TAMPEREEN
AMMATTIKORKEAKOULU

VIISTOVALOKUVAUSLAITTEISTON KUVA-ANALYSOINNIN KÄYTTÖLIITTYMÄ

Anssi Sorsakivi

Opinnäytetyö
Joulukuu 2017
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

SORSAKIVI, ANSSI:
Viistovalokuvauslaitteiston kuva-analysoinnin käyttöliittymä

Opinnäytetyö 62 sivua, joista liitteitä 32 sivua
Joulukuu 2017

Opinnäytetyöni tavoite oli luoda Tampereen ammattikorkeakoulun viistovalokuvauslaitteen kuvien analysointia tukeva käyttöliittymä, sekä päivittää jo olemassa olevan näytteiden kuvaamiseen tehty käyttöliittymä. Kuva-analysoinnin käyttöliittymä päätettiin toteuttaa MATLAB App Designer -ohjelmalla. Ohjelman toteutukseen käytettiin MATLAB-kieltä.

Tavoitteena oli myös testata viistovalokuvauslaitteiston soveltuvuutta paperi- ja kartonkipintojen sileysmittaukseen. Sileysmittausten suorittamiseen käytettiin viistovalokuvausmenetelmää sekä ilmanläpäisevyysmenetelmään perustuvaa sileysmittaria. Viistovalokuvausmenetelmällä tuotetut virtuaaliset pinnat arvioitiin silmämääräisesti.

Analysointia tukeva käyttöliittymä saatiin luotua tavoitteiden mukaisesti. Molemmat viistovalokuvauslaitteiston käyttöliittymät toimivat suunnitellusti, eli niillä on mahdollista suorittaa viistovalokuvauksen kaikki vaiheet. Sileysmittausten tulokset olivat samansuuntaiset molemmilla mittausmenetelmillä.

Viistovalokuvauslaitteistolla on selkeää potentiaalia tuottaa luotettavia tuloksia sileysmittauksessa. Kuvien analysoinnin lähdekoodiin olisi mahdollista kirjoittaa algoritmi joka laskisi sileydelle arvon suoraan analysoinnissa tuotetusta raakadatasta. Tämän lisäksi virtuaalisen pinnan z-akselin arvojen kalibrointi metrisen järjestelmän mukaiseksi helpottaisi tulosten tarkastelua.

Analysoinnin käyttöliittymää on mahdollista kehittää edelleen. Käyttäjälle voisi luoda enemmän mahdollisuuksia vaikuttaa analysoinnin kulkuun. Käyttäjälle voisi luoda mahdollisuuden vaikuttaa siihen mitä kuvaajia analyysin aikana piirretään. Raakadatan tallennuksen perusteellinen suunnittelu ja toteutus nopeuttaisi tulosten käsittelyä ja säästäisi muistia.

Asiasanat: viistovalokuvaus, käyttöliittymä, pinta, sileys

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

SORSAKIVI, ANSSI:
Graphical User Interface for Image Analysis in Photometric Stereo Equipment

Bachelor's thesis 62 pages, appendices 32 pages
December 2017

The main objective of this thesis was to create a graphical user interface (GUI) to support the image analysis in the Tampere University of Applied Sciences` (TAMK) photometric stereo equipment. A goal of updating the already existing GUI designed to operate the camera and light controller was also set.

In this thesis, the suitability of the photometric stereo equipment in measuring paper and cardboard surface smoothness was tested. Air permeability method was used as a reference to photometric stereo method. The virtual surfaces created with the photometric stereo equipment were assessed visually.

The GUI was created successfully and it was integrated in the photometric stereo system. Both of the smoothness measurement methods gave similar results.

The photometric stereo equipment in TAMK shows clear potential in producing reliable results in measuring the smoothness of paper and cardboard surfaces. To produce more precise smoothness values it is possible to write an algorithm that uses the raw data of the surface analysis. Furthermore the results would be easier to determine if the z-axis of the virtual surfaces were calibrated according to the metric system.

It is possible to continue the development of the image analysis GUI. It would be useful to create the necessary controls for selecting which figures are drawn during the image analysis. Also, carefully planning the saving and reuse of the raw data would save time and memory space in the analysing process.

Key words: photometric stereo, graphical user interface, surface, smoothness

SISÄLLYS

LYHENTEET JA TERMIT

CV	coefficient of variation, suhteellinen keskihajonta
Konenäkö	sovellus joka sisältää kameran, valonlähteen sekä kameran tuottaman datan analysoimiseen tarkoitetun ohjelmiston
MFC	Microsoft Foundation Class -ohjelmointikirjasto
SDK	Software developement kit, ohjelmiston kehittäjille tarkoitettu tieto- ja ohjelmistopaketti
VVKL	viistovalokuvauslaite
WF	wood free, eli puuvapaa paperi valmistetaan ilman mekaanisesti jauhettua massaa

1 JOHDANTO

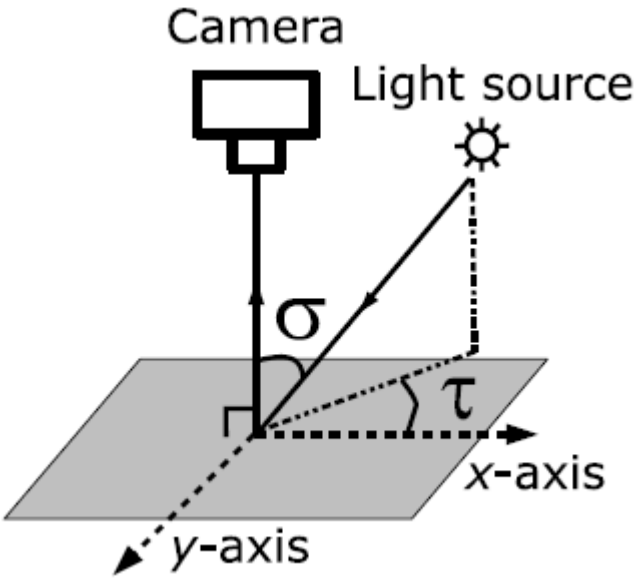
Tämän opinnäytetyön tavoite oli luoda kuvien analysoimiseen tarkoitettu käyttöliittymä nimeltä Analyzer Tampereen ammattikorkeakoulussa (TAMK) kehitettyyn viistovalokuvauslaitteeseen (VVKL), sekä parantaa olemassa olevan VVKL-käyttöliittymän toiminnallisuutta. TAMKin viistovalokuvauslaitteiston suunnittelu ja toteutus aloitettiin yhteistyössä Tampereen teknillisen yliopiston (TTY), Hämeen ammattikorkeakoulun sekä Valmet Automation Oy:n kanssa. Opinnäytetyön tilaaja on Tampereen ammattikorkeakoulu, paperi- ja pakkauslaboratorio. Alkuperäisenä ajatuksena oli, että TAMKissa valmistettaisiin TTY:llä kehitettyjä tekniikoita hyväksi käyttäen kaupallinen versio viistovalokuvauslaitteesta teollisuuden tarpeisiin.

Analyzer-käyttöliittymän toteuttamiseen käytettiin MATLAB App Designer -ohjelmaa, joka on applikaatioiden tekemiseen suunniteltu ohjelmointiympäristö. Analyzer on järkevintä toteuttaa App Designer -ohjelmalla koska MATLAB-kirjastot sisältävät erinomaiset kuvien analysoimiseen tarkoitetut funktiot. Päätöstä tukee myös se, että TTY:llä kehitetyt analysointialgoritmit, sekä Eetu Mastosalon opinnäytetyössä kirjoittamat analysoinnin skriptit ovat kirjoitettu pääosin MATLAB-kielellä.

Viistovalokuvauslaitteen tuottamista tuloksista tehtiin vertailu yleisessä käytössä olevan sileysmittarin tuottamien tulosten kanssa. Tavoitteena oli saada käsitys siitä, kuinka hyvin VVKL soveltuu sileysmittaukseen.

2 VIISTOVALOKUVAUS

TAMKin viistovalokuvauslaitteisto hyödyntää menetelmää nimeltä viistovalokuvaus, jolla on mahdollista analysoida kohteen pinnan topografiaa valokuvien avulla. Viistovalikuvauksessa kuvat otetaan tyypillisesti suoraan kohteen yläpuolelta heijastusten vähentämiseksi (kuvio 1). Jokaisen kuvan yhteydessä kohdetta valaistaan eri suunnasta, joka aiheuttaa varjoja kohteen pintaan. Varjojen avulla voidaan päätellä mitkä kohdat ovat matalammalla eli pimennossa olevia kohtia.



KUVIO 1. Viistovalokuvaustilanne (Kuparinen, 2008)

2.1 Fotometrinen stereo

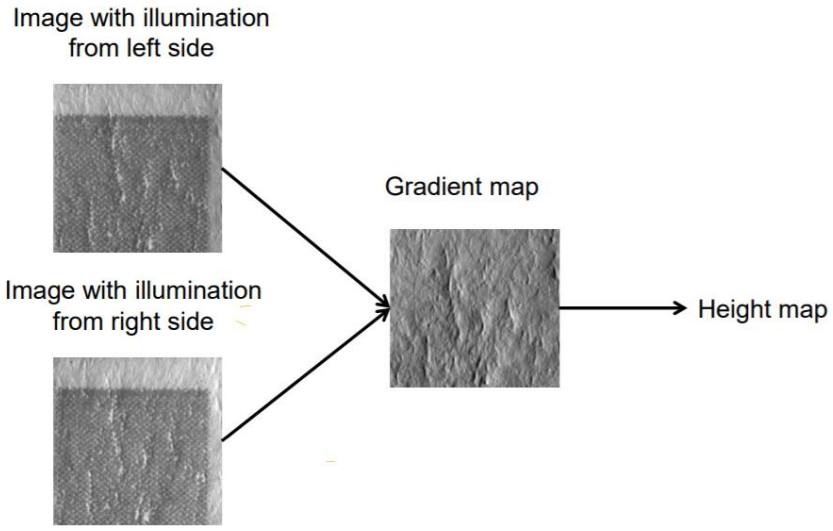
Vuonna 1978 Robert J. Woodham loi menetelmän nimeltä fotometrinen stereo, joka perustui kolmeen valonlähteeseen (Kuparinen, 2008 s. 48). TAMKin viistovalokuvauslaitteen kuva-analysoinnissa käytetään neljään valonlähteeseen perustuvaa versiota kyseisestä menetelmästä. Menetelmän ovat kehitettäneet tutkijatohtori Jukka-Pekka Raunio sekä tutkijatohtori Marja Mettänen Tampereen teknillisessä yliopistossa.

Näytteestä otetuista kuvista lasketut gradientit voidaan integroida integrointimenetelmällä virtuaaliseksi pinnaksi (Kuparinen, 2008 ss. 52–53). Kuparisen mukaan analysoinnissa oletetaan, että kuvattava kohde heijastaa valoa valaisukulmaa vastaavalla intensi-

teetillä Lambertin-lain mukaisesti. Käytännössä tämä tarkoittaa sitä, että analysointiin soveltuvat esimerkiksi mattapintaiset paperit ja -maalit sekä puuvillakankaat (Kuparinen, 2008 s. 29). Eetu Mastosalo on tehnyt opinnäytetyön TAMKin kuva-analysoinnin käyttöönotosta, jossa on käyty analysoinnin prosessit tarkemmin läpi (Mastosalo, 2017).

2.2 Kaupalliset sovellukset

Markkinoilta löytyy jo muutama viistovalokuvaustekniikkaa hyödyntävä kaupallinen sovellus. Innventia markkinoi kehittämäänsä OptiTopo -laitteistoa paperi- ja kartonki pintojen analysoimiseen. Näytteen pintaa analysoidaan, ottamalla siitä kaksi kuvaa joista rakennetaan pinnalle korkeuskartta (kuvio 2). Tulosten perusteella voidaan arvioida pinnan soveltuvuutta printtaukseen. Korkeuskartasta pystytään myös tunnistamaan pinnan virheitä. (Innventia)



KUVIO 2. OptiTopo-laitteiston kuva-analysoinnin toimintaperiaate (Innventia)

MVTec markkinoi HALCON-ohjelmistoa, joka on moniuolinen konenäkösovellusten kehitysalusta. HALCON-ohjelmistoon on sisäänrakennettu monia pinnan analysoimiseen tarkoitettuja funktioita. Ohjelmistosta löytyy myös funktiot viistovalokuvaustekniikassa käytettäviin laskuihin jotka mahdollistavat viistovalokuvaukseen perustuvan kuva-analysoinnin toteuttamisen. (MVTec)

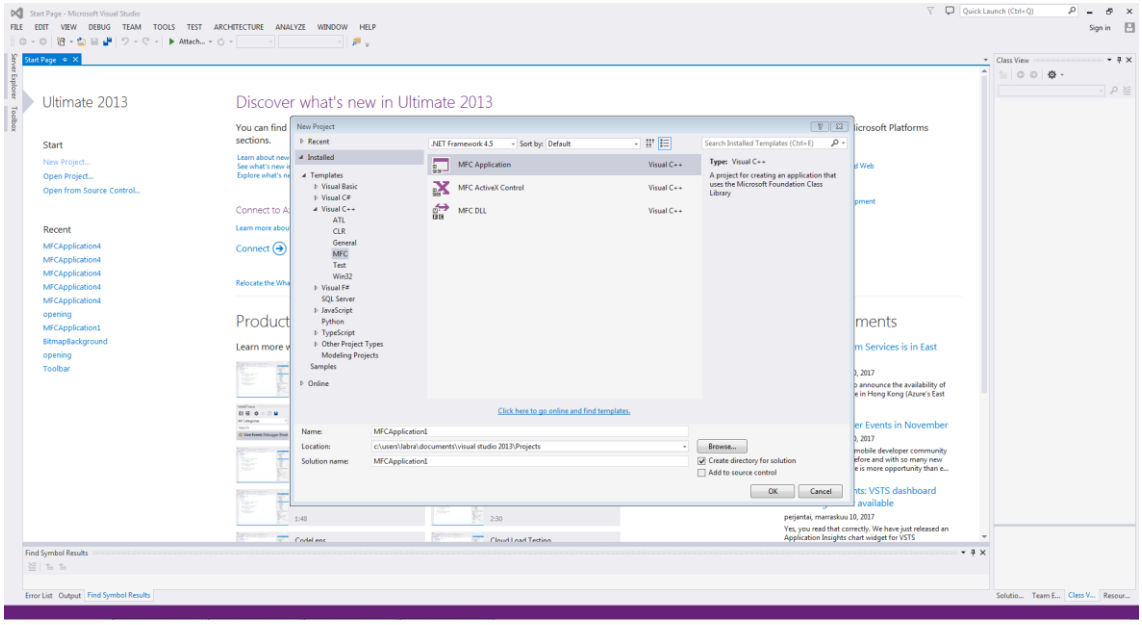
3 OHJELMOINTIYMPÄRISTÖT

Viistovalokuvauslaitteen käyttöliittymä sekä analysoinnin käyttöliittymä on toteutettu tietokoneella, johon on asennettu Microsoft Windows (7 Enterprise, Service Pack 1, versio 6.1.7601) -käyttöjärjestelmä. Erityisesti VVKL-käyttöliittymä tarvitsee Windows käyttöjärjestelmän toimiakseen, sillä VVKL-käyttöliittymässä käytetään Microsoft Foundation Class (MFC) -luokkia jotka ovat tunnusomaisia Microsoftin Windows järjestelmille.

VVKL-käyttöliittymä on toteutettu Microsoft Visual Studio (Ultimate 2013, versio 12.0.31101.00, Update 4) -ohjelmalla, joka on ohjelmistojen suunnitteluun ja toteutukseen tarkoitettu ohjelma (kuva 1). Analyzer puolestaan on kirjoitettu kokonaan ilman Windows spesifisiä kirjastoja, MATLAB (R2017a, versio 9.2.0.538062) -ohjelmalla. Analyzerin toteutukseen käytettiin käyttöliittymien kehitysympäristöä nimeltä MATLAB App Designer. MATLAB-ohjelmaan asennettiin Image Processing Toolbox -kirjasto, jota hyödynnetään kuvien analysoimisessa. Kumpaakaan käyttöliittymää ei ole testattu muilla käyttöjärjestelmillä kuin Windowsilla.

3.1 Microsoft Visual Studio

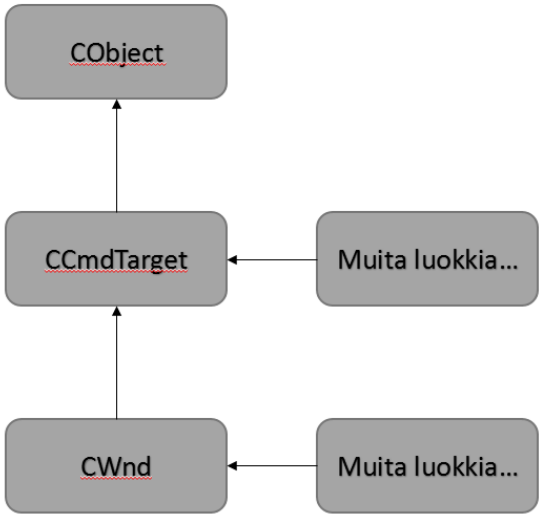
Visual Studio on ohjelmisto joka pitää sisällään ohjelmistojen kehittämiseen suunniteltuja työkaluja. Työkaluihin kuuluu esimerkiksi editori, joka on lähdekoodin muokkaamiseen tarkoitettu käyttöliittymä. Ohjelmistoon kuuluu myös kääntäjät, linkkerit ja debuggeri. (Bjönander, 2008 s. 114)



KUVA 1. Visual Studio -editori

Microsoft Foundation Class -kirjasto

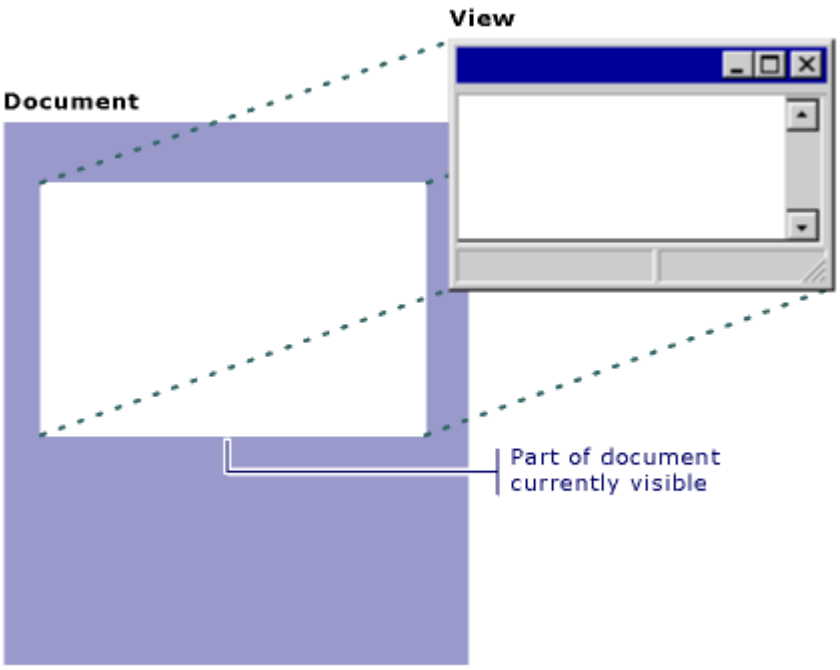
Visual Studio hyödyntää applikaatioiden teossa Windows API -kirjastoa, joka sisältää C-kielellä kirjoitettuja funktioita. Nykyään ohjelmistojen kehittämiseen käytetään kuitenkin usein C++ -kieltä. Tätä varten on kehitetty MFC-kirjasto, jonka avulla Windows API -kirjaston funktioiden toiminnallisuus saadaan käyttöön C++ -kielellä. MFC-kirjasto on rakennettu pääosin luokkahierarkian mukaisesti. Luokkahierarkian huipulla on CObject-luokka (kuvio 3). (Bjönander, 2008 s. 115)



KUVIO 3. MFC-luokkahierarkia (Microsoft, 2015b)

MFC-kirjasto tarjoaa valmiin rajapinnan Windows-ikkunoiden luomiseen ja Windows-viestien käsittelyyn. MFC-applikaation luominen tapahtuu helpoiten käyttämällä MFC Application Wizard -työkalua. Työkalu generoi automaattisesti suurehkon määrän koodia käyttäjälle. MFC-projektin voi luoda manuaalisestikin, itse kirjoittamalla tarvitsemansa koodit, mutta se on huomattavasti työläämpi vaihtoehto. Käytännössä projektista tulee MFC-projekti kun lähdekoodissa käytetään jotain MFC-luokkaa.

MFC-projekteissa hyödynnetään Document/View -arkkitehtuuria. Tämä tarkoittaa sitä, että MFC Application Wizard generoi käyttäjälle dokumentti-luokan sekä näkymä-luokan. Dokumentti-luokkaan varastoidaan kaikki data. Tämän datan esittämiseen ja käyttäjälähtöiseen muuttamiseen käytetään näkymä-luokkaa (kuvio 4). Dokumenttiin viittaavia näkymiä voidaan luoda useita samanaikaisesti. (Microsoft, 2015a)

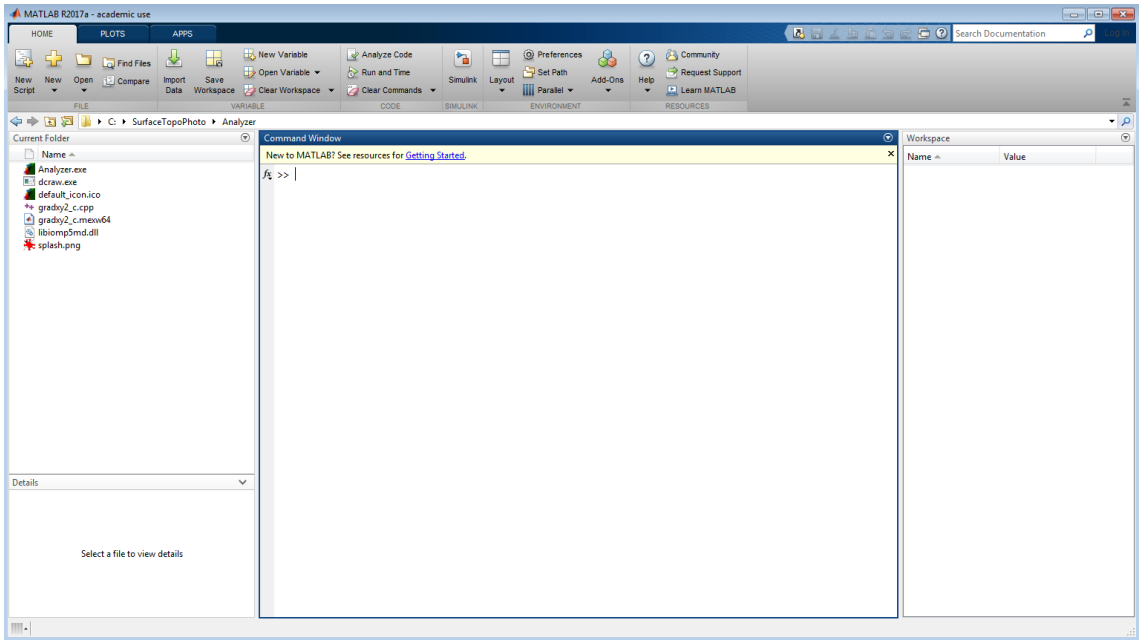


KUVIO 4. Dokumentti/Näkymä –malli (Microsoft, 2015a)

3.2 MATLAB-ohjelmisto

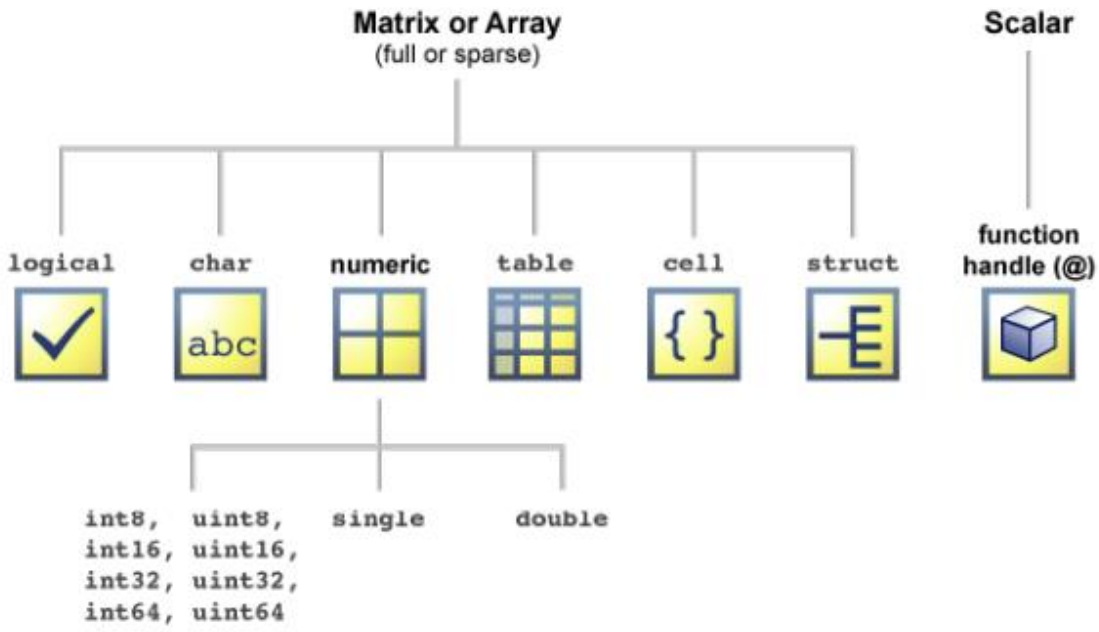
MATLAB on ohjelmisto joka pitää sisällään ohjelmistojen kehittämiseen suunniteltuja työkaluja. Työkaluihin kuuluu esimerkiksi GUIDE-editori, joka on lähdekoodin muokkaamiseen tarkoitettu käyttöliittymä (kuva 2). Ohjelmistoon kuuluu myös kääntäjät, link-

kerit ja debuggeri sekä käyttöliittymien työstöön suunniteltu App Designer -käyttöliittymä. Ohjelmistoon on mahdollista ladata Toolbox-lisäosia, jotka osaltaan lisäävät ohjelmiston toiminnallisuutta.



KUVA 2. MATLAB GUIDE -editori

MATLAB tarjoaa valmiit luokat, joita käyttäjä voi hyödyntää lähdekoodissaan erilaisten muuttujien luomiseen (kuvio 5). Lähdekoodi kirjoitetaan MATLAB-kielellä, joka on laskentaan suunniteltu matriisikieli (MathWorks ss. 41–45). Matriisikieli tarkoittaa sitä, että kaikki muuttujat ovat matriisimuotoisia, eli niihin voidaan sijoittaa useita arvoja. Skriptit tallennetaan muotoon ”.m”, ja App Designerilla luodut käyttöliittymät tallennetaan muotoon ”.mlapp”, jotka ovat MATLAB-ohjelmalle ominaisia tiedostoformaatteja.

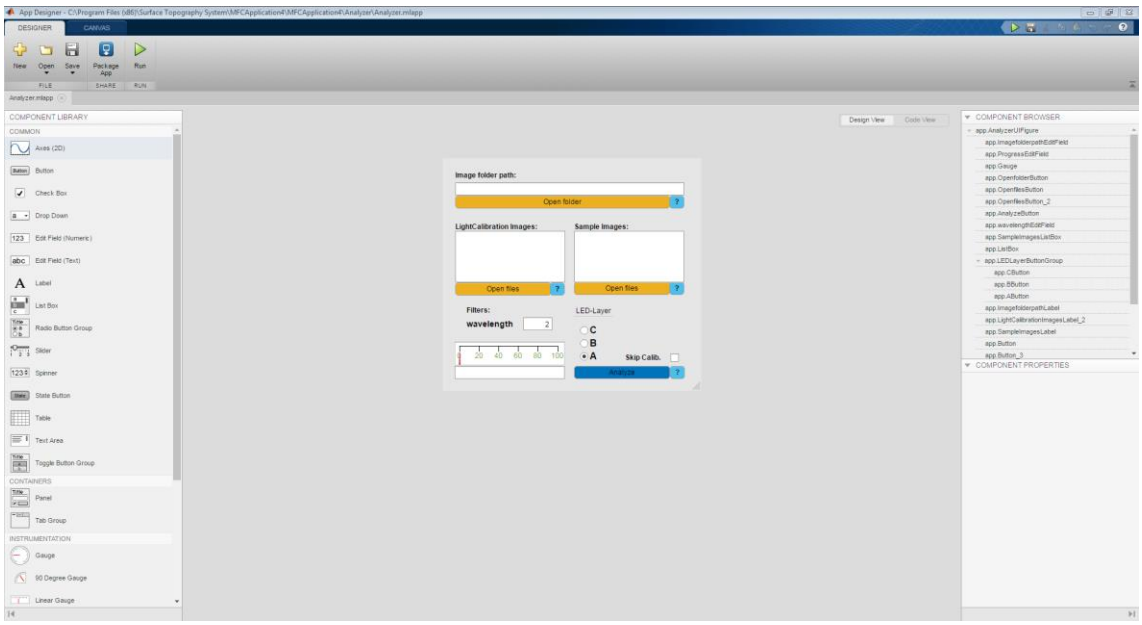


KUVIO 5. MATLAB-luokat (MathWorks s. 90)

App Designer -työkalu

App Designer on käyttöliittymä, joka on jaettu kahteen näkymään. Design View -näkymä mahdollistaa komponenttien sijoittelun suunniteltavaan käyttöliittymään (kuva 3). Komponentit ovat sijoitettu listaan vasemmalle, josta käyttäjä voi valita tarvitsemansa komponentin ja raahata sen käyttöliittymäalueelle näkymän keskelle. Oikeaan reunaan on sijoitettu lista jo asetetuista komponenteista.

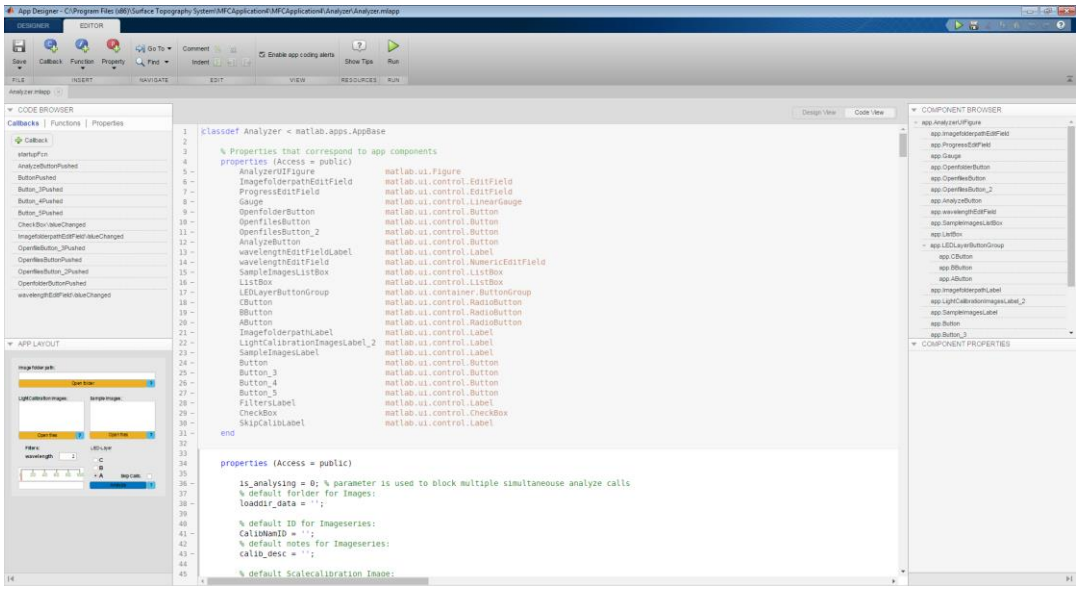
Komponenttilistasta on helppo lisätä callback-funktio kyseiselle komponentille. Callback-funktio generoidaan automaattisesti lähdekoodiin ja se ajetaan kun kyseistä komponenttia käytetään ajon aikana. Heti lisättyjen komponenttien vasemmalla puolella on nappi, josta voi vaihtaa näkymän Code View-näkymään.



KUVA 3. Design View -näkymä

Code View -näkymä mahdollistaa lähdekoodin muokkaamisen (kuva 4). Keskelle näkymää on sijoitettu käyttöliittymän lähdekoodi. Ainoastaan osa lähdekoodista on käyttäjän muokattavissa. Käyttäjä ei pysty muokkaamaan lisättyjen komponenttien deklaraatioita, callback-funktioiden deklaraatioita eikä itse applikaation alustus- ja lopetuskomentoja.

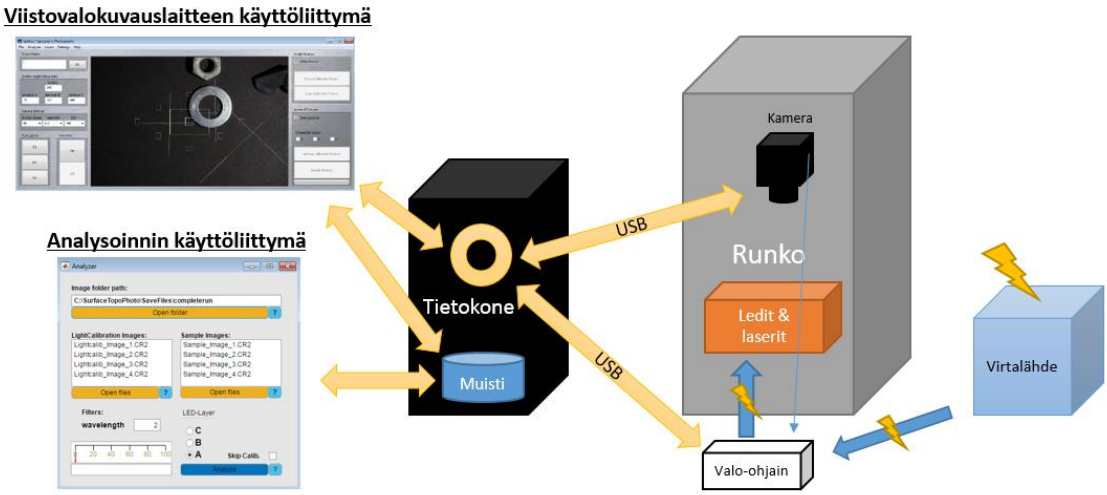
Näkymän vasempaan laitaan on sijoitettu lista luoduista callback-funktioista sekä pieni kuvake käyttöliittymän näkymästä. Oikeassa laidassa on sama komponenttilista kuin Design View -näkymässä. Myös näkymänvaihtopainikkeet ovat samassa paikassa kuin Design View -näkymässä.



KUVA 4. Code View -näkymä

4 VIISTOVALOKUVAUSLAITTEISTO

TAMKin viistovalokuvauslaitteisto koostuu alumiinisesta rungosta, valo-ohjaimesta, virtalähteestä sekä tietokoneesta, johon on asennettu ohjaukseen tarkoitetut käyttöliittymät (kuvio 6). Alumiinirunko pitää sisällään kameran, ledit sekä laserit.



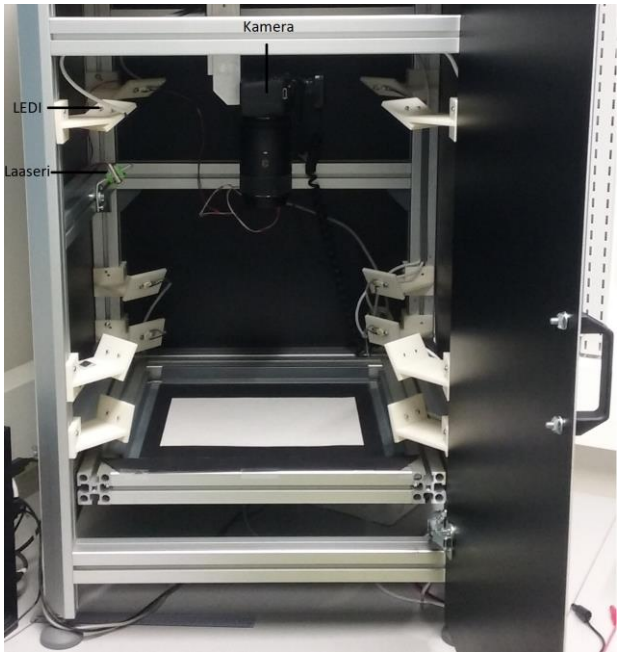
KUVIO 6. TAMKin viistovalokuvauslaitteisto

Viistovalokuvauslaitteen alumiinirungon suunnitteli ja kasasi Petri Nurminen TAMKissa vuonna 2014, laboratoriomestari Juhani Pitkäsen avustuksella. Alumiinirungossa on neljä kulmapylvästä, irrotettava kasetti eli näytealusta, ovi sekä irrotettavat seinäpaneelit ja katto.

Alumiinirungon kulmapylväisiin on sijoitettu kiskot. Jokaisen pylvään kiskoon on sijoitettu kolme kiinnikettä joita voi vapaasti liikutella pystysuunnassa kiskoa pitkin. Kiinnikeisiin on sijoitettu BXRA-40E0810-A -malliset ledit (Vuorenmaa, 2015 s. 21). Ledit on asetettu korkeudeltaan siten että alin ledi on kuvausalustan normaaliin nähden 75 asteen kulmassa, keskimmäinen 60 asteen kulmassa ja korkein on 30 asteen kulmassa. Lopputuloksena syntyy kolme eri korkeudella olevaa leditasoa, joissa kussakin on neljä lediä (kuva 5).

Kahteen kulmapylväiden väliseen tukipylvääseen on asetettu kiinnike laseria varten. Lasereita käytetään kameraa tarkennettaessa näytteen pintaan, minkä vuoksi ne osoittavat näytealustan keskikohtaan. Näytealustan yläpuolelle on sijoitettu kiinnike järjestelmäkameraa varten. Kameran kiinnikettä pystyy liikuttamaan pystysuunnassa manuaalisesti.

Laitteiston kamera on Canon EOS 100D -digitaalijärjestelmäkamera, johon on liitetty Sigma 105 mm F/2.8 EX DG OS Macro -teleobjektiivi.



KUVA 5. Viistovalokuvauslaite

Rungon ulkopuolelle ilman erillistä kiinnikettä on sijoitettu valo-ohjain (kuva 6). Valo-ohjain on suunniteltu ledien ja lasereiden ohjaamiseen tekstimuotoisen rajapinnan kautta, vuonna 2015 Panu Vuorenmaan toimesta Tampereen ammattikorkeakoulussa.

KUVA 6. Valo-ohjain

Ohjain on suunniteltu Arduino Nano -mikrokontrollerin ympärille
(Vuorenmaa, 2015 s. 12). Arduino Nano toimii virtalähteen ja viistovalokuvauslaitteen elektronisten komponenttien välissä, ohjaten kuinka paljon ja milloin virtaa annetaan kullekin komponentille.

Ledien kynnysjännite on minimissään 25,3 V ja maksimissaan 30,9 V. Virtalähteen jännite on asetettava vähintään kolme volttia kynnysjännitettä suuremmaksi ja maksimissaan 35 volttiin. Valo-ohjain lukee kameran kengästä signaalin, jonka avulla valo-ohjain osaa ajoittaa ledin sytyttämisen kuvan ottamisen ajaksi. Yhdessä kameran signaalin sekä käyttäjän antaman syötteen perusteella, valo-ohjain osaa hallita kuvauslaitteen valaistusta kuvauksen aikana (taulukko 1). (Vuorenmaa, 2015 ss. 8, 34)

TAULUKKO 1. Valo-ohjaimen rajapinta (Vuorenmaa, 2015 ss. 36, muokattu)

Käsky	Toiminto	Parametrit (xx)
$LSxx!	Asettaa valitun ledin päälle	Ledin kerros sekä numero. Kerros A,B,C ja ledin numero 1-4 (esim. A3)
$SSxx!	Asettaa sekvenssissä käytettävät kerrokset	Aloituskerros sekä lopetuskerros joiden ledejä käytetään A-C. (esim. AB) Voidaan myös asettaa vain yks kerros laittamalla molemmat parametrit samaksi.
$Fxxx!	Tarkennuksessa käytettävien lasereiden ohjaus	SON asettaa laserit päälle, OFF sammuttaa laserin
$MODx!	Kameralta tulevan signaalin asetus	X asettaa laitteen toimimaan kameran X-signaalin mukaan. Q asettaa laitteen toimimaan kameran quench-signaalin mukaan (fyysinen signaalin vaihto tapahtuu kytkimellä laitteen sisältä).

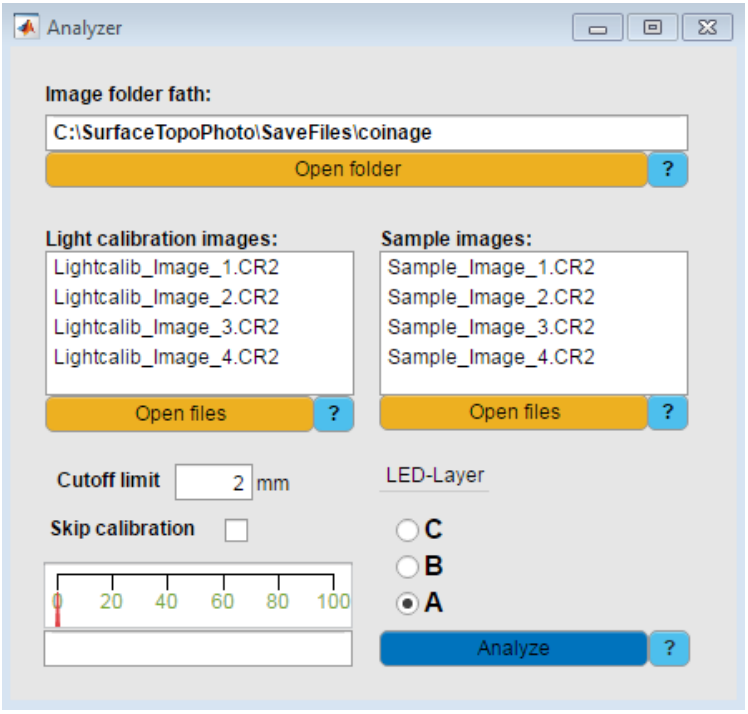
Viistovalokuvauslaitteistoa ohjataan sitä varten suunnitellulla käyttöliittymällä (kuva 7). Käyttöliittymän suunnittelu ja toteutus aloitettiin vuonna 2015 Aleksi Tapolan toimesta. Käyttöliittymä viimeisteltiin käyttövalmiiksi tämän työn tekijän toimesta kesäharjoittelussa 2016. Käyttöliittymä toimii laitteiston käyttäjän ja laitteiston välissä rajapintana, jonka avulla laitteistoa voidaan ohjata helposti ymmärrettävillä ja -käytettävillä kontrolleilla. Sillä pystytään luomaan projekteja, säätämään kameran parametreja, ohjaamaan ledejä ja lasereita sekä ottamaan yksittäisiä kuvia ja kuvasarjoja. Luotuun projektikansioon tallennetaan automaattisesti otetut kuvat, metadata.xml -tiedosto, autosave.xml -tiedosto, sekä HistogramRGB-kansio, johon tallentuu kuvista tuotetut histogrammit.

KUVA 7. Viistovalokuvauslaitteen käyttöliittymä

Käyttöliittymä on asennettu tietokoneeseen, joka on yhteydessä valo-ohjaimeen sekä kameraan erillisten USB-väylien välityksellä. Käyttöliittymä kommunikoi valo-ohjaimen kanssa USB-väylän välityksellä, valo-ohjaimen rajapinnan mukaisesti (taulukko 1). Käyttöliittymä kommunikoi myös kameran kanssa USB-väylä välityksellä, Canon SDK -rajapinnan mukaisesti. Canon SDK pitää sisällään esimerkkiohjelman kameran ohjaamiseksi, jota käytettiin mallina käyttöliittymän luokkarakenteen luomiseen.

Käyttöliittymän tämän hetkinen luokkarakenne on kuvattu liittessä 1. CMFCApplication4Dlg on käyttöliittymän näkymä-luokka, joka on periytetty CDialogEx-, ActionSource- sekä Observer-luokasta. CDialogEx on MFC-luokka, jonka kantaluokka on CObject. ActionSource sekä Observer ovat Canon SDK -luokkia, jotka ovat tarkoitettu kameran ohjaamiseen.

Viistovalokuvauslaitteella otetuista 2D-kuvista voidaan luoda kolmiulotteinen pintatopografia Analyzer-ohjelman avulla (kuva 8). Analyzer-käyttöliittymällä valitaan analysoitavat kuvat, neljä valaistuskalibrointikuvaa ja neljä näytekuvaa, ja käynnistetään kuvien analysointi. Käyttöliittymä alustaa analysoinnissa tarvittavat muuttujat ja ajaa analysoinnin skriptit. Analysoinnin lopuksi piirretään käyttäjälle virtuaalinen pinta joka on integroitu näytekuvista (kuvio 8).



KUVA 8. Kuva-analysoinnin käyttöliittymä Analyzer

5 KUVA-ANALYSOINNIN KÄYTTÖLIITTYMÄN TYÖSTÖ

Kuvien analysointia varten luotiin App Designerilla käyttöliittymä, jonka nimeksi annettiin Analyzer (kuva 8). Tavoitteena oli luoda käyttöliittymä, jolla käyttäjä pystyy valitsemaan haluamansa kuvaprojektikansion ”Windows Explorer”-selaimella. Tämän jälkeen käyttäjällä olisi mahdollisuus valita kuvat kyseisestä projektikansiosta jotka hän haluaa analysoida. Analysointi tapahtuisi Analyze-nappia painamalla, joka ajaisi analysoinnin skriptit kyseisille kuville.

Projekti aloitettiin App Designer -ohjelmalla, generoimalla uusi applikaatioprojekti New-nappia painamalla. App Designer generoi automaattisesti applikaation ja rungon lähdekoodille. Liitteessä 2 on esitetty Analyzer-käyttöliitymän lähdekoodi. Käyttöliittymään lisättiin ylälaitaan tekstikenttä kuvaprojektikansion tiedostopolkua varten. Tekstikentän alapuolelle lisättiin Open Folder -nappi, jota painamalla käyttäjälle avataan Windows Explorer -selain, joka on säädetty kansion valitsemiseen. Valitun kansion polku syötetään automaattisesti tekstikenttään.

Näiden komponenttien alapuolelle luotiin kaksi List Box –komponenttia, joihin valitaan analysoitavat kuvat. Vasemmanpuoleiseen List Box -komponenttiin valitaan valaistuskalibrointikuvat ja oikeanpuoleiseen varsinaiset näytekuvat. Kuvat valitaan List Box -komponentteihin niiden alapuolelle sijoitettuja Open File -nappeja painamalla. Nappia painamalla avataan käyttäjälle Windows Explorer -selain, joka on säädetty useamman tiedoston valitsemiseen kerralla. Yhteen List Box -komponenttiin valitaan neljä kuvaa.

Analysointi aloitetaan painamalla Analyze-nappia, joka lisättiin käyttöliittymän alalaitaan. Analysointi suorittaa ensin valaistuskalibroinnin valaistuskalibrointikuvien avulla, jonka jälkeen suoritetaan gradienttien laskenta ja integrointi näytekuville. Kun analyysi on valmis, siitä näytetään käyttäjälle viesti, joka kuitataan OK-nappia painamalla. Analyysin aikana käyttäjälle piirretään useita kuvaajia, joista hän voi valita mitkä hän haluaa tallentaa.

Virtuaalisten pintojen piirtoon tehtiin päivitys käyttöliittymän testauksen jälkeen. Piirron suorittava skripti päivitettiin siten että virtuaalipinnat vastaavat piirtosuunniltaan alkupe-

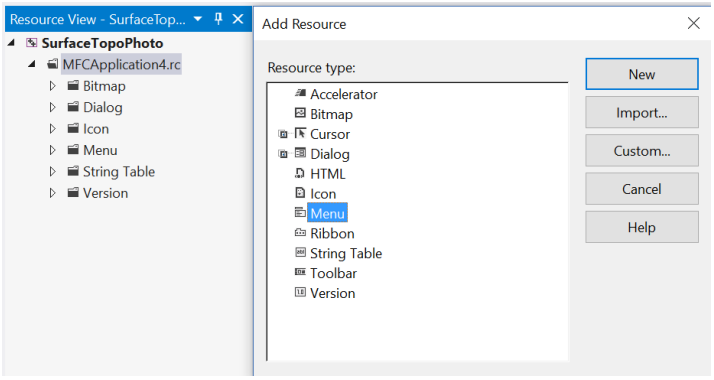
räisiä raakakuvia (liite 3). Myös valonlähteiden koordinaatit laskettiin uudelleen ja asetettiin käyttöliittymän parametreihin. Virtuaalipintojen kuvaajien ulkoasuun tehtiin myös lukuisia muutoksia.

6 VIISTOVALOKUVAUSLAITTEEN KÄYTTÖLIITTYMÄN TYÖSTÖ

VVKL-käyttöliittymän osalta tavoitteena oli parantaa sen käytettävyyttä luomalla siihen uusia toimintoja, kuten Menu, ja histogrammien tuottaminen sekä luomalla uusi selkeä graafinen ilme. Tavoitteena oli myös korjata ohjelman jo entuudestaan tunnetut bugit. Kaikki edellä mainitut tavoitteet saavutettiin.

Menu

Käyttöliittymään lisättiin Menu. Menulla tarkoitetaan käyttöliittymän ylälaidassa sijaitsevaa palkkia, joka sisältää nimettyjä painikkeita, jotka ponnauttavat valintaikkunan niitä painettaessa. Menu-palkki lisätään projektiin Visual Studiossa klikkaamalla hiiren oikeaa painiketta projektin resurssikansion päällä Resource View -näkymässä ja valitsemalla Add Resource. Tämä aktivoi ponnahdusikkunan, josta valitaan Menu ja klikataan New-painiketta (kuva 9).



KUVA 9. Menun luominen VVKL-käyttöliittymään

Kun käyttöliittymän resursseihin on lisätty Menu, aktivoidaan se käyttöliittymän MFCApplication4Dlg.cpp -tiedostossa OnInitDialog-funktiossa kuvan 10 mukaisesti,

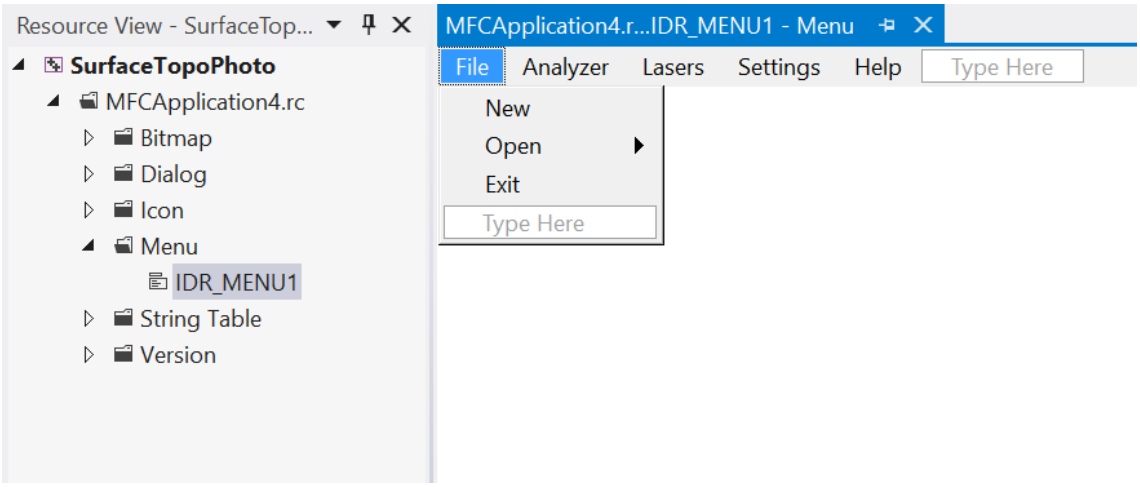
```
// Add menu
_Menu.LoadMenu(IDR_MENU1);
ASSERT(_Menu);
SetMenu(&_Menu);
```

KUVA 10. Menun aktivoiminen

jossa IDR_MENU1 on edellisessä vaiheessa resursseihin lisätty Menu, joka ladataan LoadMenu-funktiolla käyttöliittymän CMenu-luokkaan pohjautuvaan _Menu-objektiin.

ASSERT-funktiota käytetään varmistamaan että _Menu on validi, jos näin ei ole, ohjelman suoritus lopetetaan debuggeria ajettaessa. SetMenu-funktiota käytetään uuden Menun tunnistamiseen ja vanhan Menun korvaamiseen uudella.

Menuun lisättiin joukko painikkeita Visual Studion Resource View -näkymän kautta (kuva 11). Uusien painikkeiden lisääminen käy helposti kirjoittamalla tyhjään nimikenttään painikkeelle nimi kohtaan Type Here.



KUVA 11. Menun Resource View -näkymä

Painikkeiden lisäämisen jälkeen luotiin Event Handler -funktiot niitä vastaaville painikkeille. Tämä tapahtui klikkaamalla hiiren oikeaa painiketta kyseisen napin päällä ja valitsemalla valikosta ”Add Event Handler…”. Avautuvasta ikkunasta valittiin funktiolle ”.cpp”-tiedosto johon se luotiin. Samalla luotiin automaattisesti ”.cpp”-tiedostoa vastaavaan ”.h”-tiedostoon funktion deklaraatio. Funktiot luotiin ”MFCApplication4Dlg.cpp”-tiedostoon. Kyseiseen ”.cpp”-tiedostoon generoitui automaattisesti MESSAGE_MAP-deklaraatiot luoduille funktioille (kuva 12).

Menun painiketta klikatessa ajon aikana, Menu luo COMMAND-viestin kyseisen napin ID:llä. MESSAGE_MAP makron avulla voidaan ohjata haluttu funktio ajoon COMMAND-viestin sisältämän ID:n perusteella. Liitteessä 4 on esitetty VVKL-käyttöliittymään lisätyt funktiot.

```
    ON_COMMAND(ID_ANALYZER_START, &CMFCApplication4Dlg::OnAnalyzerStart)
    ON_COMMAND(ID_FILE_NEWSET, &CMFCApplication4Dlg::OnFileNewset)
    ON_COMMAND(ID_FILE_EXIT, &CMFCApplication4Dlg::OnFileExit)
    ON_COMMAND(ID_HELP_INSTRUCTIONSMANUAL, &CMFCApplication4Dlg::OnHelpInstructionsmanual)
    ON_COMMAND(ID_HELP_CANONMANUAL, &CMFCApplication4Dlg::OnHelpCanonmanual)
    ON_COMMAND(ID_OPEN_EXISTINGPROJECT, &CMFCApplication4Dlg::OnOpenExistingproject)
    ON_COMMAND(ID_OPEN_PROJECTSFOLDER, &CMFCApplication4Dlg::OnOpenProjectsfolder)
    ON_COMMAND(ID_FOCUSCALIB_LED, &CMFCApplication4Dlg::OnFocuscalibLed)
    ON_COMMAND(ID_FOCUSCALIB_LASER, &CMFCApplication4Dlg::OnFocuscalibLaser)
    ON_COMMAND(ID_LASERS_ON, &CMFCApplication4Dlg::OnLasersOn)
    ON_COMMAND(ID_LASERS_OFF, &CMFCApplication4Dlg::OnLasersOff)
    ON_COMMAND(ID_HISTOGRAM_ENABLED32789, &CMFCApplication4Dlg::OnHistogramEnabled32789)
    ON_COMMAND(ID_HISTOGRAM_DISABLED32790, &CMFCApplication4Dlg::OnHistogramDisabled32790)
END_MESSAGE_MAP()
```

KUVA 12. Lisätyt MESSAGE_MAP-deklaraatiot

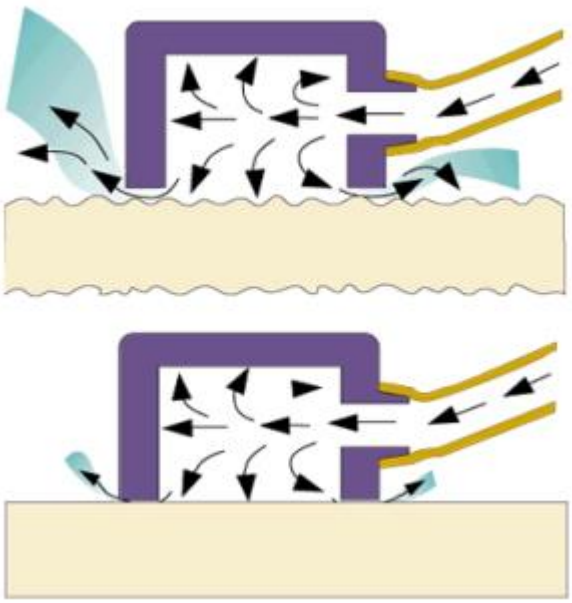
Käyttöliittymään lisättiin myös kyky tuottaa histogrammeja raakakuvista. Marja Mettänen kirjoitti histogrammin tuottamisen toteuttavan lähdekoodin MATLAB-kielellä (liite 5). Kyseinen lähdekoodi paketoitiin ”.exe”-formaattiin jotta se voidaan ajaa suoraan käyttöliittymästä. Kun ohjelma ajetaan, sille annetaan parametrina kuvan polku, josta histogrammi halutaan tuottaa.

Käyttöliittymään luotiin myös lasereiden ohjaustoiminnot. Laserit kytkettiin valo-ohjaimeen Juhani Pitkäsen avustuksella. Valo-ohjain on suunniteltu jo alun perin syöttämään virtaa myös lasereille, joten asennus sujui ongelmitta.

7 SILEYSMITTAUKSET

VVK-laitteistoa testattiin vertaamalla sen tuottamia tuloksia teollisuudessa käytettävän sileysmittarin tuloksiin. Testattavaksi valittiin kolme eri paperi/kartonki-laatua. Testiin valittiin päällystetty taivekartonki, etikettipaperi sekä WF-paperi, NovaPress Silk.

Sileysmittariksi valittiin Messmer Büchellin valmistama ilmanläpäisevyysmittari (PPS), sillä se on standardoitu menetelmä sileyden mittaamiseen. Mittarin toiminta perustuu näytteen läpi ajettavan ilman tarkkailuun ISO 8791-4:2007(E) -standardin mukaisesti. Mitä enemmän ilmaa karkaa näytteen pinnan ja puristimen pinnan välistä, sitä karkeampi on näytteen pinta (kuvio 7). Mittalaite muuntaa automaattisesti saadun sileysarvon mikrometriarvoksi. Kustakin näytteestä mitattiin PPS mittarilla kuudesta mittauspisteestä arvo. Näiden arvojen keskiarvoa käytettiin pinnan sileyden arvona.

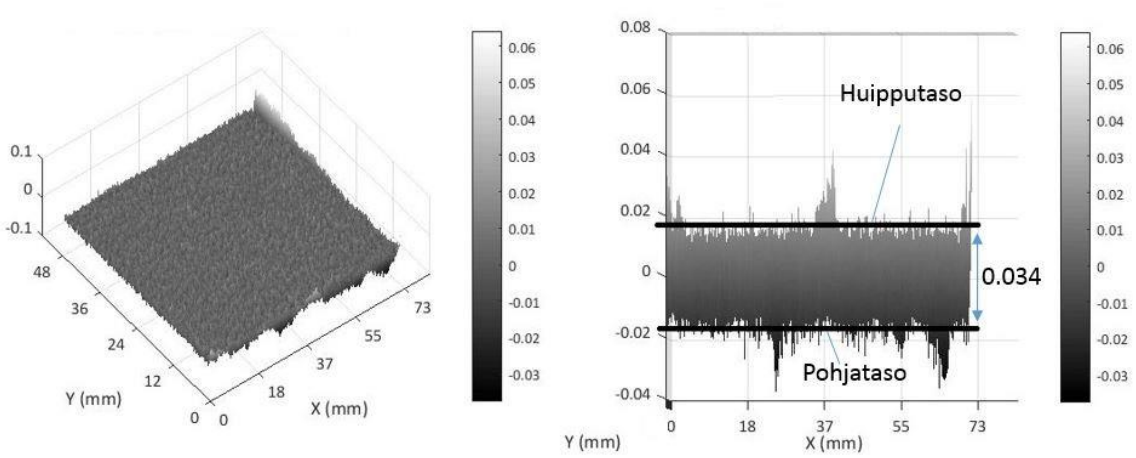


KUVIO 7. PPS-mittarin toimintaperiaate karhealla ja sileällä pinnalla (Laiteohjeet-TAMK)

Viistovalokuvauslaitteeseen ei ole ohjelmoitu varsinaista algoritmia, joka laskisi pinnan sileydelle arvon analysoinnissa tuotetusta virtuaalisesta pinnasta. Tavoitteena olikin silmämääräisesti arvioida analysoitavien näytteiden virtuaalisia pintoja, erityisesti z-akseleita.

VVKL-analysointi ei kuitenkaan tuota virtuaalisen pinnan z-akselille metrijärjestelmän mukaista arvoa joka vastaisi todellisen pinnan korkeutta. Tästä syystä päätettiin arvioida korreloivatko tulokset sileysmittarin tulosten kanssa, vertaamalla sileysarvojen suhteita paperi/kartonki-laatujen välillä. Z-akselista valittiin huippu- ja pohjataso, joiden välinen erotus määriteltiin pinnan sileysarvoksi (kuvio 8).

Virhemarginaaliksi (VM) arvioitiin 10 prosenttia, sillä keskimääräistä pinnan vaihtelua on vaikea arvioida silmämääräisesti. Keskimääräinen pinnan sileysarvo ei voi kuitenkaan olla suurempi kuin valittujen tasojen erotus, sillä huippu- ja pohjatasot on valittu niin että ne ovat tarpeeksi kaukana toisistaan. ”Todellinen” pinnan keskimääräinen sileysarvo on siis jossakin valittujen tasojen välissä.



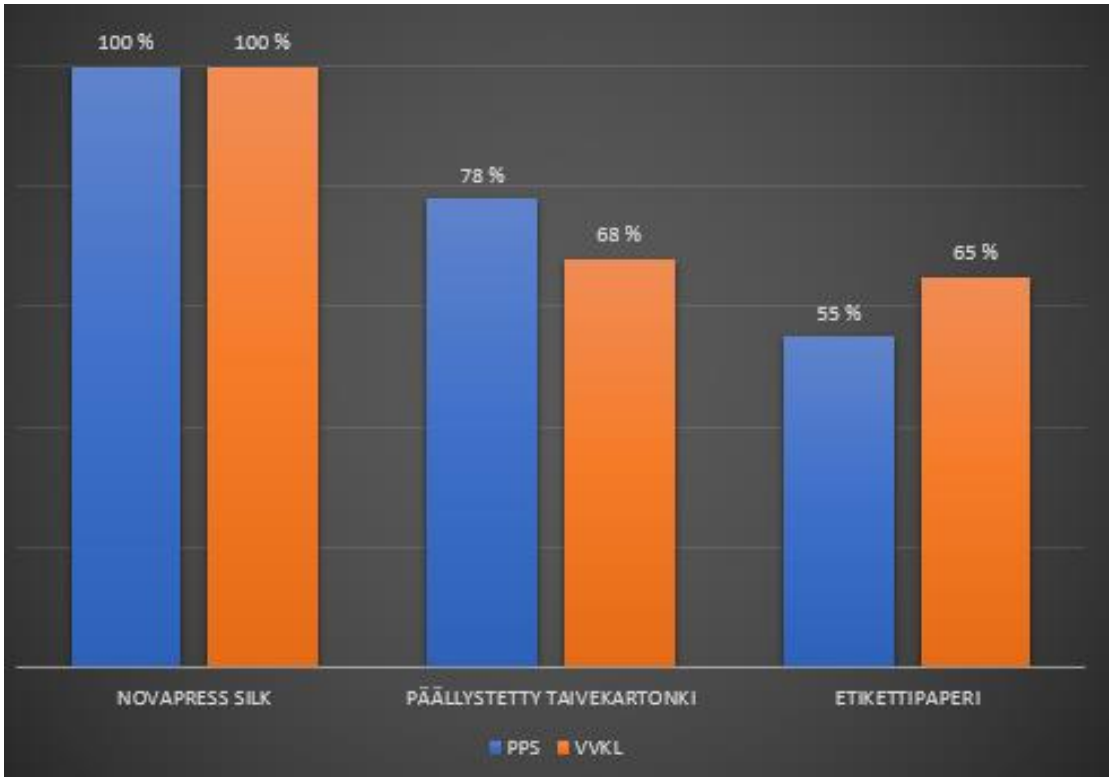
KUVIO 8. Sileysarvon määrittely virtuaalipinnasta NovaPress Silk -paperille

Taulukossa 2 kuvataan sileysmittausten tulokset ja virhemarginaalit. PPS-mittauksen tulokset on annettu mikrometreissä ja VVKL-mittauksen tulokset ovat määrittämättömiä arvoja.

TAULUKKO 2. Sileysmittausten tulokset

NÄYTE	PPS(µm)	CV(%)	VVKL	VM(%)
NovaPress Silk	1,52	2,6	0,034	10
Päällystetty taive-kartonki	1,19	4,2	0,023	10
Etikettipaperi	0,83	6,0	0,022	10

Kuviossa 9 on esitettynä vierekkäin PPS- ja VVKL tulokset kuvaajamuodossa. NovaPress Silk oli molemmissa mittauksissa karhein, minkä vuoksi se on asetettu vertauspisteeksi. Kuvion 9 kuvaajista nähdään, että tulokset ovat hyvin samansuuntaiset. PPS-mittauksissa, päällystetty taivekartonki on lähempänä NovaPress Silk - paperia kuin VVKL-mittauksissa, ja puolestaan etikettipaperi on kauempana NovaPress Silk -paperista PPS-mittauksen perusteella kuin VVKL-mittauksessa. Päällystetty taivekartonki ja etikettipaperi poikkeavat toisistaan 10 prosenttia, joten tulokset ovat virhemarginaalien sisällä, joskin jo pelkästään VVKL-mittauksien virhemarginaaliksi arvioitiin melko korkea 10%.



KUVIO 9. Sileysmittausten tulosten suhteellinen vertailu

8 POHDINTA

Opinnäytetyön tavoitteet saavutettiin asetetussa aikataulussa. Kuva-analysoinnin tueksi tehtiin käyttöliittymä, joka pystyy ajamaan analysoinnin skriptit. Käyttöliittymä vaatii vielä runsaasti kehitystyötä, jos se halutaan käyttömukavuudeltaan VVKL-käyttöliittymän tasolle. Käyttöliittymään voisi lisätä vielä useita toiminnallisuuksia jotka helpottaisivat käyttäjän työskentelyä analysoinnin parissa.

Hyvä lisä käyttöliittymään olisi, jos käyttäjä pystyisi valitsemaan mitkä kuvat analysoinnin aikana piirretään. Tämän lisäksi kuvien tallennus tapahtuu nykyisessä versiossa manuaalisesti suoraan kuvaajan ikkunasta, mikä voi olla turhauttavaa useita näytteitä analysoitaessa. Analysoinnin raakadatan tallennuksen suunnittelu ja toteutus olisi myös edellytyksenä sille, että kuvaajat voitaisiin piirtää uudelleen raakadatasta ilman raskasta analysoinnin suorittamista. Raakadatan käyttö alentaisi huomattavasti tietokoneen muistin käyttöä tietoja arkistoitaessa.

Nykyisessä versiossa kameran ottaman kuvan mittasuhteet ovat kovakoodattu käyttöliittymän lähdekoodiin. Käytännössä viistovalokuvauslaitteen kameraa voi liikuttaa kiskoa pitkin, mikä johtaa kuvan mittasuhteiden muuttumiseen. Käyttöliittymään olisikin hyvä lisätä mahdollisuus muuttaa mittasuhdeparametreja suoraan käyttöliittymästä.

VVKL-käyttöliittymään tehdyt päivitykset onnistuivat suunnitellusti, eikä ohjelmaan jäänyt tunnettuja bugeja. VVKL-käyttöliittymää voi toki kehittää edelleen lisäämällä siihen uusia toiminnallisuuksia.

VVKL-käyttöliittymään voisi esimerkiksi lisätä mahdollisuuden tarkentaa kuvaa asteittain napin painalluksella. Nykyinen versio käyttää kameran automaattitarkennusta. Muutoksia voisi tehdä myös ledien polttoaikaan. Tällä hetkellä valo-ohjain katkaisee ledin polton 5 sekunnin jälkeen, mikä rajoittaa kameran parametrien säätöä livekuvan perusteella.

Mielenkiintoinen kehitysidea olisi myös sarjakuvauksen mahdollisuuden selvittäminen. Kuvausaikaa saataisiin todennäköisesti lyhennettyä hieman, jos kuvat otettaisiin sarjassa ja ladattaisiin kamerasta yhdellä kerralla. Mitään merkittävää etua nykyiseen tilanteeseen sillä tuskin kuitenkaan saataisiin.

Hieman ongelmalliseksi laserin ohjauksessa koitui valo-ohjaimen tapa jättää valittu leditason valinta päälle valo-ohjaimeen. Mahdollisuus alustaa valo-ohjaimeen tehty leditason valinta nopeuttaisi tarkennuskalibrointikuvien ottoa muutamalla sekunnilla.

Viistovalokuvaustekniikan käyttäminen sileysmittauksessa tuo teoriassa merkittäviä etuja teollisuudessa yleisesti käytettäviin metodeihin. Viistovalokuvaus on kontaktiton mittaustapa, joten näytteen pintaa ei häiritä millään tavalla, toisin kuin perinteisimmissä metodeissa.

Sileysmittauksessa saadut tulokset ovat rohkaisevia. Tuloksissa on selvästi nähtävissä, että virtuaalisen pinnan z-akselin suuntaiset korkeuden vaihtelut ovat suhteellisesti melko lähellä PPS-mittarin tuottamia tuloksia. Helpommin ymmärrettäviä tuloksia saataisiin kalibroimalla z-akseli metrijärjestelmän mukaiseksi ja tarkempia tuloksia saataisiin laskemalla sileysarvot suoraan analysoinnin raakadatasta. Sileysmittauksen lisäksi kuva-analysointia voisi kehittää myös esimerkiksi näytteen pinnan virheiden paikantamiseen ja laskemiseen.

LÄHTEET

Bjönander, Stefan. 2008. *Microsoft Visual C++ Windows Applications by Example : Code And Explanation For Real-World MFC C++ Applications.* Birmingham : Packt Publishing Ltd, 2008.

Innventia. OptiTopo - the technique and fields of application. [Viitattu: 5. 12 2017.] http://www.innventia.com/Documents/Produktblad/Material%20processes/Pappersyta/OptiTopo%20-%20the%20technique%20and%20fields%20of%20application.pdf.

Kuparinen, Toni. 2008. *Reconstruction and analysis of surface variation using photometric stereo.* Lappeenrannan teknillinen yliopisto, 2008. Väitöskirja.

Laiteohjeet-TAMK. PPS-Sileys. [Luettu: 1.12.2017]

Mastosalo, Eetu. 2017. *Viistovalokuvauslaitteiston kuva-analysoinnin käyttöönotto.* Tampereen ammattikorkeakoulu, 2017. Opinnäytetyö.

MathWorks. MATLAB Programming Fundamentals. [Viitattu: 29. 11 2017.] https://www.mathworks.com/help/pdf_doc/matlab/matlab_prog.pdf.

Microsoft. 2015a. Document/View Architecture. 2015a. [Viitattu: 28. 11 2017.] https://msdn.microsoft.com/en-us/library/4x1xy43a.aspx.

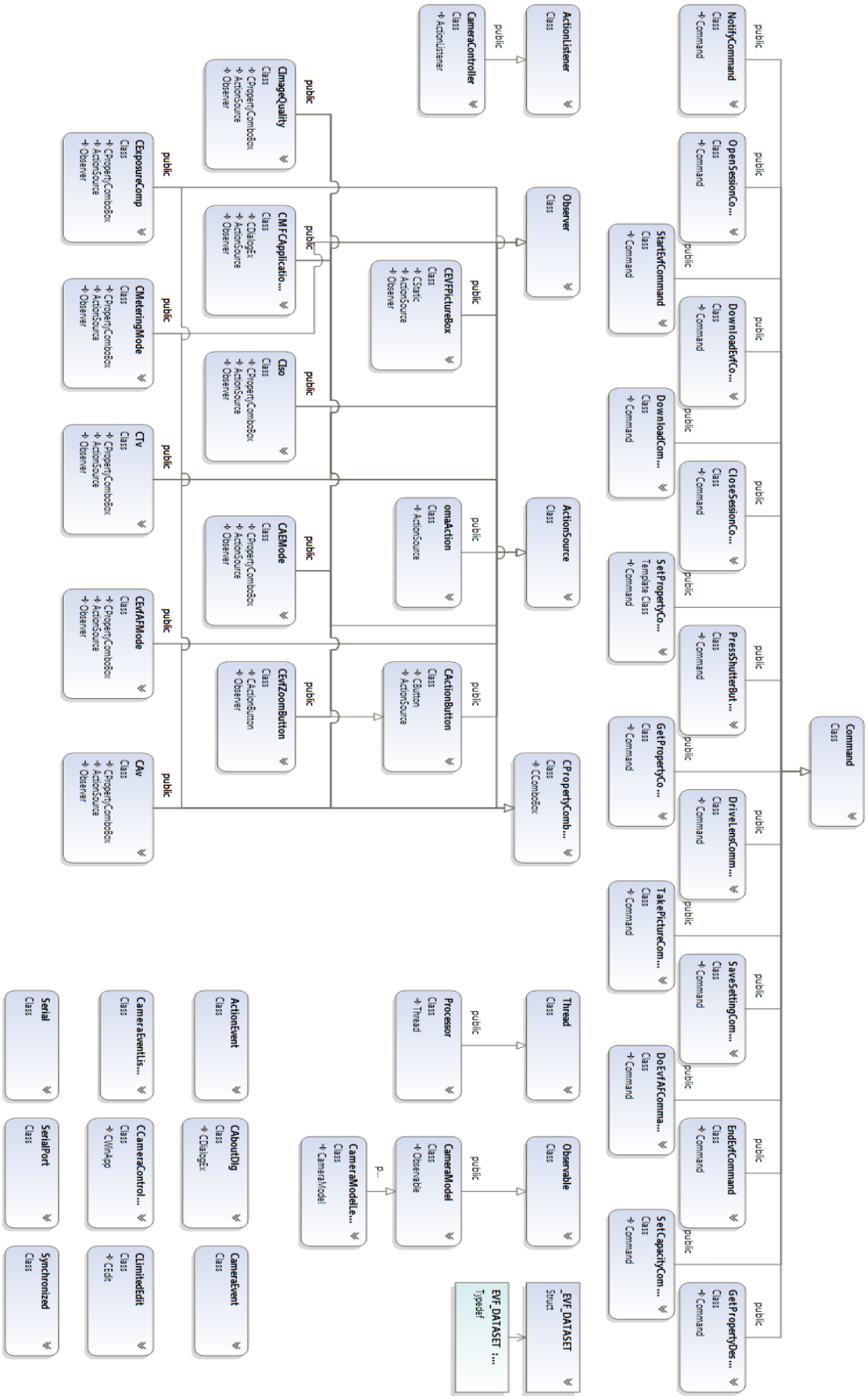
Microsoft. 2015b. Hierachy Chart. 2015b. [Viitattu: 27. 11 2017.] https://msdn.microsoft.com/en-us/library/ws8s10w4.aspx.

MVTec. Photometric Stereo. [Viitattu: 5. 12 2017.] http://www.mvtec.com/doc/halcon/13/en/toc_3dreconstruction_photometricstereo.html.

Vuorenmaa, Panu. 2015. *Viistovalokuvauslaitteen kuvausvalojen ohjainlaite.* Tampereen ammattikorkeakoulu, 2015. Opinnäytetyö.

LIITTEET

Liite 1. VVKL-käyttöliittymän luokkarakenne



Liite 2. Kuva-analysoinnin käyttöliittymän lähdekoodit


```
classdef Analyzer_temp < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        AnalyzerUIFigure            matlab.ui.Figure
        ImagefolderpathEditField    matlab.ui.control.EditField
        ProgressEditField           matlab.ui.control.EditField
        Gauge                       matlab.ui.control.LinearGauge
        OpenfolderButton            matlab.ui.control.Button
        OpenfilesButton             matlab.ui.control.Button
        OpenfilesButton_2           matlab.ui.control.Button
        AnalyzeButton               matlab.ui.control.Button
        CutofflimitEditFieldLabel   matlab.ui.control.Label
        CutofflimitEditField        matlab.ui.control.NumericEditField
        SampleImagesListBox         matlab.ui.control.ListBox
        ListBox                     matlab.ui.control.ListBox
        LEDLayerButtonGroup         matlab.ui.container.ButtonGroup
        CButton                     matlab.ui.control.RadioButton
        BButton                     matlab.ui.control.RadioButton
        AButton                     matlab.ui.control.RadioButton
        ImagefolderfathLabel        matlab.ui.control.Label
        LightcalibrationimagesLabel matlab.ui.control.Label
        SampleimagesLabel           matlab.ui.control.Label
        Button                      matlab.ui.control.Button
        Button_3                    matlab.ui.control.Button
        Button_4                    matlab.ui.control.Button
        Button_5                    matlab.ui.control.Button
        CheckBox                    matlab.ui.control.CheckBox
        SkipcalibrationLabel        matlab.ui.control.Label
        mmLabel                     matlab.ui.control.Label
    end

    properties (Access = public)
        is_analysing = 0; % parameter is used to block multiple simultaneouse analyze calls
        % default forlder for Images:
        loaddir_data = '';

        % default ID for Imageseries:
        CalibNamID = '';
        % default notes for Imageseries:
        calib_desc = '';
        % default Scalecalibration Image:
        FileNam_mm = 'Scalecalib_Image_1.CR2';
        % default Focuscalibration Image:
        FileNam_focus = 'Focuscalib_Image_1.CR2';
        % Height of the camera
        cameraHeight_mm = 340;
```

(jatkuu)

```
        % default Image size in mm, use ScalecalibrationImage to define it (Y  X)!!
        siz_kuva = [48  72];

        % default Light calibration Images:
        VKUVA1 = 'Lightcalib_Image_1.CR2';
        VKUVA2 = 'Lightcalib_Image_2.CR2';
        VKUVA3 = 'Lightcalib_Image_3.CR2';
        VKUVA4 = 'Lightcalib_Image_4.CR2';
        % default Sample Images:
        KUVA1 = 'Sample_Image_1.CR2';
        KUVA2 = 'Sample_Image_2.CR2';
        KUVA3 = 'Sample_Image_3.CR2';
        KUVA4 = 'Sample_Image_4.CR2';

        % gradient integration parameters
        lam = 2;          % filtered cutoff limit (mm)

        % calibration parameters
        N = [];
        polar_deg = 0;
        ylims = [];
        xlims = [];
        res = [];
        azims_deg = 0;
        radii = 0;
    end

    methods (Access = private)

        function results = verifyData(app)
            % VerifyData is used in the beginning of the analysing process, after pressing the
Analyze button
            % to verify that the files listed in the Image editfields are acceptable for analysis.
            % The function returns 0 when all the files passed the check, and 1 when at least
one check failed.
            % In the case of failure an error message is prompted.

            results = 0;

            if get(app.CheckBox, 'Value') == 0   % if calibration is not skipped

                % checkin Light calibration Images
                temp = app.ListBox.Items;
                slash = '\';
                cpath = strcat(app.loaddir_data,slash);
```

```
            if length(temp) >= 4      % if the length of the character array is more than or
equal to 4 continue
                % going trough Light calibration Images and verifying their existance
                for n = 1:4
                    ctpath = strcat(cpath,temp{n}); % concatenating a path for every Image
                    if exist(ctpath, 'file') == 2 && results == 0 % verifying existance
                                                        % exist returns 2 if cpath is a file
                    else
                        % File does not exist.
                        if results == 0 % if no failure has occured prompt one
                            warningMessage = sprintf('Warning: file does not exist:\n%s',
ctpath);
                            uiwait(msgbox(warningMessage));
                            results = 1;
                        end
                    end
                end
            else
                warningMessage = sprintf('Warning: Invalid Light calibration Images!');
                uiwait(msgbox(warningMessage));
                results = 1;
            end
        end
        % going through Sample Images and verifying their existance
        temp = app.SampleImagesListBox.Items;
        slash = '\';
        cpath = strcat(app.loaddir_data,slash);

        if length(temp) >= 4    % if the length of the character array is more than or equal
to 4 continue
            for n = 1:4
                ctpath = strcat(cpath,temp{n}); % concatenating a path for every Image
                if exist(ctpath, 'file') == 2 && results == 0 % verifying existance...
                                                    % exist returns 2 if cpath is a file
                    % continue
                else
                    % File does not exist.
                    if results == 0 % if no failure has occured prompt one
                        warningMessage = sprintf('Warning: file does not exist:\n%s', ctpath);
                        uiwait(msgbox(warningMessage));
                        results = 1;
                    end
                end
            end
        else
            warningMessage = sprintf('Warning: Invalid Sample Images!');
            uiwait(msgbox(warningMessage));
            results = 1;
        end
    end
```

```
%-----------------------------------------------

    function verifyDefaultImages(app)
        % verifyDefaultImages is used to verify that Default Image files exist in the cur-
rent Image folder.
        % If it is verified that images exist they are inserted in to editfields.
        % If verification fails the editfield is left empty.

        % clear all editfiels
        clear = {};
        set(app.ListBox, 'Items', clear);                % for Light calibrationImages ListBox
        set(app.SampleImagesListBox, 'Items', clear);        % for Sample Images ListBox


        % checkin Light calibration Images
        valid = 1;
        defaultfilesL{1,1} = [app.VKUVA1];
        defaultfilesL{1,2} = [app.VKUVA2];
        defaultfilesL{1,3} = [app.VKUVA3];
        defaultfilesL{1,4} = [app.VKUVA4];
        slash = '\';
        cpath = strcat(app.loaddir_data,slash);

        % going through Light calibration Images and verifying their existance
        for n = 1:4
            ctpath = strcat(cpath,defaultfilesL{n}); % concatenating a path for every Image

            if exist(ctpath, 'file') && valid == 1 % verifying existance
                %continue
            else
                % File does not exist.
                if valid == 1
                    valid = 0;
                end
            end
        end
        % if all paths so far are valid, insert them to listbox
        if valid == 1
            set(app.ListBox, 'Items', defaultfilesL);    % for LightcalibrationImagesListBox
        end

        % going trough Sample Images and verifying their existance
        valid = 1;
        defaultfilesS{1,1} = [app.KUVA1];
        defaultfilesS{1,2} = [app.KUVA2];
        defaultfilesS{1,3} = [app.KUVA3];
        defaultfilesS{1,4} = [app.KUVA4];
        slash = '\';
        cpath = strcat(app.loaddir_data,slash);
```

```
        for n = 1:4
            ctpath = strcat(cpath,defaultfilesS{n}); % concatenating a path for every Image

            if exist(ctpath, 'file') && valid == 1 % verifying existance
                %continue
            else
                % File does not exist.
                if valid == 1
                    valid = 0;
                end
            end
        end
        % if all paths so far are valid, insert them to listbox
        if valid == 1
            set(app.SampleImagesListBox, 'Items', defaultfilesS);     % for SampleImages-
ListBox
        end

    end

    function results = verifyCalibrationData(app)
        % verifyCalibrationData is used to check if calibration data already exists in the
current project folder
        % returns 1 if data was succesfully verified otherwise returns 0!

        % creating path to AnalyzerData:
        Analyzer = fullfile(app.loaddir_data, 'AnalyzerData');
        % path to light calibration data:
        calibdir = fullfile(Analyzer, 'Kalibrointi');
        tic = 0;
        results = 0;

        % run trough all the calibration files
        for jj=1:4

            if (app.LEDLayerButtonGroup.SelectedObject.Text == 'A') % If A-Layer se-
lected
                calibnam                    =                    ['Kalibrointi_',app.Ca-
libNamID,'_Kulma_','75','_Valo_',num2str(jj),'.mat'];
                calibpath = fullfile(calibdir,calibnam);

                if  exist(calibpath, 'file') == 2 % verifying existance, exist returns 2 if cali-
bpath is a file
                    %continue
                    tic = tic+1;
                    %warningMessage = sprintf('Calib. exists!');
                    %uiwait(msgbox(warningMessage));
                else
                    %warningMessage = sprintf(num2str(exist(calibpath, 'file')));
                    %uiwait(msgbox(warningMessage));
                end
```

```
        elseif (app.LEDLayerButtonGroup.SelectedObject.Text == 'B') % If B-Layer
selected
            calibnam                    =                    ['Kalibrointi_',app.Ca-
libNamID,'_Kulma_','60','_Valo_',num2str(jj),'.mat'];
            calibpath = fullfile(calibdir,calibnam);

            if  exist(calibpath, 'file') == 2 % verifying existance, exist returns 2 if cali-
bpath is a file
                %continue
                tic = tic+1;
                %warningMessage = sprintf('Calib. exists!');
                %uiwait(msgbox(warningMessage));
            else
                %warningMessage = sprintf(num2str(exist(calibpath, 'file')));
                %uiwait(msgbox(warningMessage));
            end

        elseif (app.LEDLayerButtonGroup.SelectedObject.Text == 'C') % If C-Layer
selected
            calibnam                    =                    ['Kalibrointi_',app.Ca-
libNamID,'_Kulma_','30','_Valo_',num2str(jj),'.mat'];
            calibpath = fullfile(calibdir,calibnam);

            if  exist(calibpath, 'file') == 2 % verifying existance, exist returns 2 if cali-
bpath is a file
                %continue
                tic = tic+1;
                %warningMessage = sprintf('Calib. exists!');
                %uiwait(msgbox(warningMessage));
            else
                %warningMessage = sprintf(num2str(exist(calibpath, 'file')));
                %uiwait(msgbox(warningMessage));
            end
        end
    end

    if tic == 4
        results = 1;
        load(calibpath); % loading parameters from calibration file
        app.N = N;
        app.polar_deg = polar_deg;
        app.ylims = ylims;
        app.xlims = xlims;
        app.res = res;
        app.azims_deg = azims_deg;
        app.radii = radii;
```

```
            %warningMessage = sprintf('Calibration data found!');
            %uiwait(msgbox(warningMessage));

            %if app.CheckBox.Value == 0
               %set(app.CheckBox,'value',1);      % setting 'skip calibration'-checkbox to
checked
            %end
         %else
            %if app.CheckBox.Value == 1
               %app.CheckBox.Value = 0;          % setting 'skip calibration'-checkbox to
unchecked
            %end
         end
      end

   end

   methods (Access = private)

      % Code that executes after component creation
      function startupFcn(app)
         % constructing path to SaveFiles folder
         app.loaddir_data = pwd;    % pwd returns current folder
         SaveFiles = 'SaveFiles';
         app.loaddir_data = fullfile(app.loaddir_data,SaveFiles);
         app.ImagefolderpathEditField.Value = app.loaddir_data;

         % verify that default Images exist and inserts them to editfields
         app.verifyDefaultImages

         % insert speciality parameters to editfields
         app.CutofflimitEditField.Value = app.lam;
      end

      % Button pushed function: AnalyzeButton
      % This function runs the image analysis for selected images
      function AnalyzeButtonPushed(app, event)

         if app.is_analysing == 0
            app.is_analysing = 1; % 1 is the only acceptable value to proceed to analyzing
         else
            app.is_analysing = 2; % acces to Analysis denied (already in progress)
            % popup message to user that Analysis is still in progress
            warningMessage = sprintf('Analysis is already running! Wait until it is finished
before starting a new one!');
            uiwait(msgbox(warningMessage));
         end
```

```
        % if existing calibrationdata is used parameters are loaded
        if get(app.CheckBox, 'Value') == 1
            if app.verifyCalibrationData
                % Parameters loaded from existing calibration data
                N = app.N;
                vakio = 2000;
                polar_deg = app.polar_deg;
                ylims = app.ylims;
                xlims = app.xlims;
                res = app.res;
                azims_deg = app.azims_deg;
                radii = app.radii;

            else
                infoMessage = sprintf('Calibration data was not found! Calibration is recui-
red!');
                uiwait(msgbox(infoMessage));
                app.is_analysing = 0;    % deny access to further analyzing prosess
                %pp.CheckBox.Value = 0;  % setting 'skip calibration'-checkbox to un-
checked
            end
        end


        if  ~app.verifyData && app.is_analysing == 1 % check that data exists

            % creating folder AnalyzerData for results:
            Analyzer = fullfile(app.loaddir_data, 'AnalyzerData');
            [status, msg, msgID] = mkdir(Analyzer);
            % folder for light calibration data:
            polydir = fullfile(Analyzer, 'Kalibrointi');
            [status, msg, msgID] = mkdir(polydir);
            % folder for results:
            savedir = fullfile(Analyzer,'Tulokset');
            [status, msg, msgID] = mkdir(savedir);
            % folder for figures:
            figures = fullfile(Analyzer,'Kuvaajat');
            [status, msg, msgID] = mkdir(figures);

            % Updating parameters for Analysis skripts
            loaddir_data = app.loaddir_data;
            CalibNamID = app.CalibNamID;    % ID for Imageset
            calib_desc = app.calib_desc;    % notes for Imageset
            cameraHeight_mm = app.cameraHeight_mm;
            siz_kuva = app.siz_kuva;        % Imagesize
            FileNam_mm = app.FileNam_mm;
            FileNam_focus = app.FileNam_focus;
            VKUVA1 = app.VKUVA1;
            VKUVA2 = app.VKUVA2;
            VKUVA3 = app.VKUVA3;
            VKUVA4 = app.VKUVA4;
```

```
KUVA1 = app.KUVA1;
KUVA2 = app.KUVA2;
KUVA3 = app.KUVA3;
KUVA4 = app.KUVA4;

% Cutoff filter parameter
lam = app.lam;

Gauge = 0;    % progressbar value
if get(app.CheckBox, 'Value') == 0 % checking if calibration is skipped
   set(app.Gauge, 'Value', 0) % progressbar adjust
   Gauge = 0;    % progressbar value
else
   set(app.Gauge, 'Value', 30) % progressbar adjust
   Gauge = 30;    % progressbar value
end


if (app.LEDLayerButtonGroup.SelectedObject.Text == 'A') % If A-Layer se-
lected run analysis
   % led coordinates
   pos_lights = [  205  205 76;
                -195  205 76;
                -195 -195 76;
                 205 -195 76];

   % Starting the light calibration skript
   if get(app.CheckBox, 'Value') == 0 % checking if calibration is skipped
      app.ProgressEditField.Value = 'Light calibration in progress!';
      TAMK_valaistuskalibrointi
   end

   % Starting the gradient calculation script
   app.ProgressEditField.Value = 'Gradient calculation in progress!';
   TAMK_gradienttilaskenta_main

   % Starting the Image drawing script
   app.ProgressEditField.Value = 'Image drawing in progress!';
   TAMK_kuviot
   set(app.Gauge, 'Value', 100) % progressbar adjust

   app.ProgressEditField.Value = 'Analysis Done!';
   warningMessage = sprintf('Analysis Complete!');
   uiwait(msgbox(warningMessage));
```

```
        elseif (app.LEDLayerButtonGroup.SelectedObject.Text == 'B') % If B-Layer
selected run analysis
            % led coordinates
            pos_lights = [  205  205 163;
                        -195  205 163;
                        -195 -195 163;
                         205 -195 163];

            % Starting the light calibration skript
            if get(app.CheckBox, 'Value') == 0 % checking if calibration is skipped
               app.ProgressEditField.Value = 'Light calibration in progress!';
               TAMK_valaistuskalibrointi
            end

            % Starting the gradientcalculation script
            app.ProgressEditField.Value = 'Gradient calculation in progress!';
            TAMK_gradienttilaskenta_main

            % Starting the Image drawing script
            app.ProgressEditField.Value = 'Image drawing in progress!';
            TAMK_kuviot
            set(app.Gauge, 'Value', 100) % progressbar adjust

            app.ProgressEditField.Value = 'Analysis Done!';
            warningMessage = sprintf('Analysis Complete!');
            uiwait(msgbox(warningMessage));

        elseif (app.LEDLayerButtonGroup.SelectedObject.Text == 'C') % If C-Layer
selected run analysis
            % led coordinates
            pos_lights = [  205  205 490;
                        -195  205 490;
                        -195 -195 490;
                         205 -195 490];

            % Starting the light calibration skript
            if get(app.CheckBox, 'Value') == 0 % checking if calibration is skipped
               app.ProgressEditField.Value = 'Light calibration in progress!';
               TAMK_valaistuskalibrointi
            end

            % Starting the gradientcalculation script
            app.ProgressEditField.Value = 'Gradient calculation in progress!';
            TAMK_gradienttilaskenta_main
```

```
            % Starting the Image drawing script
            app.ProgressEditField.Value = 'Image drawing in progress!';
            TAMK_kuviot
            set(app.Gauge, 'Value', 100) % progressbar adjust

            app.ProgressEditField.Value = 'Analysis Done!';
            warningMessage = sprintf('Analysis Complete!');
            uiwait(msgbox(warningMessage));

        end
        app.is_analysing = 0; % Analysis done and free to start another
    else
        app.is_analysing = 0; % Analysis done and free to start another
    end
end

% Button pushed function: Button
function ButtonPushed(app, event)
    infoMessage = sprintf('To start analyzing images you need to locate them first! Click the Open folder button and select the project folder that contains the images you want to analyze. If there are images present in the folder they will automatically appear in the representive boxes below!');
    uiwait(msgbox(infoMessage));
end

% Button pushed function: Button_3
function Button_3Pushed(app, event)
    infoMessage = sprintf('The box above contains the light calibration images that are ready for the analyzing prosess. You need 4 light calibration images and the first one needs to correspond to led number 1 and so on. Sometimes the default images are not the images you want to use. Maybe you need images 5 thru 8. To do this click the Open files button on the left and select the images you need. You need to select them all at one go!');
    uiwait(msgbox(infoMessage));
end

% Button pushed function: Button_4
function Button_4Pushed(app, event)
    infoMessage = sprintf('The box above contains the sample images that are ready for the analyzing prosess. You need 4 sample images to analyze at the same time and the first one needs to correspond to led number 1 and so on. Sometimes the default images are not the images you want to use. Maybe you need images 5 thru 8. To do this click the Open files button on the left and select the images you need. You need to select them all at one go!');
    uiwait(msgbox(infoMessage));
end
```

```
        % Button pushed function: Button_5
        function Button_5Pushed(app, event)
            infoMessage = sprintf('To start analyzing you need 4 light calibration images and
4 sample images. When you have them in the boxes above all you need to do is to select
the ledlayer that was used to take the images and press Analyze. If you are unsure what
ledlayer was used you can check the metadata.xml file in the project folder! You can skip
the light calibration by checking the box, Skip calib! However calibration data is always
recuired to complete the analysis.');
            uiwait(msgbox(infoMessage));
        end
        % Value changed function: CheckBox
        function CheckBoxValueChanged(app, event)
            value = app.CheckBox.Value;
            %if value == 1
                %if app.verifyCalibrationData == 0
                    %infoMessage = sprintf('No existing calibration data was found! Calibration
is still recuired!');
                    %uiwait(msgbox(infoMessage));
                    %app.CheckBox.Value = 0;
                %end
            %end
        end

        % Value changed function: CutofflimitEditField
        function CutofflimitEditFieldValueChanged(app, event)
            app.lam = app.CutofflimitEditField.Value;
        end

        % Value changed function: ImagefolderpathEditField
        function ImagefolderpathEditFieldValueChanged(app, event)
        end

        % Callback function
        function OpenfileButton_3Pushed(app, event)
        end

        % Button pushed function: OpenfilesButton
        function OpenfilesButtonPushed(app, event)
            % Button that opens files and inserts them in SampleImageslistbox
            [filename, pathname] = uigetfile('*.CR2', 'Choose four Sample Images', app.load-
dir_data, 'MultiSelect', 'on');
            if (iscell(filename))
                if (length(filename) >= 4)
                    app.KUVA1 = char(filename(1));
                    app.KUVA2 = char(filename(2));
                    app.KUVA3 = char(filename(3));
                    app.KUVA4 = char(filename(4));
                end
                set(app.SampleImagesListBox, 'Items', filename);
            end
        end
```

```
        % Button pushed function: OpenfilesButton_2
        function OpenfilesButton_2Pushed(app, event)
            % Button that opens files and inserts them in Light calibrationImageslistbox
            [filename, pathname] = uigetfile('*.CR2', 'Choose four Light calibration Images',
app.loaddir_data, 'MultiSelect', 'on');

            if (iscell(filename))
                if (length(filename) >= 4)
                    app.VKUVA1 = char(filename(1));
                    app.VKUVA2 = char(filename(2));
                    app.VKUVA3 = char(filename(3));
                    app.VKUVA4 = char(filename(4));
                end

                set(app.ListBox, 'Items', filename);
            end
        end

        % Button pushed function: OpenfolderButton
        function OpenfolderButtonPushed(app, event)
            % Button opens a folderfinder and assign chosen path to edit field

            folderpath = uigetdir(app.loaddir_data, 'Choose the folder that contains the pictu-
res you wish to analyze');

            if (ischar(folderpath))
                app.ImagefolderpathEditField.Value = folderpath;
                app.loaddir_data = folderpath;
                % extracting the name of the folder to use as identifier
                [pathstr,name,ext] = fileparts(app.loaddir_data);
                app.CalibNamID = name;
                app.calib_desc = name;

                % reset deafult Image names
                % default Scalecalibration Image:
                app.FileNam_mm = 'Scalecalib_Image_1.CR2';
                % default Light calibration Images:
                app.VKUVA1 = 'Lightcalib_Image_1.CR2';
                app.VKUVA2 = 'Lightcalib_Image_2.CR2';
                app.VKUVA3 = 'Lightcalib_Image_3.CR2';
                app.VKUVA4 = 'Lightcalib_Image_4.CR2';
                % default Sample Images:
                app.KUVA1 = 'Sample_Image_1.CR2';
                app.KUVA2 = 'Sample_Image_2.CR2';
                app.KUVA3 = 'Sample_Image_3.CR2';
                app.KUVA4 = 'Sample_Image_4.CR2';
```

```
            % verifying that default Images exists and inserting them to editfields
            app.verifyDefaultImages;
            %app.verifyCalibrationData;
        end
    end
end

% App initialization and construction
methods (Access = private)
    % Create UIFigure and components
    function createComponents(app)
        % Create AnalyzerUIFigure
        app.AnalyzerUIFigure = uifigure;
        app.AnalyzerUIFigure.Color = [0.902 0.902 0.902];
        app.AnalyzerUIFigure.Position = [100 100 437 393];
        app.AnalyzerUIFigure.Name = 'Analyzer';
        app.AnalyzerUIFigure.Resize = 'off';
        setAutoResize(app, app.AnalyzerUIFigure, true)
        % Create ImagefolderpathEditField
        app.ImagefolderpathEditField = uieditfield(app.AnalyzerUIFigure, 'text');
        app.ImagefolderpathEditField.ValueChangedFcn = createCallbackFcn(app,
@ImagefolderpathEditFieldValueChanged, true);
        app.ImagefolderpathEditField.FontWeight = 'bold';
        app.ImagefolderpathEditField.Position = [22 331 387 22];
        % Create ProgressEditField
        app.ProgressEditField = uieditfield(app.AnalyzerUIFigure, 'text');
        app.ProgressEditField.FontColor = [0.4706 0.6706 0.1882];
        app.ProgressEditField.Position = [21 21 186 22];
        % Create Gauge
        app.Gauge = uigauge(app.AnalyzerUIFigure, 'linear');
        app.Gauge.MajorTicks = [0 20 40 60 80 100];
        app.Gauge.MinorTicks = [];
        app.Gauge.FontColor = [0.4706 0.6706 0.1882];
        app.Gauge.Position = [21 44 186 40];
        % Create OpenfolderButton
        app.OpenfolderButton = uibutton(app.AnalyzerUIFigure, 'push');
        app.OpenfolderButton.ButtonPushedFcn = createCallbackFcn(app, @Openfol-
derButtonPushed, true);
        app.OpenfolderButton.BackgroundColor = [0.9294 0.6902 0.1294];
        app.OpenfolderButton.Position = [22 309 363 22];
        app.OpenfolderButton.Text = 'Open folder';
        % Create OpenfilesButton
        app.OpenfilesButton = uibutton(app.AnalyzerUIFigure, 'push');
        app.OpenfilesButton.ButtonPushedFcn = createCallbackFcn(app, @Openfi-
lesButtonPushed, true);
        app.OpenfilesButton.BackgroundColor = [0.9294 0.6902 0.1294];
        app.OpenfilesButton.Position = [223 163 162 22];
        app.OpenfilesButton.Text = 'Open files';
        % Create OpenfilesButton_2
        app.OpenfilesButton_2 = uibutton(app.AnalyzerUIFigure, 'push');
```

```
        app.OpenfilesButton_2.ButtonPushedFcn = createCallbackFcn(app, @OpenfilesButton_2Pushed, true);
        app.OpenfilesButton_2.BackgroundColor = [0.9294 0.6902 0.1294];
        app.OpenfilesButton_2.Position = [22 162 162 22];
        app.OpenfilesButton_2.Text = 'Open files';
        % Create AnalyzeButton
        app.AnalyzeButton = uibutton(app.AnalyzerUIFigure, 'push');
        app.AnalyzeButton.ButtonPushedFcn = createCallbackFcn(app, @AnalyzeButtonPushed, true);
        app.AnalyzeButton.BackgroundColor = [0 0.451 0.7412];
        app.AnalyzeButton.Position = [223 21 162 22];
        app.AnalyzeButton.Text = 'Analyze';
        % Create CutofflimitEditFieldLabel
        app.CutofflimitEditFieldLabel = uilabel(app.AnalyzerUIFigure);
        app.CutofflimitEditFieldLabel.HorizontalAlignment = 'right';
        app.CutofflimitEditFieldLabel.FontWeight = 'bold';
        app.CutofflimitEditFieldLabel.Position = [24 126 67 15];
        app.CutofflimitEditFieldLabel.Text = 'Cutoff limit';
        % Create CutofflimitEditField
        app.CutofflimitEditField = uieditfield(app.AnalyzerUIFigure, 'numeric');
        app.CutofflimitEditField.ValueChangedFcn = createCallbackFcn(app, @CutofflimitEditFieldValueChanged, true);
        app.CutofflimitEditField.Limits = [1 Inf];
        app.CutofflimitEditField.Position = [100 121 47 22];
        app.CutofflimitEditField.Value = 2;
        % Create SampleImagesListBox
        app.SampleImagesListBox = uilistbox(app.AnalyzerUIFigure);
        app.SampleImagesListBox.Items = {};
        app.SampleImagesListBox.Position = [223 184 186 87];
        app.SampleImagesListBox.Value = {};
        % Create ListBox
        app.ListBox = uilistbox(app.AnalyzerUIFigure);
        app.ListBox.Items = {};
        app.ListBox.Position = [22 184 186 87];
        app.ListBox.Value = {};
        % Create LEDLayerButtonGroup
        app.LEDLayerButtonGroup = uibuttongroup(app.AnalyzerUIFigure);
        app.LEDLayerButtonGroup.BorderType = 'none';
        app.LEDLayerButtonGroup.TitlePosition = 'centertop';
        app.LEDLayerButtonGroup.Title = 'LED-Layer';
        app.LEDLayerButtonGroup.BackgroundColor = [0.902 0.902 0.902];
        app.LEDLayerButtonGroup.Position = [223 45 66 100];
        % Create CButton
        app.CButton = uiradiobutton(app.LEDLayerButtonGroup);
        app.CButton.Text = 'C';
        app.CButton.FontSize = 16;
        app.CButton.FontWeight = 'bold';
        app.CButton.Position = [11 50 33.5625 20];
```

```
            % Create BButton
            app.BButton = uiradiobutton(app.LEDLayerButtonGroup);
            app.BButton.Text = 'B';
            app.BButton.FontSize = 16;
            app.BButton.FontWeight = 'bold';
            app.BButton.Position = [11 28 33.5625 20];
            % Create AButton
            app.AButton = uiradiobutton(app.LEDLayerButtonGroup);
            app.AButton.Text = 'A';
            app.AButton.FontSize = 16;
            app.AButton.FontWeight = 'bold';
            app.AButton.Position = [11 6 33.5625 20];
            app.AButton.Value = true;
            % Create ImagefolderfathLabel
            app.ImagefolderfathLabel = uilabel(app.AnalyzerUIFigure);
            app.ImagefolderfathLabel.FontWeight = 'bold';
            app.ImagefolderfathLabel.Position = [22 358 105 15];
            app.ImagefolderfathLabel.Text = 'Image folder fath:';
            % Create LightcalibrationimagesLabel
            app.LightcalibrationimagesLabel = uilabel(app.AnalyzerUIFigure);
            app.LightcalibrationimagesLabel.FontWeight = 'bold';
            app.LightcalibrationimagesLabel.Position = [22 271 145 15];
            app.LightcalibrationimagesLabel.Text = 'Light calibration images:';
            % Create SampleimagesLabel
            app.SampleimagesLabel = uilabel(app.AnalyzerUIFigure);
            app.SampleimagesLabel.FontWeight = 'bold';
            app.SampleimagesLabel.Position = [223 271 97 15];
            app.SampleimagesLabel.Text = 'Sample images:';
            % Create Button
            app.Button = uibutton(app.AnalyzerUIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed, true);
            app.Button.BackgroundColor = [0.302 0.749 0.9294];
            app.Button.FontWeight = 'bold';
            app.Button.Position = [384 309 25 22];
            app.Button.Text = '?';
            % Create Button_3
            app.Button_3 = uibutton(app.AnalyzerUIFigure, 'push');
            app.Button_3.ButtonPushedFcn = createCallbackFcn(app, @Button_3Pushed,
true);
            app.Button_3.BackgroundColor = [0.302 0.749 0.9294];
            app.Button_3.FontWeight = 'bold';
            app.Button_3.Position = [182.5 162 25 22];
            app.Button_3.Text = '?';
            % Create Button_4
            app.Button_4 = uibutton(app.AnalyzerUIFigure, 'push');
            app.Button_4.ButtonPushedFcn = createCallbackFcn(app, @Button_4Pushed,
true);
            app.Button_4.BackgroundColor = [0.302 0.749 0.9294];
            app.Button_4.FontWeight = 'bold';
            app.Button_4.Position = [384 163 25 22];
            app.Button_4.Text = '?';
```

```
            % Create Button_5
            app.Button_5 = uibutton(app.AnalyzerUIFigure, 'push');
            app.Button_5.ButtonPushedFcn = createCallbackFcn(app, @Button_5Pushed,
true);
            app.Button_5.BackgroundColor = [0.302 0.749 0.9294];
            app.Button_5.FontWeight = 'bold';
            app.Button_5.FontColor = [0.149 0.149 0.149];
            app.Button_5.Position = [385 21 25 22];
            app.Button_5.Text = '?';
            % Create CheckBox
            app.CheckBox = uicheckbox(app.AnalyzerUIFigure);
            app.CheckBox.ValueChangedFcn = createCallbackFcn(app, @CheckBoxVa-
lueChanged, true);
            app.CheckBox.Text = '';
            app.CheckBox.FontSize = 16;
            app.CheckBox.Position = [130 94 37.453125 20];
            % Create SkipcalibrationLabel
            app.SkipcalibrationLabel = uilabel(app.AnalyzerUIFigure);
            app.SkipcalibrationLabel.FontWeight = 'bold';
            app.SkipcalibrationLabel.Position = [24 97 91 15];
            app.SkipcalibrationLabel.Text = 'Skip calibration';
            % Create mmLabel
            app.mmLabel = uilabel(app.AnalyzerUIFigure);
            app.mmLabel.Position = [149 124 25 15];
            app.mmLabel.Text = 'mm';
        end
    end
    methods (Access = public)
        % Construct app
        function app = Analyzer_temp()
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.AnalyzerUIFigure)
            % Execute the startup function
            runStartupFcn(app, @startupFcn)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.AnalyzerUIFigure)
        end
    end
end
```

Liite 3. Virtuaalipintojen piirtämisen lähdekoodit

1(6)

```
%% Kuvaajien piirtäminen tulosten pohjalta
% EM 16.4.2017
% muokattu AS 17.11.2017

disp('Kuvaajien piirtäminen...')

% Määritetään akselit millimetreissä
xx = linspace(0,siz_kuva(2),N(2)); % x-akseli (mm)
yy = linspace(0,siz_kuva(1),N(1)); % y-akseli (mm)
cutLims = [1591,1885,2457,2751];

xxc = xx(cutLims(3):cutLims(4));
yyc = yy(cutLims(1):cutLims(2));
x0_mm = xx(cutLims(3));
y0_mm = yy(cutLims(1));
Wcut = xx(cutLims(4))-xx(cutLims(3));
Hcut = yy(cutLims(2))-yy(cutLims(1));


% AS 24.8.2017
% creating axels for full images
yAxelFullI = linspace(0,siz_kuva(1)); % y-axel for full image from 0
to max in (mm)
xAxelFullI = linspace(0,siz_kuva(2)); % x-axel for full image from 0
to max in (mm)
% creating axels for cut images
y_len_cut = (cutLims(2)-cutLims(1)) * siz_kuva(1)/N(1);  % the length
of the cut image in y direction (mm)
x_len_cut = (cutLims(4)-cutLims(3)) * siz_kuva(2)/N(2);  % the length
of the cut image in x direction (mm)
yAxelCutI = linspace(0,y_len_cut); % x-axel for cut image from 0 to
max in (mm)
xAxelCutI = linspace(0,x_len_cut); % y-axel for cut image from 0 to
max in (mm)
% flipping images to create custom y axis
grad_x_full = flipud(gx);              % gradient image must be flipped
to change y-axis direction
grad_y_full = flipud(gy);              % gradient image must be flipped
to change y-axis direction
grad_x1 = flipud(gx1);                 % gradient image must be flipped
to change y-axis direction
grad_y1 = flipud(gy1);                 % gradient image must be flipped
to change y-axis direction
f1 = flipud(f);                        % image must be flipped to
change y-axis direction
f_full1 = flipud(f_full);              % image must be flipped to
change y-axis direction
ff1 = flipud(ff);                      % image must be flipped to
change y-axis direction
ff_full1 = flipud(ff_full);            % image must be flipped to
change y-axis direction
```

(jatkuu)

```
% X-suuntainen gradientti koko kuvasta
set(0,'DefaultFigureColor','White','defaultaxesfontsize',12,'Defaul-
tAxesFontname','Calibri','DefaultTextFontName','Calibri')
gradX = figure;
imagesc(xAxelFullI,yAxelFullI,grad_x_full)
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel 'X (mm)'
ylabel 'Y (mm)'
title 'X-gradient'
set(gca,'YDir','normal')
set(gradX, 'name', 'gx', 'numbertitle', 'off');
%baseFileName = [CalibNamID,'_Kulma',num2str(polar_deg),'_x-gra-
dientti.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');

% Y-suuntainen gradientti koko kuvasta
gradY = figure;
imagesc(xAxelFullI,yAxelFullI,grad_y_full)
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel 'X (mm)'
ylabel 'Y (mm)'
title 'Y-gradient'
set(gca,'YDir','normal')
set(gradY, 'name', 'gy', 'numbertitle', 'off');
%baseFileName = [CalibNamID,'_Kulma',num2str(polar_deg),'_y-gra-
dientti.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');


Gauge = Gauge+4;
set(app.Gauge, 'Value', Gauge) % progressbar adjus


% X-suuntainen gradientti integroitavan pinnan osiosta
set(0,'DefaultFigureColor','White','defaultaxesfontsize',12,'Defaul-
tAxesFontname','Calibri','DefaultTextFontName','Calibri')
gradXp = figure;
imagesc(xAxelCutI,yAxelCutI,grad_x1)
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel 'X (mm)'
ylabel 'Y (mm)'
title 'X-gradient, small'
set(gca,'YDir','normal')
set(gradXp, 'name', 'gx1', 'numbertitle', 'off');
%baseFileName = [CalibNamID,'_Kulma',num2str(polar_deg),'_x-gra-
dientti_osa.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');
```

```
% Y-suuntainen gradientti integroitavan pinnan osiosta
gradYp = figure;
imagesc(xAxelCutI,yAxelCutI,grad_y1)
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel 'X (mm)'
ylabel 'Y (mm)'
title 'Y-gradient, small'
set(gca,'YDir','normal')    % changing y-axel direction to normal
set(gradYp, 'name', 'gy1', 'numbertitle', 'off');
%baseFileName = [CalibNamID,'_Kulma',num2str(polar_deg),'_y-gra-
dientti_osa.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');


Gauge = Gauge+4;
set(app.Gauge, 'Value', Gauge) % progressbar adjus


% Suodattamaton integroitu pinta osio
F = figure;
imagesc(xAxelCutI,yAxelCutI,f1)
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel 'X (mm)'
ylabel 'Y (mm)'
title 'Integrated surface, small(unfiltered)'
set(gca,'YDir','normal')    % changing y-axel direction to normal
set(F, 'name', 'f', 'numbertitle', 'off');
%baseFileName = [CalibNamID,'_Kulma',num2str(po-
lar_deg),'_pinta_osio.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');

% Suodattamaton integroitu pinta
F_full = figure;
imagesc(xAxelFullI,yAxelFullI,f_full1)
%hr = rectangle('Position',[x0_mm, y0_mm, Wcut, Hcut],'edgeco-
lor','r');
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel 'X (mm)'
ylabel 'Y (mm)'
title 'Integrated surface(unfiltered)'
set(gca,'YDir','normal')    % changing y-axel direction to normal
set(F_full, 'name', 'f_full', 'numbertitle', 'off');
%baseFileName = [CalibNamID,'_Kulma',num2str(polar_deg),'_pinta.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');


Gauge = Gauge+4;
set(app.Gauge, 'Value', Gauge) % progressbar adjus
```

```
% Suodatettu integroitu pinta
FF = figure;
imagesc(xAxelCutI,yAxelCutI,ff1); axis image; colormap gray;
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel 'X (mm)'
ylabel 'Y (mm)'
title (['Integrated surface, small (filtered, cutoff limit
',num2str(lam),' mm)']);
set(gca,'YDir','normal')    % changing y-axel direction to normal
set(FF, 'name', 'f_full', 'numbertitle', 'off');
%baseFileName = [CalibNamID,'_Kulma',num2str(po-
lar_deg),'_pinta_suod_osio.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');

% Suodatettu integroitu pinta
FF_full = figure;
imagesc(xAxelFullI,yAxelFullI,ff_full1); axis image; colormap gray;
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel 'X (mm)'
ylabel 'Y (mm)'
set(FF_full, 'name', 'ff_full', 'numbertitle', 'off');
set(gca,'YDir','normal')    % changing y-axel direction to normal
title (['Integrated surface (filtered, cutoff limit ',num2str(lam),'
mm)']);
%baseFileName = [CalibNamID,'_Kulma',num2str(po-
lar_deg),'_pinta_suod.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');


Gauge = Gauge+4;
set(app.Gauge, 'Value', Gauge) % progressbar adjus


% 3D topografia suodattamattomasta pinnasta
topo3Df = figure;
h = surf(f1);
set(h, 'edgecolor', 'none')
hold on
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel('X (mm)', 'fontweight', 'bold')
ylabel('Y (mm)', 'fontweight', 'bold')
%axis([0 300 0 300])
ax = gca;
ax.XTick = [0, 75, 150, 225, 300];
ax.XTickLabel = {'0','1','2','3', '4'};
ax.YTick = [0, 75, 150, 225, 300];
ax.YTickLabel = {'0','1','2','3', '4'};
view (-37.5,70);
title('3D unfiltered surface, small', 'FontSize', 16, 'fontweight',
'bold')
set(topo3Df, 'name', '3D f', 'numbertitle', 'off');
%baseFileName = [CalibNamID,'_Kulma',num2str(po-
lar_deg),'_3D_pinta_osio.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');
```

```
% 3D topografia suodattamattomasta pinnasta
topo3Df_full = figure;
h = surf(f_full1);
set(h, 'edgecolor', 'none')
hold on
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel('X (mm)', 'fontweight', 'bold')
ylabel('Y (mm)', 'fontweight', 'bold')
ax = gca;
ax.XTick = [0,1325,2650,3975,5300];
ax.XTickLabel = {'0','18','37','55','73'};
ax.YTick = [0, 875, 1750, 2625, 3500];
ax.YTickLabel = {'0','12','24','36', '48'};
view (-37.5,70);
title('3D unfiltered surface', 'FontSize', 16, 'fontweight', 'bold')
set(topo3Df_full, 'name', '3D f', 'numbertitle', 'off');
%set(gca, 'ydir', 'reverse');
%baseFileName = [CalibNamID,'_Kulma',num2str(po-
lar_deg),'_3D_pinta.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');


Gauge = Gauge+4;
set(app.Gauge, 'Value', Gauge) % progressbar adjus


% 3D topografia suodatetusta pinnasta
topo3Dff = figure;
h = surf(ff1);
set(h, 'edgecolor', 'none')
hold on
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel('X (mm)', 'fontweight', 'bold')
ylabel('Y (mm)', 'fontweight', 'bold')
ax = gca;
ax.XTick = [0, 75, 150, 225, 300];
ax.XTickLabel = {'0','1','2','3', '4'};
ax.YTick = [0, 75, 150, 225, 300];
ax.YTickLabel = {'0','1','2','3', '4'};
view (-37.5,70);
title(['3D surface, small (filtered, cutoff limit ',num2str(lam),'
mm)'], 'FontSize', 16, 'fontweight', 'bold')
set(topo3Dff, 'name', '3D ff', 'numbertitle', 'off');
%set(gca, 'ydir', 'reverse');
%baseFileName = [CalibNamID,'_Kulma',num2str(po-
lar_deg),'_3D_pinta_suod_osio.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');
% 3D topografia suodatetusta pinnasta
topo3Dff_full = figure;
h = surf(ff_full1);
set(h, 'edgecolor', 'none')
hold on
colormap gray
c = colorbar;
%c.Label.String = 'Z (mm)';
xlabel('X (mm)', 'fontweight', 'bold')
ylabel('Y (mm)', 'fontweight', 'bold')
ax = gca;
```

```
ax.XTick = [0,1325,2650,3975,5300];
ax.XTickLabel = {'0','18','37','55','73'};
ax.YTick = [0, 875, 1750, 2625, 3500];
ax.YTickLabel = {'0','12','24','36', '48'};
view (-37.5,70);
title(['3D surface (filtered, cutoff limit ',num2str(lam),' mm)'],
'FontSize', 16, 'fontweight', 'bold')
set(topo3Dff_full, 'name', '3D ff', 'numbertitle', 'off');
%set(gca, 'ydir', 'reverse');
%baseFileName = [CalibNamID,'_Kulma',num2str(po-
lar_deg),'_3D_pinta_suod.jpg'];
%saveas(gca, fullfile(kuvaajat, baseFileName), 'jpeg');

disp('Kuvaajat piirretty.')
disp('=======================================================');
```

Liite 4. VVKL-käyttöliittymään lisätyt funktiot


```
void CMFCApplication4Dlg::OnAnalyzerStart()
{
    // This function runs Analyzer program using ShellExecute() if
    // user chooses YES in the messagebox

    // If YES is clicked, Analyzer is run
    if (MessageBox(L"Do you wish to run Analyzer?",
        L"Attention!", MB_YESNO) == IDYES)
    {
        // path to Analyzer program
        std::string path = "Analyzer\\Analyzer.exe";
        // convert path string to LPCWSTR
        std::wstring stemp = s2ws(path);
        LPCWSTR Analyzer_path = stemp.c_str();

        if ((int)ShellExecute(0, L"open", Analyzer_path, 0, 0, SW_SHOWDEFAULT) <= 32)
        {
            LPCWSTR viesti = L"Error starting Analyzer!";
            ::MessageBox(NULL, viesti, NULL, MB_OK);
        }
    }

}
void CMFCApplication4Dlg::OnHistogramEnabled()
{
    // This function sets a check to Enabled and removes a check from Disabled
    // in the menuitems under Settings/Histogram and sets _histogram_Enabled parameter to true.

    CMenu* mmenu = &_Menu;
    mmenu->CheckMenuItem(ID_HISTOGRAM_ENABLED32789, MF_CHECKED | MF_BYCOMMAND);
    mmenu->CheckMenuItem(ID_HISTOGRAM_DISABLED32790, MF_UNCHECKED | MF_BYCOMMAND);
    _histogram_Enabled = true;
}
void CMFCApplication4Dlg::OnHistogramDisabled()
{
    // This function sets a check to Disabled and removes a check from Enabled
    // in the menuitems under Settings/Histogram and sets _histogram_Enabled parameter to false.

    CMenu* mmenu = &_Menu;
    mmenu->CheckMenuItem(ID_HISTOGRAM_DISABLED32790, MF_CHECKED | MF_BYCOMMAND);
    mmenu->CheckMenuItem(ID_HISTOGRAM_ENABLED32789, MF_UNCHECKED | MF_BYCOMMAND);
    _histogram_Enabled = false;
}
```

(jatkuu)

```
void CMFCApplication4Dlg::OnFileNewset()
{
    // This function starts a new project.
    // A message is promted for user, if klicked YES run resetDlg() function
    // which will format all key parameters to start a new project.

    // If YES is clicked, resetDlg() is run
    if (MessageBox(L"Do you wish to start a new project?",
        L"Attention!", MB_YESNO) == IDYES)
    {
        resetDlg();
    }
}


void CMFCApplication4Dlg::OnFileExit()
{
    // This function will end the program if user chooses YES in the messagebox.

    // If YES is clicked, EndDialog() will end the program!
    if (MessageBox(L"Are you sure you wish to exit the program?",
        L"Attention!", MB_YESNO) == IDYES)
    {
        EndDialog(-1);
    }
}


void CMFCApplication4Dlg::OnHelpInstructionsmanual()
{
    // This function opens the Instructions Manual document

    // path to Instructions Manual document
    std::string path = "Manuals\\MANUAL_SurfaceTopographySystem.pdf";
    // convert path string to LPCWSTR
    std::wstring stemp = s2ws(path);
    LPCWSTR Manual_path = stemp.c_str();

    if ((int)ShellExecute(0, L"open", Manual_path, 0, 0, SW_SHOWDEFAULT) <= 32)
    {
        LPCWSTR viesti = L"Error!";
        ::MessageBox(NULL, viesti, NULL, MB_OK);
    }
}
```

```
void CMFCApplication4Dlg::OnHelpCanonmanual()
{
    // This function opens the Canon Manual document

    // path to Canon Manual document
    std::string path = "Manuals\\MANUAL_eos-rebelsl1-100d-im-en.pdf";
    // convert path string to LPCWSTR
    std::wstring stemp = s2ws(path);
    LPCWSTR Manual_path = stemp.c_str();

    if ((int)ShellExecute(0, L"open", Manual_path, 0, 0, SW_SHOWDEFAULT) <= 32)
    {
        LPCWSTR viesti = L"Error!";
        ::MessageBox(NULL, viesti, NULL, MB_OK);
    }
}

void CMFCApplication4Dlg::OnOpenExistingproject()
{
    // This function will open existing project for user.
    // Folder explorer is deployed.
    // If folder is selected check if autosave file exists.
    // If autosave file exists, transfer data.

    // Initializing folder picker dialog
    CFolderPickerDialog folderPickerDialog(NULL, OFN_FILEMUSTEXIST | OFN_ALLOWMULTISELECT
                                           | OFN_ENABLESIZING, this, sizeof(OPENFILENAME));
    CString folderPath;
    // Creating path to SaveFiles folder where all projects are saved
    TCHAR szDirectory[MAX_PATH];
    ::GetCurrentDirectory(sizeof(szDirectory) - 1, szDirectory);
    folderPath = szDirectory;
    folderPath += _T("\\SaveFiles");
    folderPickerDialog.m_ofn.lpstrInitialDir = folderPath;

    if (folderPickerDialog.DoModal() == IDOK)   // Launch folder picker, if "select folder" clicked...
    {
        // Extracting path of chosen folder
        POSITION pos = folderPickerDialog.GetStartPosition();
        folderPath = folderPickerDialog.GetNextPathName(pos);

        nimi->SetWindowText(PathFindFileName(folderPath));  // Setting name to editfield nimi

        // Building path to autosave.xml
        CString rootfile = _T("SaveFiles\\");   // Two slashes to tell windows there is only one slash.
        CString oma;
        CString asFile = _T("\\autosave.xml");
        nimi->GetWindowTextW(oma);
        rootfile += oma;
        setName(rootfile);          // Setting the project path.
        rootfile += asFile;         // rootfile now "SaveFiles\\ + oma + \\autosave.xml"
```

4(7)

```
std::ifstream inputFile(rootfile);  // Initializing inputfilestream
std::string line;

if (inputFile.is_open())
{
    getline(inputFile, line);   // read line off
    getline(inputFile, line);   // read line off
    getline(inputFile, line);   // read line off

    while (getline(inputFile, line))    // While lines available, get a line
    {
        std::istringstream ss(line);    // Initializing stringstream with line from autosave.xml
        std::string name;
        int var;
        ss >> name >> var;              // Taking parameters from the line in stringstream

        // Updating project variables with autosave.xml file parameters

        // ints
        if (name == "piccountf_:"){
            piccountf_ = var;
        }
        else if (name == "piccounts_:"){
            piccounts_ = var;
        }
        else if (name == "piccountl_:"){
            piccountl_ = var;
        }
        else if (name == "piccount_sa_:"){
            piccount_sa_ = var;
        }
        else if (name == "piccountf_total_:"){
            piccountf_total_ = var;
        }
        else if (name == "piccounts_total_:"){
            piccounts_total_ = var;
        }
        else if (name == "piccountl_total_:"){
            piccountl_total_ = var;
        }
        else if (name == "piccount_sa_total_:"){
            piccount_sa_total_ = var;
        }
        else if (name == "resetcount_:"){
            resetcount_ = var;
        }
```

```
                // bools
                else if (name == "focus_c_taken_:"){
                    focus_c_taken_ = var;
                }
                else if (name == "scale_c_taken_:"){
                    scale_c_taken_ = var;
                }
                else if (name == "lighting_c_taken_:"){
                    lighting_c_taken_ = var;
                }
                else if (name == "take_calibration_images_:"){
                    take_calibration_images_ = var;
                }
                else if (name == "calibration_images_taken_:"){
                    calibration_images_taken_ = var;
                }
                else if (name == "calibration_images_present_:"){
                    calibration_images_present_ = var;
                }

            }
            inputFile.close();

            // Activation of controls in the dialog...
            led_a->EnableWindow(true);
            led_b->EnableWindow(true);
            led_c->EnableWindow(true);
            //nappi1->EnableWindow(true);
            nappi2->EnableWindow(false);
            nappi4->EnableWindow(true);
            //nappi5->EnableWindow(true);
            //nappi6->EnableWindow(true);
            //nappi7->EnableWindow(true);
            //nappi8->EnableWindow(true);
            nappi9->EnableWindow(true);
            nappi13->EnableWindow(true);
            nappi14->EnableWindow(true);
            nimi->EnableWindow(false);          // Disable step 1 textbox
            camera_h->EnableWindow(false);
            A_h->EnableWindow(false);
            B_h->EnableWindow(false);
            C_h->EnableWindow(false);
            setStage("");
        }
        else{
            LPCWSTR viesti = L"Cannot open autosave.xml file!";
            ::MessageBox(NULL, viesti, NULL, MB_OK);
            resetDlg();     // Reset status for main dialog
        }
    }
}
```

6(7)

```
void CMFCApplication4Dlg::OnOpenProjectsfolder()
{
    // This function opens a browser in dynamic path "//..SaveFiles"

    ShellExecute(NULL, L"open", L"SaveFiles", NULL, NULL, SW_SHOWDEFAULT);
}

void CMFCApplication4Dlg::OnFocuscalibLed()
{
    // This function sets a check to LED and removes a check from Lasers
    // in the menuitems under Settings/Focus Calib.

    CMenu* mmenu = &_Menu;
    mmenu->CheckMenuItem(ID_FOCUSCALIB_LED, MF_CHECKED | MF_BYCOMMAND);
    mmenu->CheckMenuItem(ID_FOCUSCALIB_LASER, MF_UNCHECKED | MF_BYCOMMAND);
    focusMode = LEDFOCUS;
}


void CMFCApplication4Dlg::OnFocuscalibLaser()
{
    // This function sets a check to Lasers and removes a check from LED
    // in the menuitems under Settings/Focus Calib.

    CMenu* mmenu = &_Menu;
    mmenu->CheckMenuItem(ID_FOCUSCALIB_LED, MF_UNCHECKED | MF_BYCOMMAND);
    mmenu->CheckMenuItem(ID_FOCUSCALIB_LASER, MF_CHECKED | MF_BYCOMMAND);
    focusMode = LASERFOCUS;
}


void CMFCApplication4Dlg::OnLasersOn()
{
    // This funktion turns lasers on and checks the Lasers/On button
    // and unchecks Lasers/Off button
    if (!_testLaserOn)
    {
        CMenu* mmenu = &_Menu;
        mmenu->CheckMenuItem(ID_LASERS_ON, MF_CHECKED | MF_BYCOMMAND);
        mmenu->CheckMenuItem(ID_LASERS_OFF, MF_UNCHECKED | MF_BYCOMMAND);
        _testLaserOn = true;
        ledControllerWrite("$FSON!");

    }
}
```

```
void CMFCApplication4Dlg::OnLasersOff()
{
    // This funktion turns lasers off and checks the Lasers/Off
    // and unchecks Lasers/On
    if (_testLaserOn)
    {
        CMenu* mmenu = &_Menu;
        mmenu->CheckMenuItem(ID_LASERS_OFF, MF_CHECKED | MF_BYCOMMAND);
        mmenu->CheckMenuItem(ID_LASERS_ON, MF_UNCHECKED | MF_BYCOMMAND);
        _testLaserOn = false;
        ledControllerWrite("$FOFF!");

    }
}


int CMFCApplication4Dlg::PreTranslateMessage(MSG* pMsg)
{
    m_pToolTipCtrl.RelayEvent(pMsg);

    return CDialog::PreTranslateMessage(pMsg);

}



/// Graphics customisation ///

BOOL CMFCApplication4Dlg::OnEraseBkgnd(CDC* pDC)
{
    CDC dcMemory;

    dcMemory.CreateCompatibleDC(pDC);
    CBitmap* pOldBitmap = dcMemory.SelectObject(&Background);
    CRect rcClient;
    GetClientRect(&rcClient);
    const CSize& sBitmap = BitmapSize;
    pDC->BitBlt(0, 0, sBitmap.cx, sBitmap.cy, &dcMemory, 0, 0, SRCCOPY);
    dcMemory.SelectObject(pOldBitmap);

    return TRUE;
    //return CDialog::OnEraseBkgnd(pDC); Remove and return TRUE
}


void CMFCApplication4Dlg::OnDestroy()
{
    CDialog::OnDestroy();

    Background.DeleteObject(); // Delete Background Bitmap
    BrushHol.DeleteObject();
}

HBRUSH CMFCApplication4Dlg::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
    HBRUSH hbr = CDialog::OnCtlColor(pDC, pWnd, nCtlColor);

    if (pWnd->GetDlgCtrlID() == IDC_STATIC)
    {
        pDC->SetTextColor(RGB(0, 0, 0));
        pDC->SetBkMode(TRANSPARENT);
        //pDC->SetBkMode(OPAQUE);
        return (HBRUSH)GetStockObject(NULL_BRUSH);
    }

    // TODO:  Return a different brush if the default is not desired
    return hbr;
}

/// END Graphics customisation END ///
```

Liite 5. Histogrammit tuottavan Hist.exe-ohjelman lähdekoodit

```
function [ output_arg ] = Hist( input_args )
    % Histogrammi .CR2-raakakuvasta
    % MM 15.6.2017
    % muokattu: AS 16.6.2017

    % Fileparts separates path to parts.
    % pathstr: path to folder
    % name:    name of the file in the folder
    [pathstr,name,ext] = fileparts(input_args); %input_args is a filepath

    % readraw palauttaa 3476 x 5208 uint16
    img = readraw(input_args);
    %figure, imagesc(img); axis image; %colorbar
    % Jos tätä zoomaat reilusti, näkyy Bayer-kuvio eli kuva on ihan
    % ruudullinen. Se johtuu siitä, että joka toinen havainto (pikseli) on
    % vihreästä valosta, ja joka neljäs punaisia ja joka neljäs sinisiä.

    % demosaic interpoloi kaikkiin pikseleihin kaikki väriarvot, näin
    % RGB-array on 3476 x 5208 x 3 uint16 (jossa kolmas ulottuvuus käsittää
    % värit Red, Green, Blue)
    RGB = demosaic(img,'rggb');

    % teillä on valkoiset valot, joten voisi ottaa keskiarvon kaikista väreistä
    % (tulee 3476 x 5208 double)
    I = mean(RGB,3);
     %figure; imagesc(I); axis image; colormap gray; colorbar

    figname = 'Histogram_';
    hist_title = strcat(figname,name); % title for histogram
    N = numel(I); % noin 18 miljoonaa pikseliä
    Nh = 4000; % histogrammipylväiden lukumääräksi suunnilleen sqrt(N)
    % I(:) tarkoittaa että I, joka oli alunperin iso 3-ulotteinen array,
    % luetaan pitkäksi vektoriksi. hist-funktio haluaa inputin sellaisena.
    [h,x] = hist(I(:),Nh);
    figure('Name',hist_title,'NumberTitle','off'), plot(x,h) % käyränä
    % figure, bar(x,h) % pylväinä
    grid on % jos et halua ruudukkoa, tämän voi jättää pois, tai 'grid off'
    xlabel('Intensity'); ylabel('Pixel count')

    % Kuvan tulostus
    % - jälkimmäinen parametri (r400) on resoluutioon liittyvä (400 dpi)
    Printopts = {'-djpeg' '-r400'};

    % creating a folder to save the plot in
    savefolder = 'HistogramsRGB';
    savepath = fullfile(pathstr,savefolder);
    savename = fullfile(savepath,name);
    mkdir(pathstr, savefolder); % creating the folder

    % 'gcf' tarkoittaa Get Current Figure, eli aktiivisena oleva Figure lähtee
    % printtaukseen
    print(gcf,savename,Printopts{:})

    output_arg = 0;
end
```