2 Parandused

Pildi suuruse limiidi probleemi lahendamine oli kerge, sest leidus õpetus [13], kus on täpselt esile toodud SDK konfiguratsioon (Joonis 20 ja Joonis 21), mis võimaldas teha täisresolutsiooniga pilte. Õpetuses on esile toodud, et seadistused võimaldavad maksimaalselt resolutsiooni 2176x1944, aga proovimise käigus selgus, et sai kätte ka oodatud resolutsiooni 2592x1944.

```
    *** SDRAM Definitions ***
(0x80000000) SDRAM Base address
(0x08000000) SDRAM size
    *** SDRAM Memory map ***
(0x80000100) Kernel ATAGS parameter address
(0x80008000) Kernel physical start address (multiple of 4096 bytes)
(0x80008000) Kernel entry physical address
[ ] Use kernel compression
    Video Output (Component)  --->
    Component Standard (1080I-30)  --->
    Maximum Video Output Buffer Size (480P)  --->
    Maximum Video Input Buffer Size (Other (enter buffer size))  --->
(50761728) Buffer size for v4l2 input (recommend 3 buffers min)
(131072) NAND block size
(2048) NAND page size
```

Joonis 20. Uus SDK konfiguratsiooni menüü Architecture configuration seadistus

```
[*] Texas Instruments CodecEngine for DM368
[*] Automatically calculate kernel memory size
(0x4D00000) Memory reserved for DVSDK
[*] Automatically setup CMEM at the beggining of the reserved memory area
(0x1900000) Amount of reserved memory for cmemk
[ ] Install TI DMAI demos on file system
[ ] Enable CMEM debug library
[*] Gstreamer DMAI Plugins
      SVN Plugins Source (Branch)  --->
```

Joonis 21. Uus SDK konfiguratsiooni menüü Proprietary software seadistus

Bayer'i mustri kättesaamine oli keerulisem, sest ei leidunud ühtegi allikat, mis oleks selgelt välja toonud lahenduse. Pooleldi katse-eksituse meetodil ja pooleldi dokumentatsioonides kolades jõuti tulemusele, et pildistamise käsus v4l2src argument chain-ipipe=true peale väljastas v4l2src pildi NV12 või UYVY formaadis. Selle parameetri vaikimisi väärtus on samuti true, mis tähendas, et proovimise käigus argumendi ära jätmine ei muutnud tulemust.

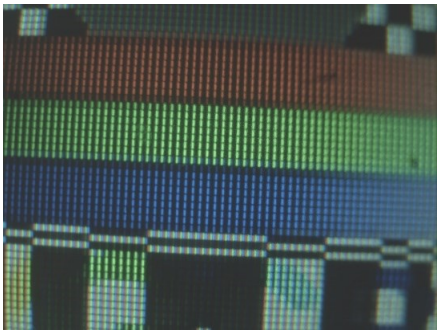
Antud järeldusele jõudes sai muuta argumendi chain-ipipe väärtuseks false ja tekkis võimalus saada pilte otse Bayer'i mustrina. Kõrvalnähtusena kadus ära võimalus kasutada dmaiaccel komponenti, mille eesmärgiks on tagada, et seda läbivad puhvrid (pildid) on mälus üksteise kõrval [14] ning ka dmaienc_jpeg komponenti, mis konverteeris toore pildi JPEG formaati.

4.3.3 Praegu parimad pildid

Käsuga

```
gst-launch -e v4l2src chain-ipipe=false always-copy=false num-buffers=1 !
filesink location=image_of_screen.raw
```

saab kätte Joonis 22 pildi. Toore Bayer'i mustri konverteerimiseks kasutatakse OpenCV teeki (vt Lisa 2)



Joonis 22. Pilt arvuti ekraanist. Toores Bayeri muster on konverteeritud vaadatavaks kasutades OpenCV teeki

Piltide vaatamiseks tuleb teha pilt, saata see kiibilt arvutisse ja konverteerida vaadatavaks formaadiks. Suurem osa tuleb täita käsureal, mis teeb protsessi läbimise ajakulukaks. Protsessi lihtsustamise huvides loodi Python'i skript (Lisa 3), mis kuulab TCP ühendust ning ootab pildi saabumist, konverteerib selle nähtavaks formaadiks ning näitab kasutajale. Praegusel kujul see skript ei salvesta tulemust.

Kasutamiseks piisab käsust:

```
gst-launch -e v4l2src chain-ipipe=false always-copy=false ! 'video/x-raw-
bayer,format=bggr,width=640,height=480' ! tcpclientsink host=<<IP>>
port=<<PORT>> &
```

kus <<IP>> on vaja asendada serveri IP-ga ja <<PORT>> on vaja asendada serveri pordiga. Sel juhul pole ka mõtet määrata v4l2src argumenti *num-buffers*, sest siis tuleb pidev jada piltidest (video).

4.4 Sensori MT9P031 parameetrite muutmine

Üks peamisi põhjuseid, miks oli vaja pilt teha oli näha mis võimalusi pakub sensor, et seda saaks kalibreerida.

Selgus, et RidgeRun SDKs leidub vahend nimega ipiped[1], millega saab muuta kaamera parameetreid, aga kui SDK konfiguratsioonist valida, et ipiped installida, siis kompileerimine ebaõnnestub veateatega, et puudub sõltuvus librraew[2], mis on tasuline.

Alternatiivse lahendusena sai loodud programm, mis võimaldab muuta kaamera parameetreid. Kasutatud on V4L2 API't, mis määrab kaameraga suhtlemiseks ioctl käsu argumendid.

[1] https://github.com/RidgeRun/ipiped

[2] https://www.ridgerun.com/store/Auto-Exposure-Auto-White-Balance-DM36x-processor-only-p59063248

5 Kaugseire tulemite salvestamine, analüüs ja kasutajale esitamine

Maajaama jõudnud kaugseire tulemid on vaja salvestada ning viia kujule, mis oleksid kasulikud inimesele.

5.1 Salvestamine

Kaugseire tulemid on põhiliselt pildid ja videod. Suurim resolutsioon, mida sensor suudab väljastada on 2752x2004, aga kasuliku infoga on suurim resolutsioon 2592x1944 (vt peatükk 3.1). Tuleb arvestada, et kaugseire tulemeid on vaja salvestada vaid ühe korra ning edasist vajadust neid muuta ei ole. Samas võib tekkida vajadus salvestada sama kaugseire tulemit korduvalt erinevates analüüsi staadiumides. Kaugseire tulemite lugemisi on siiski oodata mitu korda rohkem, mis tähendab, et salvestamisel tuleb keskenduda kiirele andmete lugemisele.

Otsustamaks parim lahendus kaugseire tulemite salvestamiseks on läbi viidus võrdlus põhiliselt kolme salvestusvõimaluse vahel: objektipõhine salvestussüsteem (*object storage*), failisüsteem ja relatsiooniline andmebaas (Tabel 3).

Objektipõhise salvestussüsteemi esindajaks on valitud Minio, sest tegemist on avatud lähtekoodiga lahendusega. Relatsioonilise andmebaasi esindab juba missioonitarkvaras kasutusel olev PostgreSQL andmebaas.

	Minio	**failisüsteem**	**PostgreSQL**
Kaitse riistvaravigade vastu	Mitme serveri jooksutamine koos andmete sünkroniseerimisega ja varukoopiad [15], [16]	Kõvaketta sektsioonideks jagamine ja varukoopiad	varukoopiad
Kaitse tarkvara planeerimata töö lõpetamise vastu	Mitme serveri jooksutamine [16]	-	*write ahead logging*

Ühilduvus	Amazon S3 süsteemiga ühilduv	-	*Foreign table* võimaldab ühendust välistesse andmebaasidesse
Kiirus	Piisavalt kiire suurte failide puhul	Kiire lugemine ja kirjutamine ka suurte failide korral	Suurte failide puhul aeglane
Integreerimise lisatöö	Keskmine, tuleb lisada projekti Minio klient ja teha tarkvarasse kiht peitmaks otsesed Minio API väljakutsed	Suur, tuleb teha uus moodul, mis tegeleb failiõigustega, andmete varukoopiate tegemisega ja kõige muuga	Minimaalne, vaid klassid andmete küsimiseks andmebaasist
Dokumentatsioon	Hea	Peab ise looma vastavalt ise loodud moodulile	Väga hea ja pika ajalooga
Meta andmete salvestamine	Saab salvestada koos failidega, aga üksikute andmete vaatamiseks tuleb kogu fail mällu lugeda	Vaja salvestada eraldi failis	Tabelites saab salvestada normaliseeritult, teha päringuid üksikute andmete pihta
Juurdepääsu logimine	Vaid vigade logimine, juurdepääsu logimine tuleb teha läbi serveri [17]	Peab tegema ise	Saab seadistada juurdepääsu logimise
Turvalisus	Võimaldab turvalist ühendust [18]	Vaja ise teha	Võimaldab turvalist ühendust [19]
Juurdepääsu õigus	Üks kasutaja täisõigustega, failidele on võimalik anda ajutisi õiguseid. Reaalsuses vaja peita juurdepääs missioonijuhtimise tarkvara API taha	Vaja ise teha	Saab teha kasutajaid erinevate õigustega
Failile leidmine	Kasutades id ja *bucket*'i nime abil	Vaja ise hallata failide hierarhiat ja otsingut [20]	SQL laused

Võimekuse laiendamine	Saab lisada servereid [16]	-	Indeksid, jagada andmebaas serveritesse
Andmetele reageerimine	Saab koodis kuulata *event*'e [21]	-	Trigerid
Varia		võimaldab faile vaadata/lugeda bait haaval. Võimaldab faile muuta bait haaval [22]	PostGIS laiendus spetsialiseeritud geograafilise info salvestamisele

Tabel 3. Salvestussüsteemide võrdlus

Lähtudes võrdlusest on otsustatud kaugseire tulemite salvestamiseks otsustatud kasutada minio objektisalvestus tarkvara. Minio on avatud lähtekoodiga ning koos Amazon S3 API'ga ühilduv objektisalvestus tarkvara, millel eksisteerib Java klient, korralik dokumentatsioon ning lihtne ühilduvus docker konteineriga.

Objektisalvestus tarkvara kasuks on otsustatud, sest failisüsteemi salvestamisel tuleb luua eraldi liides failide juurde ligipääsemiseks, kus tuleb tegeleda failisüsteemi õigustega, kasutajate/ligipääsu õigustega, samaaegsete protsessidega (*multithreading*), manuaalsete varukoopiate tegemisega ning tuleb tagada andmete püsivus. Kõik see tekitab ohtu vigadeks ning Minio'l on juba olemas vajalikud omadused. Lisaks ei anna failisüsteemi salvestamine juurde piisavalt kasu, et see oleks väärt kogu lisatööd.

Meta andmed salvestatakse andmebaasi koos viitega Minio id'le, sest andmebaas võimaldab mugavamaid otsinguid ning PostgreSQL andmebaasis on laiendus PostGIS, mis võimaldab geograafiliste andmete salvestamise ning nende põhjal päringute tegemist.

Salvestus osa realisatsiooni puhul on kasutatud unit teste ning ka manuaalset testimist, et tagada salvestuse püsivus ning meta andmete ja pildifailide kooskõla.

5.2 Analüüs

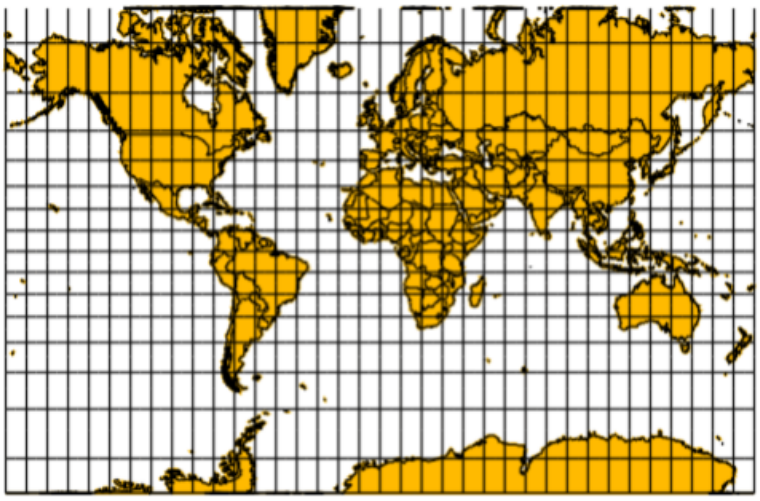
Pildid ja videod tuleb seada vastavusse maal pildistatava ala koordinaatidega. Selleks on vaja teada kus satelliit asus pildi tegemise hetkel ning kuhu olid kaamerad suunatud.

5.2.1 Georeferencing

Georeferencing tähendab geograafiliste andmete joondamist mingisse teadaolevasse koordinaatide süsteemi [23]. Antud projektis on geograafilisteks andmeteks satelliidi tehtud pildid maast, millele tuleb vastavusse viia koordinaadid. Koordinaatsüsteeme, mis kirjeldavad maad on mitmeid. Näiteks levinud pikkus- ja laiuskraadi põhised või maa tükeldamine tsoonideks ja igale tsoonile oma nullpunkti vastavusse seadmine.

Ka pikkus- ja laiuskraadide põhistel süsteemisel võivad olla erinevad nullpunktid. Erinevad koordinaatsüsteemid on standardiseeritud. Näiteks eksisteerib standard WGS84 EPSG (*European Petrolium Survey Group*) andmebaasis USA kaitseministeeriumi poolt, mida kasutatakse GPS süsteemides. WGS84 nullpunk on maa keskel ja koordinaatsüsteem arvestab maaga kui ellipsoidiga [24]. Google kasutab aplikatsioonis Maps koordinaatsüsteemi EPSG:900913 (Google Maps Global Mercator) [25], mis arvestab maaga kui täiusliku keraga, et arvutusi lihtsa ja kiirena hoida. Need süsteemid kujutavad 3D maailma, aga leiduvad ka tasandile suunatud projektsioone.

Tuntud projektsioon on Mercatori projektsioon (Joonis 23), kus pindala näib tegelikkusest suuremana mida kaugemale lõunasse või põhja ekvaatorist minna. Veel on kasutusel projektsioon UTM (Universal Transverse Mercator), kus maakera on jaotatud tsoonideks. UTM korral kasutatakse koordinaatidena meetreid tsooni algpunktist. Tuleb arvestada, et iga tsoon on oma koordinaatsüsteem [26]. Näiteks Tallinn asub UTM tsoonis 35N (kirjeldatud standardis EPSG:32635) koordinaatidel 372012, 6589858.



Joonis 23. Mercatori projektsioon [27]

Koordinaatsüsteeme on palju, sest maa on ümmargune ning seda ei saa ilma moonutuseta kujutada tasandil ning erinevad süsteemid pakuvad võimalusi valida, mis osad moonduvad.

Pilti saab vaadata kui koordinaatsüsteemi pikslist positsioonil (0, 0) pikslini positsioonil (pildi laius, pildi kõrgus). *Georeference*'i tulemusel on vaja leida transformatsioon, mis seab igale pikslile vastavusse koordinaadi valitud koordinaatsüsteemist. Selleks saab kasutada afiinset transformatsiooni.

Afiinne transformatsioon võimaldab muuta koordinaatsüsteemi kallet (*skew*), skaalat (*scale*), rotatsiooni, positsiooni (*translation*) ja pööramist (*flips*). Afiinne transformatsioon hoiab sirgeid jooni ning paralleelsust [28], [29]. See tähendab, et vanas koordinaatsüsteemis sirged jooned on ka uues koordinaatsüsteemis sirged, aga näiteks joonte pikkused võivad olla muutunud.

Afiinne transformatsioon tasandil kujutab endas arvutust

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_{00}x+m_{01}y+ m_{02} \\ m_{10}x+ m_{11}y+ m_{12} \\ 1 \end{bmatrix} \quad (1)$$

kus x' ja y' on lõppsüsteemi koordinaadid, x ja y on algsüsteeme koordinaadid. Afiinse maatriksi koefitsiendid m_{ij} määravad ära transformatsiooni [29]. Suures pildis on arvutus kirjutatav valemina

$$x' = Ax \quad (2)$$

kus

$$x' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, x = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \; ja \; A = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ 0 & 0 & 1 \end{bmatrix}$$

Afiinset maatriksit kasutab veel georeference'imisel failistandard *world file*, mis on tekstifail, kus igal real on üks afiinse maatriksi koefitsient. *World file* kirjeldab kuidas

pildi koordinaatidest saada päris maailmas vastavad koordinaadid (valitud koordinaatsüsteemis). [28]

5.2.2 Koefitsientide leidmine

Lihtsaim viis on võtta pildi alumine vasak piksel ning määrata see väikseimaks koordinaadiks, pildi ülemine parem piksel määrata suurimaks koordinaadiks ja nende põhjal teha ristkülik. See meetod arvestab, et pilt on suunatud põhja. Muus suunas pildi korral tuleb enne pilt pöörata õigesse suunda, aga see pole hea, sest muudab originaali.

Tuleb luua afiinne maatriks, mis seab pildi igale pikslile vastavusse koordinaadi maailmas valitud koordinaatsüsteemi järgi. Lihtsaimal juhul on lahendusel lähtutud allikast [30]. Lihtsaima juhu korral tuleb võtta translatsiooni maatriks

$$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

kus tx väärtuseks valida pildi alumisele vasakule nurgale vastav maailma x koordinaat korrutatud läbi -1'ga, ty valida pildi alumisele vasakule nurgale vastav maailma y koordinaat korrutatud läbi -1'ga.

Koordinaatsüsteemi liigutamisest üksi ei piisa. Koordinaatsüsteemi peab ka skaleerima, et oleks vastavus pildiga. Selleks tuleb koostada maatriks

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

kus s_x'iks tuleb valida suhe

$$\frac{pildi\ pikkus\ pikslites}{suurim\ maailma\ x\ koordinaat\ pildil - väikseim\ maailma\ x\ koordinaat\ pildil}$$

ja s_y väärtuseks tuleb valida sarnaselt

$$\frac{pildi\ kõrgus\ pikslites}{suurim\ maailma\ y\ koordinaat\ pildil - väikseim\ maailma\ y\ koordinaat\ pildil}$$

Suurim maailma koordinaatpaar on pildi paremale ülemisele pikslile vastav koordinaat ning väikseim maailma koordinaatpaar on pildi vasakule alumisele pikslile vastav koordinaat.

Piltide piksleid hakatakse tavaliselt lugema (pikslite indeksid suurenevad) ülevalt vasakust nurgast alla paremasse nurka. Seetõttu on vaja pildi y suund ümber pöörata maatriksiga

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & ty \\ 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

Kus ty on pildi kõrgus pikslites.

Viimase maatriksi saamiseks tuleb maatriksid kokku korrutada järjekorras (5), (4), (3). Javas on klass AffineTransform, mille abil saab korrutamise teha lihtsalt [30]:

```
AffineTransform translate = AffineTransform.getTranslateInstance(-tx, -ty);
AffineTransform scale = AffineTransform.getScaleInstance(sx, sy);
AffineTransform mirror_y = new AffineTransform(1, 0, 0, -1, 0, ty);

AffineTransform worldToPixel = new AffineTransform(mirror_y);
worldToPixel.concatenate(scale);
worldToPixel.concatenate(translate);
```

Lõpptulemuseks on maatriks, mis seab maailma koordinaadile vastavusse pildi piksli koordinaadi. Eesmärk on saada vastupidine maatriks, selleks tuleb võtta maatriks pöördmaatriks, mis koodis tähendab [30]:

```
try {
	AffineTransform pixelToWorld = worldToPix.createInverse();
} catch (NoninvertibleTransformException e) {
	e.printStackTrace();
}
```

Java AffineTransform klassi dokumentatsioonis [29] on veel kirjeldatud maatriksid rotatsiooniks:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Ja kaldeks:

$$\begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7)$$

Võiks oodata, et rotatsiooni maatriksi abil saab eemaldada nõude, et pilt peab olema põhja suunatud, aga rotatsiooni lisades pole saadud oodatavat tulemust.

Alternatiivselt on võimalik manuaalselt leida punkte, mis vastavad satelliidi pildid ja mingil tausta kaardil, mis on teadaolevas koordinaatsüsteemis. Eksisteerib mitmeid programme, mis seda lihtsustavad nagu QGIS või ArcGIS. Ise olen proovinud QGIS, aga seal muudetakse rotatsiooni korral originaalpilti (Originaal Joonis 24 ja moondatud Joonis 25).

Joonis 24. Google maps'ist võetud pilt (otse ülevalt alla ja 90 kraadi põhjast eemale pööratud)

Joonis 25. Google maps'ist võetud pilt, millel on katsetamise mõttes on proovitud koordinaate vastavusse seada kasutades QGIS vahendit lisandiga Georeferencer

Teades vähemalt kolme piksli vastavust päris koordinaatidele on võimalik arvutada afiinse maatriksi koefitsiendid. Kui loome maatriksi teadaolevate koordinaatidega

$$X = \begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 \\ & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix}, \quad (8)$$

kus n on teadaolevate koordinaatide arv, vektori afiinse maatriksi koefitsientidest

$$a = [m_{00} \quad m_{01} \quad m_{02} \quad m_{10} \quad m_{11} \quad m_{12}]^T \quad (9)$$

ja vektori, kus elemendid vastavad (8) elementidele

$$x' = \begin{bmatrix} x'_0 \\ y'_0 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}, \quad (10)$$

siis kehtib

$$Xa = x'$$

kust

$$a = X \setminus x'$$

mille lahendus a vähimruutude meetodil on otsitava afiinse maatriksi koefitsiendid. [31]

Antud lahenduse puhul töötab arvutus test punktide korral, aga kasutades java Geotools teeki transformatsiooni läbiviimiseks sattusid x ja y telg vahetusse. Vahetades originaalvalemis x' positsioonid olid näidispunktidel nüüd teljed järjekorras y ja siis x, aga peale transformatsiooni Geotools'i poolt oli ikka samasugune väljund nagu ilma telgesid vahetamata.

5.3 Kaugseire tulemite kasutajale esitamine

Kasutaja peab saama vaadata kõiki pilte, olema võimeline neid filtreerima ja otsima vastavalt vajadusele. Oodatud on kaardi olemasolu piltide otsimiseks/filtreerimiseks. See on lahendatud kasutades Google maps API't kaardi kuvamiseks (Lisa 4, Lisa 5), võimaldamaks kasutaja valida filtreerimise ala ning antud ala koordinaatide kättesaamiseks.

Koordinaadid saadetakse põhisüsteemi, kus andmebaasis salvestatud piltide regiooni/koordinaatide alusel leitakse kasutaja valitud alasse jäävad pildid. Kasutatakse PostgreSQL poolt pakutavat geomeetria tüüpe ja funktsioone. Geomeetria tüübid ei arvesta maa kumerusega ning võib tekkida vajadus üle minna PostGIS laienduse kasutamisele.

Antud otsing eeldab, et piltidele (või videotele) on vastavusse seatud koordinaadid. Veel eeldab selline otsing, et pildid on tehtud maast. Kuna eksisteerib võimalus, kus pildil on vaid osa maast või maa täiesti puudub, siis peab olema eraldi võimalus valida kas otsingus on ka (või ainult) pildid kus on kosmos.

Kasutajaliides (Lisa 4) on minimaalne, et manuaalselt testida põhisüsteemi API't ning piltide filtreerimist. Kasutajaliidesest on puudu kaugseire tulemite detailvaade ja allalaadimise võimalus. Esitatud info on minimaalne - oleks hea juba listivaates esitada olulisemad meta andmed. Veel on eeldatud, et satelliit teeb RGB ja NIR pildi alati koos, aga tõenäoliselt on kaamerad eraldi kontrollitavad, mis võimaldab ka vaid ühe pildi olemasolu.

Peatükis 2 esitatud nõuetest on suurem osa täitmata. Antud töös on keskendutud kasutajaliidese API väljatöötamisele ja funktsionaalsete Angular'i komponentide loomisele (filtreerimiseks kasutatav kaart). Edasi oleks vaja viia olemasolev kasutajaliides nõuetele vastavusse.

6 Kokkuvõte

Töö eesmärk on saada pilt testkaameraga ning välja mõelda ja realiseerida pildi teekond raadiost kasutajani. Teekonnale jäävad pildi analüüs ning salvestamine.

Kaameras, mis pildistab inimesele nähtavas valguse sagedusvahemikus, on sensor MT9P031, mis väljastab Bayer'i mustri. Saamaks sellest pilti on vaja viia läbi interpolatsioon puuduvate piksli väärtuste leidmiseks. Antud töös on kasutatud OpenCV teegi lahendust.

Interpolatsiooni läbiviimiseks on vaja lisaks mustrile tagastada tehtud pildi resolutsioon, baitide arv värvi kohta ja pildistatud kaadrite arv. Satelliidilt on oodatud tagasi ka asukoht või pildistamise ajatempel, et arvutada maajaamas satelliidi asukoht ning on vaja teada ka satelliidi asimuuti ja *off-nadir* nurka.

Testkaameraga korraliku pildi tegemisele läks ootamatult kaua aega. Tulemuseks on kõikide kaameraga pildistamise nõuete täitmine. Esile on toodud käsud, millega saab teha pilte, aga kaamera sensori parameetrite muutmiseks on loodud eraldi programm, sest olemasolev programm on tasuline ja ei õigusta selle projekti raames ostu.

Kaugseire põhitulem (pilt, video) salvestatakse Minio objektipõhisesse salvestussüsteemi ning tulemiga kaasasolevad meta andmed salvestatakse PostgreSQL andmebaasi. Otsuse põhjuseks on põhitulemi suur failisuurus muudaks faili pärimise andmebaasist aeglaseks, aga väiksemad meta andmete peal on andmebaasis hea teha päringuid.

Põhianalüüs, mida kaugseire tulemitel on vaja läbi viia on tulemite vastavusse viimine geograafiliste koordinaatidega. Projekti käigus osutus antud ülesande automatiseerimine liiga keerukaks. Parim tulemus on saavutatud manuaalselt QGIS programmiga, aga isegi siis on tulemus muundunud võrreldes originaaliga.

Kaugseire tulemite kasutajale esitamise osas on tehtud esimene versioon kasutajaliidesest. Loodud kasutajaliidese on aegunud ja ei täida kõiki nõudeid ning vajab täiendamist või ümbertegemist.

Kasutatud kirjandus

[1] „TTÜ100 Satelliidi tutvustus,“ [Võrgumaterjal]. Available: https://www.ttu.ee/projektid/mektory-est/satelliidiprogramm-4/satelliidiprogramm/. [Kasutatud 15 Mai 2018].

[2] S. Romanov, *Kuupsateliidi missioonijuhtimistarkvara arhitektuur,* Tallinn: Tallinna Tehnikaülikool, 2017.

[3] „MT9P031 datasheet,“ [Võrgumaterjal]. Available: https://www.mouser.ee/datasheet/2/308/MT9P031-D-1103275.pdf. [Kasutatud 9 Mai 2018].

[4] „Getting Started Guide for DM368 DM365 LeopardBoard,“ [Võrgumaterjal]. Available: https://developer.ridgerun.com/wiki/index.php?title=Getting_Started_Guide_for_DM368_DM365_LeopardBoard. [Kasutatud 10 Mai 2018].

[5] „RidgeRun Irazu SDK User Guide,“ [Võrgumaterjal]. Available: http://developer.ridgerun.com/wiki/index.php?title=RidgeRun_Irazu_SDK_User_Guide. [Kasutatud 10 Mai 2018].

[6] „Code Sourcery ARM toolchain 2009q1-203,“ [Võrgumaterjal]. Available: https://developer.ridgerun.com/wiki/index.php?title=Code_Sourcery_ARM_toolchain_2009q1-203. [Kasutatud 10 Mai 2018].

[7] „Getting Started Guide for Leopard Board DM365 and DM368,“ [Võrgumaterjal]. Available: http://processors.wiki.ti.com/index.php/Getting_Started_Guide_for_Leopard_Board_DM365_and_DM368. [Kasutatud 10 Mai 2018].

[8] „Linux Digital Video Software Development Kit (DVSDK) for DM365, DM368 Digital Media Processors,“ [Võrgumaterjal]. Available: http://www.ti.com/tool/linuxdvsdk-dm36x. [Kasutatud 10 Mai 2018].

[9] „LeopardBoard 368 Hardware Guide Rev. 1.0,“ 2011.

[10] „GStreameri dokumentatsioon: gst-launch-1.0,“ [Võrgumaterjal]. Available: https://gstreamer.freedesktop.org/documentation/tools/gst-launch.html#pipeline-examples. [Kasutatud 10 Mai 2018].

[11] „GStreameri dokumentatsioon: v4l2src,“ [Võrgumaterjal]. Available: https://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-good/html/gst-plugins-good-plugins-v4l2src.html. [Kasutatud 10 MaI 2018].

[12] „LeopardBoard 368 365 GStreamer Pipelines - Legacy SDK,“ [Võrgumaterjal]. Available: https://developer.ridgerun.com/wiki/index.php?title=LeopardBoard_368_365_GStreamer_Pipelines_-_Legacy_SDK. [Kasutatud 10 Mai 2018].

[13] „Capturing JPEG images at full (allowed) resolution on DM36x platform,“ [Võrgumaterjal]. Available: https://developer.ridgerun.com/wiki/index.php/Capturing_JPEG_images_at_full_(allowed)_resolution_on_DM36x_platform. [Kasutatud 10 Mai 2018].

[14] „Dmaiaccel GStreamer element,“ [Võrgumaterjal]. Available: https://developer.ridgerun.com/wiki/index.php/Dmaiaccel_GStreamer_element. [Kasutatud 10 Mai 2018].

[15] „Minio Erasure Code Quickstart Guide,“ [Võrgumaterjal]. Available: https://docs.minio.io/docs/minio-erasure-code-quickstart-guide. [Kasutatud 20 mai 2018].

[16] „Distributed Minio Quickstart Guide,“ [Võrgumaterjal]. Available: https://docs.minio.io/docs/distributed-minio-quickstart-guide. [Kasutatud 20 mai 2018].

[17] „No logging from Docker container,“ 10 november 2016. [Võrgumaterjal]. Available: https://github.com/minio/minio/issues/3241. [Kasutatud 20 mai 2018].

[18] „How to secure access to Minio server with TLS,“ [Võrgumaterjal]. Available: https://docs.minio.io/docs/how-to-secure-access-to-minio-server-with-tls. [Kasutatud 20 mai 2018].

[19] „Secure TCP/IP Connections with SSL,“ [Võrgumaterjal]. Available: https://www.postgresql.org/docs/10/static/ssl-tcp.html. [Kasutatud 20 mai 2018].

[20] C. Bradford, „Storage Wars: File vs Block vs Object Storage,“ 30 jaanuar 2017. [Võrgumaterjal]. Available: https://blog.storagecraft.com/object-storage-systems/. [Kasutatud 20 mai 2018].

[21] „Minio Bucket Notification Guide,“ [Võrgumaterjal]. Available: https://docs.minio.io/docs/minio-bucket-notification-guide. [Kasutatud 20 mai 2018].

[22] J. Martin, „Difference between Object Storage And File Storage,“ [Võrgumaterjal]. Available: https://stackoverflow.com/a/47524241/8258936. [Kasutatud 20 mai 2018].

[23] „GIS Dictionary: georeferencing,“ [Võrgumaterjal]. Available: https://support.esri.com/en/other-resources/gis-dictionary/term/georeferencing. [Kasutatud 10 Mai 2018].

[24] „World Geodetic System 1984 (WGS84),“ [Võrgumaterjal]. Available: https://confluence.qps.nl/qinsy/en/world-geodetic-system-1984-wgs84-29855173.html. [Kasutatud 10 Mai 2018].

[25] „Google Maps Global Mercator -- Spherical Mercator (unofficial - used in open source projects / OSGEO),“ [Võrgumaterjal]. Available: https://epsg.io/900913. [Kasutatud 10 Mai 2018].

[26] M. Corey, „Choosing the Right Map Projection,“ 20 Detsember 2013. [Võrgumaterjal]. Available: https://source.opennews.org/articles/choosing-right-map-projection/. [Kasutatud 10 Mai 2018].

[27] „KAARDI MATEMAATILINE ALUS,“ [Võrgumaterjal]. Available: https://www.ttu.ee/public/e/ehitusteaduskond/Instituudid/Teedeinstituut/Geodeesia_oppetool/oppematerjalid/Kaardiprojektsioonid20II.pdf. [Kasutatud 10 Mai 2018].

[28] „Affine Transformations,“ [Võrgumaterjal]. Available: http://www.quantdec.com/GIS/affine.htm. [Kasutatud 10 Mai 2018].

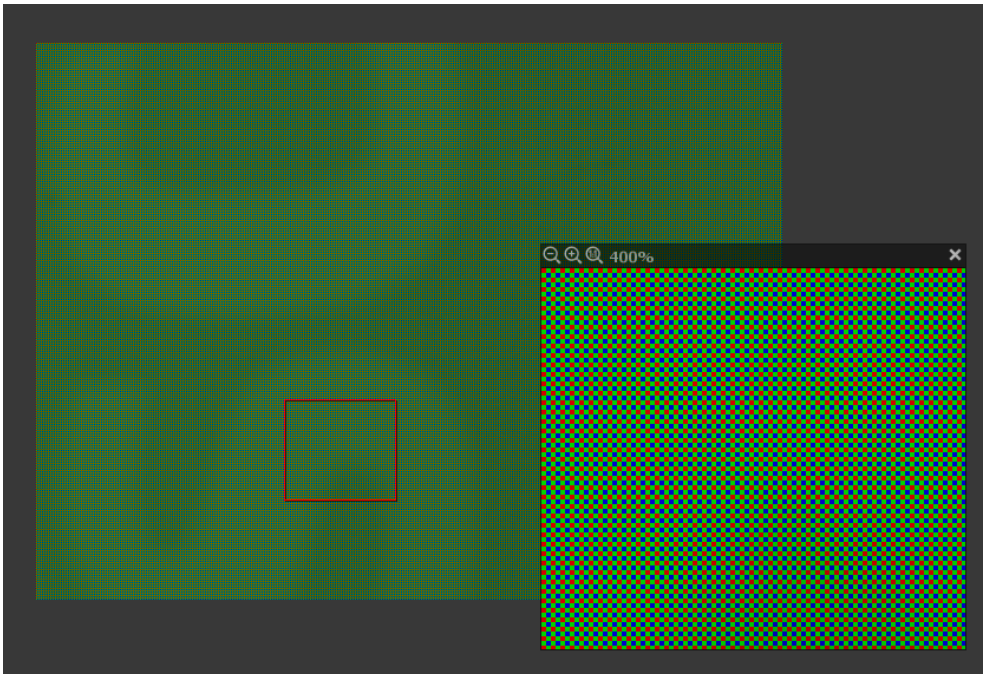
[29] „Java dokumendatsioon: Class AffineTransform,“ [Võrgumaterjal]. Available: https://docs.oracle.com/javase/7/docs/api/java/awt/geom/AffineTransform.html. [Kasutatud 10 Mai 2018].

[30] „AffineTransformation Tutorial,“ [Võrgumaterjal]. Available: http://docs.geotools.org/stable/userguide/tutorial/affinetransform.html. [Kasutatud 10 Mai 2018].

[31] S. Bagon, „Given Three Points Compute Affine Transformation,“ [Võrgumaterjal]. Available: https://stackoverflow.com/questions/22954239/given-three-points-compute-affine-transformation. [Kasutatud 10 Mai 2018].

Lisa 1 - *demosaicking*

Interpolatsiooni käigus tehakse Bayer'i muster (Joonis 26) korralikuks pildiks (Joonis 27).



Joonis 26. Bayer'i mustrina esitatav pilt kasutades RawTherapee 5.4 vahendit

Joonis 27. OpenCV-ga interpoleeritud pilt

Lisa 2 – convertToRGB.py

```
import cv2
import numpy as np

WIDTH = 640
HEIGHT = 480
BUFFERS = 1
file_path = "PATH TO RAW FILE"
output_path = "PATH TO OUTPUT.png"

raw_img_arr = np.fromfile(file_path, dtype=np.uint8).reshape((BUFFERS,
HEIGHT, WIDTH))
for i in range(BUFFERS):
    raw_img = raw_img_arr[i]
    img = cv2.cvtColor(raw_img, cv2.COLOR_BAYER_GB2RGB)
    cv2.imshow("frame {}".format(i), img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
cv2.imwrite(output_path, img)
```

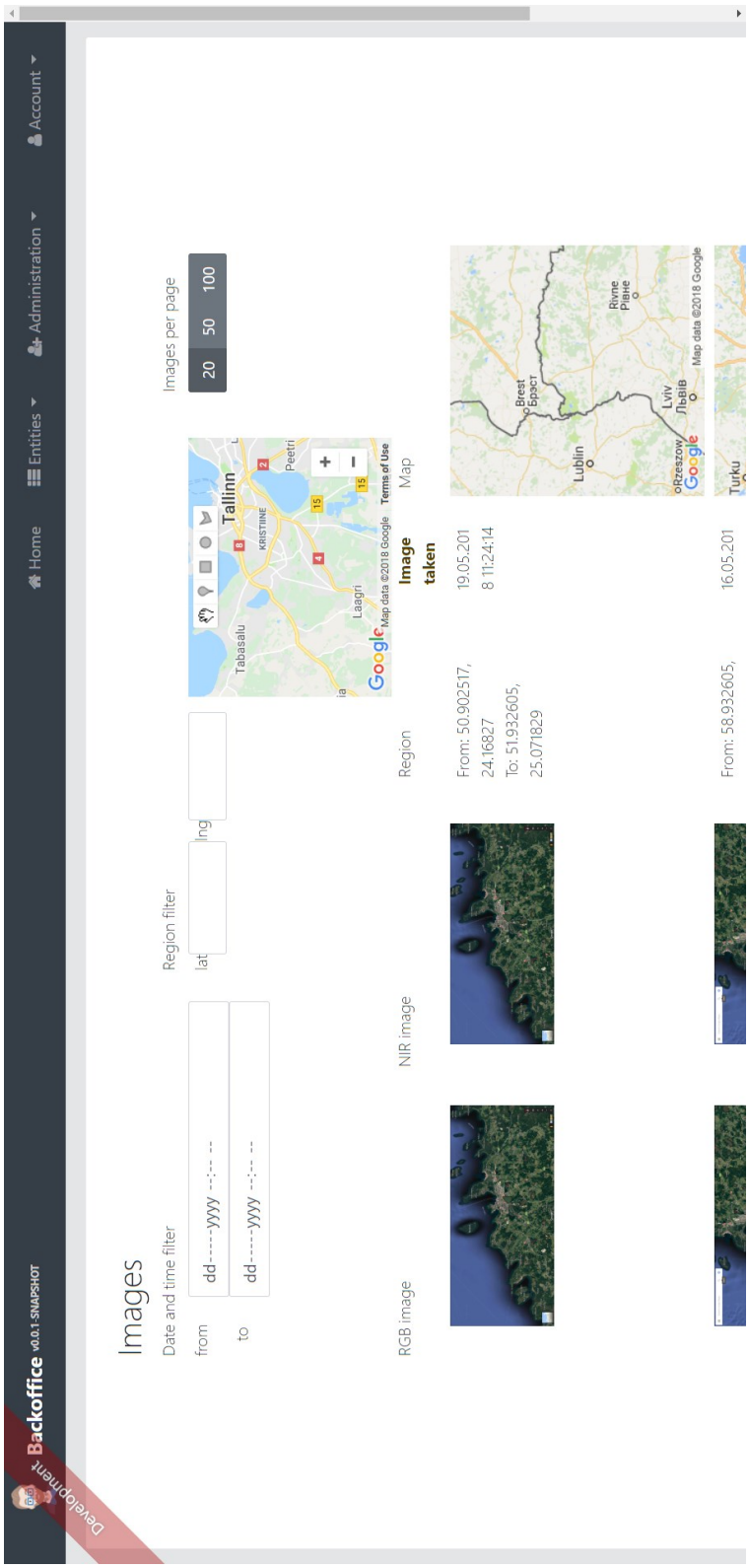
Lisa 3 – streamServer.py

```
# based on https://pymotw.com/3/socket/tcp.html
import cv2
import numpy as np
import socket

PORT = 8500
WIDTH = 640
HEIGHT = 480
sock = socket.socket(socket.AF_INET,
                     socket.SOCK_STREAM) # UDP - socket.SOCK_DGRAM, TCP -
socket.SOCK_STREAM
sock.bind(("192.168.1.105", PORT))
sock.listen(5)
while True:
    # Wait for a connection
    print('waiting for a connection')
    connection, client_address = sock.accept()
    try:
        print('connection from', client_address)

        frame = None
        while True:
            data = connection.recv(WIDTH * HEIGHT)
            if data:
                if not frame: frame = data
                else: frame += data
                if len(frame) == WIDTH * HEIGHT:
                    raw_img = np.frombuffer(frame, np.uint8, len(frame),
0).reshape(HEIGHT, WIDTH)
                    img = cv2.cvtColor(raw_img, cv2.COLOR_BAYER_GB2RGB)
                    cv2.imshow('', img)
                    if cv2.waitKey(25) & 0xFF == ord('q'):
                        break
                    frame = None
            else:
                print('no data from', client_address)
                break
    finally:
        # Clean up the connection
        connection.close()
```

Lisa 4 - kasutajaliides



Lisa 5 - kasutajaliides regiooni filtreerimisega