TAMPEREEN
AMMATTIKORKEAKOULU

VIISTOVALOKUVAUSJÄRJESTELMÄN KÄYTTÖLIITTYMÄ

Aleksi Tapola

Opinnäytetyö
Joulukuu 2015
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

TAPOLA ALEKSI:
Viistovalokuvausjärjestelmän käyttöliittymä

Opinnäytetyö 33 sivua, joista liitteitä 14 sivua
Joulukuu 2015

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa käyttöliittymä Tampereen ammattikorkeakoululla kehitteillä olevaan viistovalokuvauslaitteeseen. Käyttöliittymä kehitettiin helpottamaan pintatopografiamallinnuksessa tarvittavien näytekuvasarjojen ottamista, sekä jatkokäsittelyssä tarvittavien kuvausparametrien tallentamista varten. Käyttöliittymä kehitettiin yhteensopivaksi viistovalokuvauslaitteeseen jo suunnitellun valo-ohjaimen ja laitteessa käytetyn järjestelmäkameran kanssa. Käyttöliittymää käytetään vain näytekuvasarjojen ottamiseen, ja varsinainen pintatopografiamallinnus tapahtuu Matlab-ohjelmalla.

Käyttöliittymä kehitettiin C++ -ohjelmointikielellä. Kehityksessä käytettiin Microsoft Visual Studio 2012 -kehitysympäristöä, Microsoft Forms -kehitystyökaluja, sekä järjestelmäkameravalmistaja Canonin tarjoamia ohjelmointikirjastoja, jotka tarjoavat ohjelmointirajapinnan kyseisen valmistajan järjestelmäkameroille.

Käyttöliittymä saatiin siihen kuntoon, että näytekuvasarjojen ottaminen sen avulla on mahdollista. Käyttöliittymän helpon käytettävyyden kannalta käyttöliittymästä puuttuu ominaisuuksia, kuten uuden sarjan aloittaminen.

Asiasanat: ohjelmistokehitys, sulautetut järjestelmät, viistovalokuvaus

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Name of the Degree Programme
Name of the Option

TAPOLA ALEKSI:
User interface for photometric stereo equipment

Bachelor's thesis 33 pages, appendices 14 pages
December 2015

The purpose of this thesis was to design and implement a user interface for photometric stereo equipment under development at the Tampere University of Applied Sciences. User interfaces purpose was to make taking series of pictures needed for surface topology modeling easier, and to save picture parameters for post processing. The user interface was designed to be compatible with the already implemented light controller and with the digital single-lens reflex camera used in the equipment. The user interface is only used to take series of pictures, and the actual surface topography modelling is done using Matlab.

The user interface was developed using C++ programming language. Microsoft Visual Studio 2012 integrated development environment, Microsoft Forms development tools and libraries offered by the single-lens reflex camera manufacturer Canon, which offers application programming interface for the said manufacturer's cameras, were used in the development.

The current version of the user interface allows user to take picture series using the interface. However, to make the experience easy and pleasing, few features are needed in the further development including the ability to start a new picture serie.

Key words: software development, embedded systems, photometric stereo

SISÄLLYS

LYHENTEET JA TERMIT

SDK	Software development kit
EVF	Electronic view finder
XML	Extensible markup language

1 JOHDANTO

Viistovalokuvauslaite on optinen mittalaite, jonka avulla voidaan tutkia pinnan muotoja, eli pintatopografiaa. Tässä työssä kuvaillaan Tampereen ammattikorkeakoulun, Tampereen teknillisen yliopiston, Hämeen ammattikorkeakoulun ja Valmet automation oyj:n yhteistyönä kehittämän viistovalokuvauslaitteiston käyttöliittymän suunnittelua ja kehitystä. Kyseinen viistovalokuvauslaite on suunniteltu erityisesti paperien ja kartonkien pintatopografioita varten.

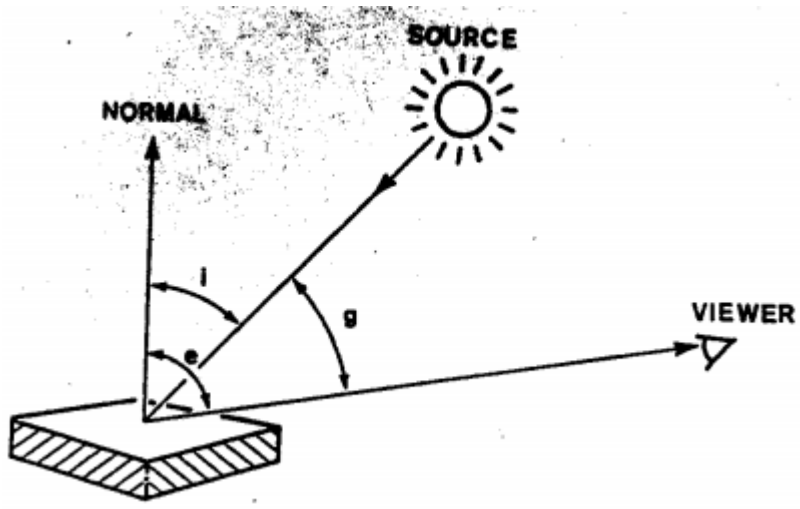
Käyttöliittymän välttämättömiin ominaisuuksiin kuului kyky ottaa valokuvia viistovalokuvauslaitteeseen kuuluvalla järjestelmäkameralla sekä kommunikointikyky laitteen valo-ohjaimen kanssa. Varsinainen valokuvista tapahtuva pintatopografian määrittäminen tapahtuu Matlab-ohjelmalle kehitetyillä algoritmeilla.

Käyttöliittymä toteutettiin käyttäen C++ -ohjelmointikieltä, ja kehitysympäristöksi valittiin Microsoft Visual Studio -ohjelmisto. Microsoft Visual Studio -ohjelmisto sisältää graafisen suunnittelunäkymän Microsoft Forms-ohjelmien kehittämiseen, mitä hyödynnettiin käyttöliittymän suunnittelussa ja toteutuksessa.

2 VIISTOVALOKUVAUS

Viistovalokuvaus, josta käytetään myös nimeä fotometrinen stereo, on optinen menetelmä, jonka avulla voidaan selvittää halutun pinnan normaali kussakin kuvapisteessä, ottamalla kohteesta valokuvia eri suunnista ja kulmista valaistuna. Kohteesta otetuista kuvista voidaan määrittää kohteen pinnan normaali halutussa pisteessä kuvan varjostuksia hyödyntäen.

Vaikka kappaleen heijastaman valon hajontaan vaikuttaa kappaleen pinnan optiset ominaisuudet, kappaleen pinnan mikrorakenne, sekä useat valaistusparametrit, useiden pintojen tiettyyn suuntaan heijastaman valon määrään vaikuttaa merkittävästi ainoastaan pinnan normaali. Pinnan heijastusominaisuudet voidaan esittää kolmen kulman funktiona $\phi(i, e, g)$, jossa i on pinnan normaalin ja valonlähteen välinen kulma, e on pinnan normaalin ja katsojan välinen kulma ja g on valonlähteen ja katsojan välinen kulma (kuva 1). Heijastusominaisuuksia kuvaavan funktiona avulla voidaan muodostaa heijastuskartta, joka vastaa pinnan gradienttia kussakin kuvapisteessä. (Woodham 1980, 1)



KUVA 1. Heijastusominaisuuksia kuvaavat kulmat (Woodham 1980, 2)

Heijastuskartan avulla voidaan muodostaa yhtälö (1), jossa I on otettu kuva, x on otetun kuvan x-koordinaatti, y on otetun kuvan y-koordinaatti, R on kuvan heijastuskartta, p on kuvan heijastuskartan p-koordinaatti ja q on kuvan heijastuskartan q-koordinaatti. Yhdestä kuvasta ja heijastuskartasta voidaan tehdä joitakin päätelmiä kappaleen muodosta kuvan intensiteetin avulla, mutta tarkempaan analyysiin tarvitaan vähintään kolme ku-

vaa, joissa esimerkiksi valolähteen ja katsojan välinen kulma on pidetty samana. (Woodham 1980, 2–3)

$$I(x, y) = R(p, q) \tag{1}$$

Jos muodostetaan vektori yhtälön 2 mukaan, missä I on muodostettava vektori, I_1 on ensimmäisen kuvan intensiteetti jossakin pisteessä, I_2 on toisesta suunnasta valaistuna otetun kuvan intensiteetti samassa pisteessä kuin I_1 ja I_3 on kolmannesta suunnasta valaistuna otetun kuvan intensiteetti samassa pisteessä kuin I_1 ja I_2, voidaan kirjoittaa yhtälö (3), missä I on intensiteettivektori, ϱ on pisteen heijastuskerroin, N on valaistussuuntavektoreista koostettu matriisi ja n on pinnan yksikkönormaalivektori, josta on ratkaistavissa sekä kyseisen pisteen heijastuskerroin että pinnan normaali kyseisessä pisteessä. (Woodham 1980, 3–4)

$$I = [I_1 \quad I_2 \quad I_3] \tag{2}$$

$$I = \varrho N n \tag{3}$$

3 VALO-OHJAIN

Viistovalokuvauslaitteen valo-ohjaimen suunnittelu ja toteutus teetätettiin Tampereen ammattikorkeakoulussa opinnäytetyönä. Ohjaimen suunnittelusta ja toteutuksesta vastasi Panu Vuorenmaa.

Valo-ohjain kommunikoi tietokoneen kanssa virtuaalisen sarjaportin rajapinnan välityksellä (Vuorenmaa 2015, 35). Tietokoneella suoritettava käyttöliittymä antaa valo-ohjaimelle kussakin tilanteessa halutun kuusimerkkisen komennon, joka sisältää aloitus- ja lopetusmerkin. Taulukossa 1 on esitetty valo-ohjaimen ohjauskomennot (Vuorenmaa 2015, 36). Valo-ohjain kuittaa jokaisen käskyn merkkijonolla ”ACK”.

TAULUKKO 1. Valo-ohjaimen ohjauskomennot

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)
$Fxxx!	Tarkennuksessa käytettävien lasereiden ohjaus	SET Asettaa kanavassa olevan laserin 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).

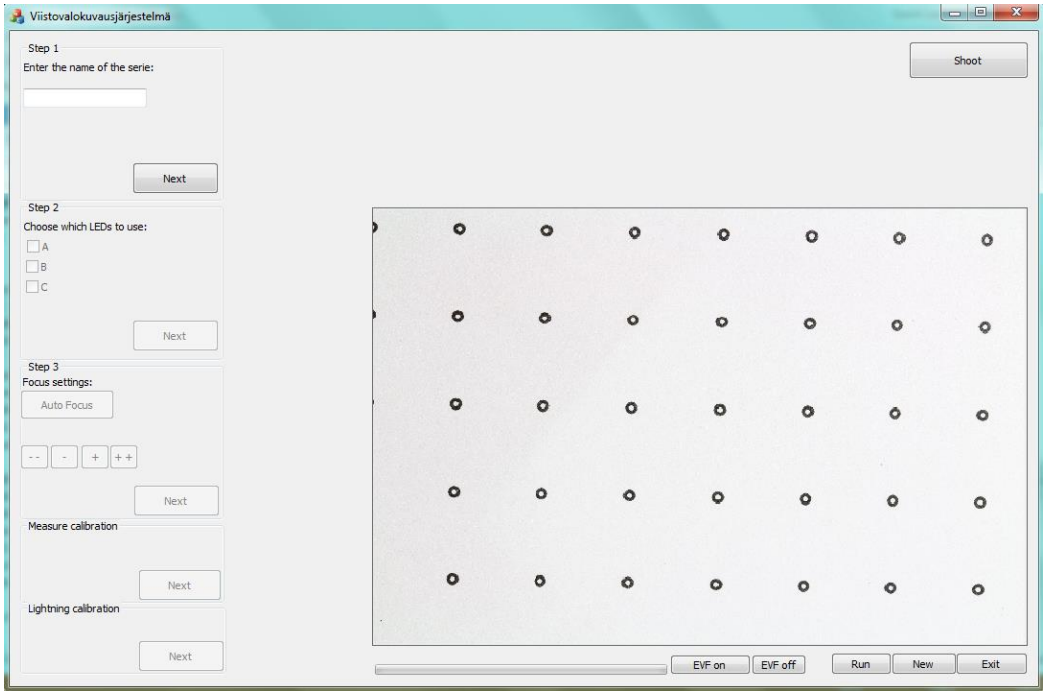
4 CANON SDK-KEHITYSALUSTA

Canon SDK -kehitysalusta on Canon inc:n tarjoama kehitystyökalupaketti, jonka avulla voidaan kehittää ohjelmia, jotka kykenevät ohjaamaan Canonin valmistamia järjestelmäkameroita. Kehitystyökalupaketti sisältää ohjelmointikirjastot C sharp-, Visual basic- ja C++ -ohjelmointikielille, sekä esimerkkisovelluksen, joka havainnollistaa kirjastojen käyttöä.

Canon SDK-kehitysalusta tarjoaa ohjelmistokehittäjälle helpon rajapinnan Canon järjestelmäkameroiden ohjaamiseen oman koodin välillä. Kehitysalustan avulla ohjelmistokehittäjä voi etähallita kameraa, ja hallinnoida otettuja kuvia. Kuvat on mahdollista ladata kamerasta ns. raakamuotoisina, eli minimaalisesti käsiteltyinä.

5 KÄYTTÖLIITTYMÄ

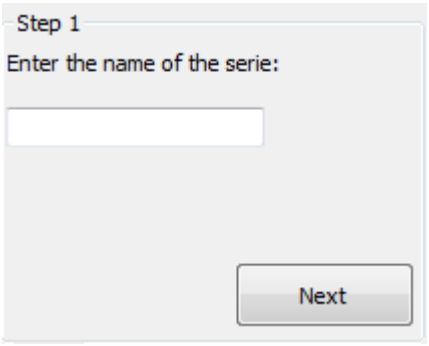
Viistovalokuvauslaitteen käyttöliittymää käytetään pintatopografian määritykseen tarvittavien valokuvien ottamiseen. Käyttöliittymä ottaa vähintään 7 kuvaa yhtä näytettä kohden, eli tarkkuuskalibrointikuvan, valaistuskalibrointikuvan, mittakaavakalibrointikuvan sekä vähintään 4 kuvaa, joissa kohdetta on valaistu eri suunnista. Kuvassa 2 on esitetty ruutukaappaus käyttöliittymän koko näkymästä.



KUVA 2. Ruutukaappaus käyttöliittymästä

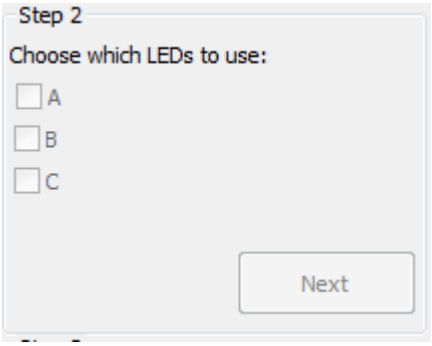
5.1 Kuvien ottaminen käyttöliittymän avulla

Käyttöliittymän toiminta etenee vaiheittain. Ensimmäisessä vaiheessa käyttäjää kehotetaan antamaan nimi kuvattavalle sarjalle. Käyttäjän antaman nimen perusteella ohjelma luo samannimisen alihakemiston hakemistoon, jossa suoritettava binääritiedosto on. Mikäli samanniminen alihakemisto on jo olemassa, antaa ohjelma asiasta virheilmoituksen englanniksi, eikä käyttöliittymä etene seuraavaan vaiheeseen. Kuvassa 3 on esitetty rajattu ruutukaappaus käyttöliittymän ensimmäiseen vaiheeseen liittyvästä osasta.



KUVA 3. Ruutukaappaus käyttöliittymän ensimmäisestä vaiheesta

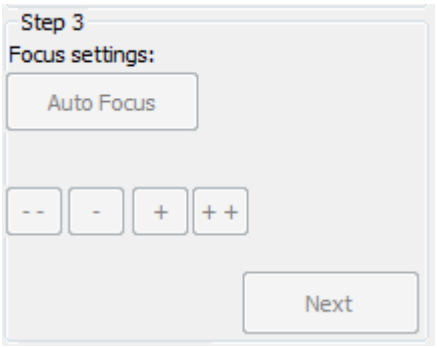
Toisessa vaiheessa käyttäjä valitsee kuvasarjassa käytettävän valaistuksen, valitsemalla halutut valintaruudut. Valintaruutujen vieressä on merkinnät A, B ja C. Valintaruutua A vastaavat valot ovat laitteessa näytteeseen nähden matalimmalla olevat led-valot, valintaruutua B vastaavat led-valot ovat keskikorkeudella paperiin nähden ja valintaruutua C vastaavat led-valot taas korkeimmalla olevat led-valot näytteeseen nähden. Käyttäjän tulee valita 1 - 3 valintaruutua edetäkseen seuraavaan vaiheeseen. Kuvassa 4 on esitetty rajattu ruutukaappaus käyttöliittymän toiseen vaiheeseen liittyvästä osasta.



KUVA 4. Ruutukaappaus käyttöliittymän toisesta vaiheesta

Käyttöliittymän kolmannessa vaiheessa käyttäjän tulee tarkentaa kamera näytteeseen. Siirryttäessä kolmanteen vaiheeseen ohjelma antaa valo-ohjaimelle käskyn käynnistää laserristikko, joka helpottaa tarkennusta, etenkin jos tarkennusta tehtäessä laitteen ovi on kiinni, sekä antaa kameralle käskyn käynnistää EVF-toiminto ja näyttää sen näkymän käyttöliittymän kuvalaatikossa. Kun käyttäjä painaa aktivoitunutta Next-painiketta, antaa ohjelma järjestelmäkameralle käskyn ottaa valokuva, sekä siirtää otetun tarkennuskalibrointikuva ensimmäisessä vaiheessa luotuun alihakemistoon jatkokäsittelyä

varten nimellä ”focuscalib.CR2”. Kuvassa 5 on esitetty rajattu ruutukaappaus käyttöliittymän kolmannesta vaiheeseen liittyvästä osasta.



KUVA 5. Ruutukaappaus käyttöliittymän kolmannesta vaiheesta

Neljänteen vaiheeseen siirryttäessä ohjelma kehottaa käyttäjää asettamaan mittakaavakalibrointikuvan laitteeseen, ja valo-ohjain sammuttaa tarkennuksessa käytetyn laserristikon. Kun käyttäjä on asettanut mittakaavakalibrointikuvan viistovalokuvauslaitteeseen ja painaa aktivoitunutta Next-painiketta, antaa käyttöliittymä järjestelmäkameralle käskyn ottaa valokuva, ja siirtää otetun kuvan ensimmäisessä vaiheessa luotuun alihakemistoon nimellä ”measurecalib.CR2”.

Kun käyttöliittymä etenee viidenteen vaiheeseen, kehottaa ohjelma käyttäjää asettamaan valaistuskalibrointikohteen laitteeseen. Käyttäjän painaessa aktivoitunutta Next-painiketta, käskee ohjelma kameraa lopettamaan EVF-toiminnon ja ottamaan kuvan.

Käyttöliittymän viimeinen vaihe on varsinaisen näytteen kuvaaminen. Käyttäjän painaessa Run-painiketta, ohjelma ottaa 4, 8 tai 12 kuvaa käyttäjän valitsemien valokerrosten mukaan. Kuvasarjan ottamisen aikana käyttöliittymän alareunassa sijaitseva edistyspalkki osoittaa sarjan edistymistä. Edistyspalkki päivittyy aina, kun kamerasta on ladattu uusi kuva.

Käyttöliittymä luo kuvasarjasta meta-tieto -tiedoston jatkokäsittelyä varten. Tiedosto luodaan ensimmäisessä vaiheessa luotuun kansioon nimellä ”meta.xml”. Jatkokäsittelyssä jokaisesta kuvasta tulee tietää ainakin kulloinkin käytetyn led-valon x-, y- ja z-koordinaatit kuvan keskipisteeseen nähden, kuvan ottamisessa käytetty valotusaika sekä kuvan ottamisessa käytetty kameran aukon koko. Käyttöliittymän tämänhetkinen versio luo meta-tieto -tiedoston ja luo jokaisesta otetusta kuvasta image-tietueen, mutta merkit-

see led-valon x-, y- ja z-koordinaattien kohdalle vain ”aaa”, ”bbb” tai ”ccc” käytetyn valokerroksen mukaan. Tiedostoon ei tule merkintää valotusajasta tai aukon koosta. Kuvassa 6 on esitetty esimerkki a-valokerroksella kuvatun kuvasarjan meta-tiedot sisältävästä XML-tiedostosta.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <image_data>
3      <image>
4          <X value="aaa"/>
5          <Y value="aaa"/>
6          <Z value="aaa"/>
7      </image>
8      <image>
9          <X value="aaa"/>
10         <Y value="aaa"/>
11         <Z value="aaa"/>
12     </image>
13     <image>
14         <X value="aaa"/>
15         <Y value="aaa"/>
16         <Z value="aaa"/>
17     </image>
18     <image>
19         <X value="aaa"/>
20         <Y value="aaa"/>
21         <Z value="aaa"/>
22     </image>
23 </image_data>
```

KUVA 6. XML-muotoinen meta-tieto -tiedosto

5.2 Ohjelmakoodi

Ohjelman pohja luotiin Visual Studio 2012:n MFC Application wizardin avulla, joka luo tarvittavat tiedostot ja koodin graafisen, dialog-pohjaisen sovelluksen tarpeisiin. Luotuun sovellukseen lisättiin tarvittavat painikkeet, tekstikentät, valintaruudut ja muut elementit graafisen suunnittelutyökalun avulla, sekä tehtiin luotuihin kooditiedostoihin tarvittavat muutokset ja lisäykset halutun toiminnallisuuden saavuttamiseksi. Suurimmat muutokset ja lisäykset tehtiin MCFApplication4Dlg.cpp-tiedostoon (ks. Liite 1).

5.2.1 Elementtien käyttäminen

Jotta graafisen suunnittelutyökalun avulla luotuja elementtejä voidaan hyödyntää omassa ohjelmakoodissa, voidaan luoda elementtiä vastaava osoitin. Elementin osoite saadaan kutsumalla osoitteen palauttavaa funktiota GetDlgItem, ja antamalla parametriksi graafisen suunnittelutyökalun elementille antama ID-tunnus, joka koostuu elementin tyypistä ja juoksevasta numerosta (kuva 7). Osoittimen välityksellä voidaan käyttää myös kullekin elementille ominaisia ominaisuuksia.

```
CButton *nappi14 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON14));   // Create a pointer to the
nappi14->EnableWindow(true);                                               // button with ID "IDC_BUTTON14
```

KUVA 7. Elementin aktivoiminen luodun osoittimen avulla

5.2.2 Kansion luominen

Käyttöliittymän ensimmäisessä kohdassa luodaan kansio, joka nimetään käyttäjän syötteen mukaan. Ohjelmakoodissa luodaan osoitin, joka osoittaa tekstikenttään, ja luetaan osoittimen avulla kentässä oleva teksti CString-tyyppiseen muuttujaan (kuva 8). Kun käyttäjän syöte on luettu muuttujaan, yritetään luoda samanniminen kansio käyttäen filesystem-kirjaston CreateDirectory-funktiota. Jos kansion luominen onnistuu, siirrytään vaiheeseen kaksi, muuten näytetään käyttäjälle virheilmoitus ja pysytään vaiheessa yksi.

```
CString oma;
nimi->GetWindowTextW(oma);
setName(oma);
if(!getName().IsEmpty())
{
    if(CreateDirectory(getName(),0))        // Try to create the folder.
    {
        nappi3->EnableWindow(true);         // Activation of step 2 controls
        led_a->EnableWindow(true);
        led_b->EnableWindow(true);
        led_c->EnableWindow(true);
        nimi->EnableWindow(false);          // Disable step 1 textbox

    }
    else
    {
        LPCWSTR viesti = L"Directory already exists";   // CreateDirectory function return false,
        ::MessageBox(NULL,viesti,NULL,MB_OK);           // if creation fails, eg. Directory already
    }                                                   // exists.
}
```

KUVA 8. Kansion luominen

5.2.3 Kameran ohjaaminen

Koska käyttöliittymän pohjana on käytössä Canon SDK -kehitysalustan esimerkkisovellus ja sen luokkarakenne, voidaan kameralle antaa selkokielisiä komentoja. MFCApplication4Dlg-luokka periytyy esimerkkisovelluksen ActionSource-luokasta, joten MFCApplication4Dlg-luokalle voidaan asettaa string-tyyppinen komento, joka periytyy ActionSource-luokasta. Kun string-tyyppinen komento on asetettu, voidaan kutsua niin ikään ActionSource-luokasta periytyvää fireEvent-funktiota ilman parametreja. Kuvassa 9 on esitetty valokuvan ottamiseen tarvittava ohjelmakoodi MFC4ApplicationDlg.cpp-tiedostosta. Kuvassa 9 esitetyllä menetelmällä kameralle voidaan antaa myös muita komentoja, kuten tarkennuksen säätöön liittyvät komennot sekä EVF:n käynnistäminen ja lopettaminen.

```
this->setActionCommand("TakePicture");      // Before calling fireEvent function, proper ActionCommand
this->fireEvent();                          //  must be set. "TakePicture" makes the camera take
                                            //  a picture. FireEvent excecutes the set command.
```

KUVA 9. Valokuvan ottaminen ohjelmakoodissa

5.2.4 Meta-tiedon kirjoittaminen

Käyttöliittymä kirjoittaa jatkokäsittelyssä tarvittavia tietoja erilliseen XML-tiedostoon, joka luodaan samaan alihakemistoon, johon kuvasarjan valokuvat siirretään. Tiedosto kirjoitetaan käyttäen fstream-kirjaston ofstream-luokkaa. Kuvassa 10 on esitetty XML-tiedoston luominen ja kahden ensimmäisen rivin kirjoittaminen.

```
std::ofstream *kirjuri = new std::ofstream();   // Create a pointer to new ofstream object
kirjuri->open(getName()+(CString)"\\meta.xml"); // Create the meta file in requested location

if(kirjuri->is_open())                          // Testing if the file creation was succesfull
{
    kirjuri->write("<?xml version=\"1.0\" encoding=\"utf-8\"?>\n",39); // Writing the first line
    kirjuri->write("<image_data>\n",13);        // Writing the second line
    kirjuri->close();                           // Closing the file for now
}

if(kirjuri != NULL)
{
    delete kirjuri;                             // Delete the ofstream object
    kirjuri = NULL;                             // and set the pointer to NULL.
}
```

KUVA 10. XML-tiedoston luominen ja käsittely ohjelmakoodissa

5.2.5 Valo-ohjaimen ohjaaminen

Käyttöliittymä antaa komentoja valo-ohjaimelle virtuaalisen sarjaportin välityksellä. Kuvassa 11 on esitetty funktio, jonka avulla komennot lähetetään valo-ohjaimelle, kunhan oikea sarjaportti on määritetty koodissa aikaisemmin setComport-funktion avulla.

```
void CMFCApplication4Dlg::ledControllerWrite(std::string command)
{
    std::wstringstream wss;
    unsigned char commandchar[6];

    for(int i = 0; i < 6; i++)                  // A loop to copy the command string to a commandchar array
    {
        commandchar[i] = command.at(i);
    }

    wss << getComport().c_str();                // Reading the desired comport to wss with getComport
    wss.str().c_str();                          // getter. Comport must be set earlier with setComport.

    SerialPort *omaportti = new SerialPort();   // Creation of SerialPort object
    omaportti->connect((LPWSTR)wss.str().c_str());  // Open the desired comport with connect function
    omaportti->sendArray(commandchar,6);        // Sending commandchar array
    omaportti->disconnect();                    // Closing the comport
    if(omaportti != NULL)
    {
        delete omaportti;                       // Deleting the SerialPort object
        omaportti = NULL;                       // and setting the pointer to NULL
    }
}
```

KUVA 11. Funktio, jonka avulla annetaan komentoja valo-ohjaimelle

6 POHDINTA

Vaikka Tampereen ammattikorkeakoulun viistovalokuvauslaite onkin periaatteessa käyttökelpoinen, on sen tämänhetkisessä käyttöliittymässä vielä paljon sujuvan käytön kannalta oleellisia puutteita. Kuvasarjan ottamista voitaisiin parantaa myös useilla lisäominaisuuksilla, kuten näyttämällä käyttäjälle kuvan histogrammi jokaisen otetun kuvan jälkeen, jolloin kokenut käyttäjä osaisi heti päätellä, onko kuva yli- tai alivalottunut.

Meta-tietojen kirjoittamista suorittava ohjelmakoodi on keskeneräinen, eikä tiedostoon tällä hetkellä kirjata keskeisten kuvausparametrien arvoja, ja valolähteiden sijaintitiedot ovat puutteelliset ja hankalasti muutettavissa. Ohjelmakoodissa on olemassa logiikka, jolla ohjelma osaa päätellä, minkä valokerroksen valolähteellä kohde on valaistu kuvan meta-tietoa kirjoitettaessa, mutta logiikka valokerroksen yksittäisen valolähteen tunnistamiseksi puuttuu.

Valo-ohjaimen käyttämä virtuaalinen sarjaportti on tällä hetkellä asetettu arvoon ”COM3” ohjelmakoodissa. Sarjaportin vaihtaminen vaatii muutoksen tekemistä ohjelmakoodiin ja koodin uudelleen kääntämistä konekielelle. Ohjelmakoodi sisältää funktion, jonka tarkoitus on etsiä valo-ohjaimen käyttämä sarjaportti lähettämällä erinumeroisiin sarjaportteihin valo-ohjaimen tuntemia käskyjä ja odottamalla valo-ohjaimen vahvistusviestiä, mutta funktion käytöstä luovuttiin sen epäluotettavan toiminnan takia.

Käyttöliittymässä näkyvä New-painike ei ole toiminnassa käyttöliittymän tämänhetkisessä versiossa. Painikkeen on tarkoitus palauttaa käyttöliittymä ensimmäiseen vaiheeseen uuden kuvasarjan ottamista varten, sekä kysyä käyttäjältä, haluaako tämä säilyttää edellisessä kuvasarjassa mahdollisesti otettuja kalibrointikuvia.

LÄHTEET

Vuorenmaa, P. 2015. Viistovalokuvauslaitteen kuvausvalojen ohjainlaite. Tampereen ammattikorkeakoulu. Luettu 26.11.2015.
https://www.theseus.fi/bitstream/handle/10024/91297/Vuorenmaa_Panu.pdf

Woodham, R. 1980. Photometric method for determining surface orientation from multiple images. University of California. Luettu 26.11.2015.
https://classes.soe.ucsc.edu/cmps290b/Fall05/readings/Woodham80c.pdf

LIITTEET

Liite 1. MFCApplication4Dlg.cpp-tiedosto

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 1

```
1
2  // MFCApplication4Dlg.cpp : implementation file
3  //
4
5  #include "stdafx.h"
6  #include "MFCApplication4.h"
7  #include "MFCApplication4Dlg.h"
8  #include "afxdialogex.h"
9  #include "EDSDK\Header\EDSDK.h"
10 #include "EDSDK\Header\EDSDKErrors.h"
11 #include "EDSDK\Header\EDSDKTypes.h"
12 #include "sarjaportti.h"
13 #include "omaAction.h"
14 #include <direct.h>
15 #include <thread>
16 #include <filesystem>
17 #include <sstream>
18 #include <string>
19
20
21 #ifdef _DEBUG
22 #define new DEBUG_NEW
23 #endif
24
25 #define WM_USER_DOWNLOAD_COMPLETE       WM_APP+1
26 #define WM_USER_PROGRESS_REPORT         WM_APP+2
27 // CAboutDlg dialog used for App About
28
29 class CAboutDlg : public CDialogEx
30 {
31 public:
32     CAboutDlg();
33
34     // Dialog Data
35     enum { IDD = IDD_ABOUTBOX };
36
37 protected:
38     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
39
40     // Implementation
41 protected:
42     DECLARE_MESSAGE_MAP()
43 };
44
45 CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)
46 {
47 }
48
49 void CAboutDlg::DoDataExchange(CDataExchange* pDX)
50 {
51     CDialogEx::DoDataExchange(pDX);
52 }
53
54 BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
55 END_MESSAGE_MAP()
56
57
58 // CMFCApplication4Dlg dialog
59
60
61
62 CMFCApplication4Dlg::CMFCApplication4Dlg(CWnd* pParent /*=NULL*/)
63     : CDialogEx(CMFCApplication4Dlg::IDD, pParent)
64     , leda(FALSE)
65 {
66     piccount_ = 0;
67     aled = false;
68     bled = false;
69     cled = false;
70     valmis = true;
71     setPicCount(0);
72     setPicCountA(0);
73     setPicCountB(0);
74     setPicCountC(0);
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 2

```
75      setPictindex(0);
76      setComport("COM3");
77      //scanCom();
78      std::string debug = getComport();
79      setStage("naming");
80
81      m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
82  }
83
84  void CMFCApplication4Dlg::DoDataExchange(CDataExchange* pDX)
85  {
86      CDialogEx::DoDataExchange(pDX);
87      DDX_Control(pDX, IDC_BUTTON10, testi);
88      DDX_Control(pDX, IDC_BUTTON11, evfon);
89      DDX_Control(pDX, IDC_BUTTON12, evfoff);
90      DDX_Control(pDX, IDC_STATIC, _pictureBox);
91      //DDX_Control(pDX, IDC_BUTTON8, nappi8);
92      //DDX_Control(pDX, IDC_STATIC, nappi8);
93
94      DDX_Check(pDX, IDC_CHECK1, leda);
95  }
96
97  BEGIN_MESSAGE_MAP(CMFCApplication4Dlg, CDialogEx)
98      ON_WM_SYSCOMMAND()
99      ON_WM_PAINT()
100     ON_WM_QUERYDRAGICON()
101     ON_BN_CLICKED(IDC_BUTTON2, &CMFCApplication4Dlg::OnBnClickedButton2)
102     ON_BN_CLICKED(IDC_BUTTON3, &CMFCApplication4Dlg::OnBnClickedButton3)
103     ON_MESSAGE(WM_USER_DOWNLOAD_COMPLETE, OnDownloadComplete)
104     ON_MESSAGE(WM_USER_PROGRESS_REPORT, OnProgressReport)
105     ON_WM_CLOSE()
106     ON_BN_CLICKED(IDC_BUTTON5, &CMFCApplication4Dlg::OnBnClickedButton5)
107     ON_BN_CLICKED(IDC_BUTTON6, &CMFCApplication4Dlg::OnBnClickedButton6)
108     ON_BN_CLICKED(IDC_BUTTON7, &CMFCApplication4Dlg::OnBnClickedButton7)
109     ON_BN_CLICKED(IDC_BUTTON8, &CMFCApplication4Dlg::OnBnClickedButton8)
110     ON_BN_CLICKED(IDC_CHECK1, &CMFCApplication4Dlg::OnBnClickedCheck1)
111     ON_BN_CLICKED(IDC_CHECK2, &CMFCApplication4Dlg::OnBnClickedCheck2)
112     ON_BN_CLICKED(IDC_CHECK3, &CMFCApplication4Dlg::OnBnClickedCheck3)
113     ON_BN_CLICKED(IDC_BUTTON9, &CMFCApplication4Dlg::OnBnClickedButton9)
114     //ON_LBN_SELCHANGE(IDC_LIST1, &CMFCApplication4Dlg::OnLbnSelchangeList1)
115     //ON_LBN_SELCHANGE(IDC_LIST2, &CMFCApplication4Dlg::OnLbnSelchangeList2)
116     ON_BN_CLICKED(IDC_BUTTON4, &CMFCApplication4Dlg::OnBnClickedButton4)
117     ON_BN_CLICKED(IDC_BUTTON13, &CMFCApplication4Dlg::OnBnClickedButton13)
118     ON_BN_CLICKED(IDC_BUTTON14, &CMFCApplication4Dlg::OnBnClickedButton14)
119     ON_BN_CLICKED(IDC_BUTTON15, &CMFCApplication4Dlg::OnBnClickedButton15)
120     ON_BN_CLICKED(IDC_BUTTON1, &CMFCApplication4Dlg::OnBnClickedButton1)
121     ON_BN_CLICKED(IDC_BUTTON12, &CMFCApplication4Dlg::OnBnClickedButton12)
122     ON_BN_CLICKED(IDOK, &CMFCApplication4Dlg::OnBnClickedOk)
123 END_MESSAGE_MAP()
124
125
126 // CMFCApplication4Dlg message handlers
127
128 BOOL CMFCApplication4Dlg::OnInitDialog()
129 {
130     CDialogEx::OnInitDialog();
131
132     // Add "About..." menu item to system menu.
133
134     // IDM_ABOUTBOX must be in the system command range.
135     ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
136     ASSERT(IDM_ABOUTBOX < 0xF000);
137
138     CMenu* pSysMenu = GetSystemMenu(FALSE);
139     if (pSysMenu != NULL)
140     {
141         BOOL bNameValid;
142         CString strAboutMenu;
143         bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
144         ASSERT(bNameValid);
145         if (!strAboutMenu.IsEmpty())
146         {
147             pSysMenu->AppendMenu(MF_SEPARATOR);
148             pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 3

```
149         }
150     }
151
152     // Set the icon for this dialog.  The framework does this automatically
153     //  when the application's main window is not a dialog
154     SetIcon(m_hIcon, TRUE);         // Set big icon
155     SetIcon(m_hIcon, FALSE);        // Set small icon
156
157     // TODO: Add extra initialization here
158
159     setupListener(_controller);
160     setupObserver(getCameraModel());
161
162     //Execute controller
163     _controller->run();
164
165
166     return FALSE;  // return TRUE  unless you set the focus to a control
167 }
168
169 void CMFCApplication4Dlg::OnSysCommand(UINT nID, LPARAM lParam)
170 {
171     if ((nID & 0xFFF0) == IDM_ABOUTBOX)
172     {
173         CAboutDlg dlgAbout;
174         dlgAbout.DoModal();
175     }
176     else
177     {
178         CDialogEx::OnSysCommand(nID, lParam);
179     }
180 }
181
182 // If you add a minimize button to your dialog, you will need the code below
183 //  to draw the icon.  For MFC applications using the document/view model,
184 //  this is automatically done for you by the framework.
185
186 void CMFCApplication4Dlg::OnPaint()
187 {
188     if (IsIconic())
189     {
190         CPaintDC dc(this); // device context for painting
191
192         SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
193
194         // Center icon in client rectangle
195         int cxIcon = GetSystemMetrics(SM_CXICON);
196         int cyIcon = GetSystemMetrics(SM_CYICON);
197         CRect rect;
198         GetClientRect(&rect);
199         int x = (rect.Width() - cxIcon + 1) / 2;
200         int y = (rect.Height() - cyIcon + 1) / 2;
201
202         // Draw the icon
203         dc.DrawIcon(x, y, m_hIcon);
204     }
205     else
206     {
207         CDialogEx::OnPaint();
208     }
209 }
210
211 // The system calls this function to obtain the cursor to display while the user drags
212 //  the minimized window.
213 HCURSOR CMFCApplication4Dlg::OnQueryDragIcon()
214 {
215     return static_cast<HCURSOR>(m_hIcon);
216 }
217
218
219
220 void CMFCApplication4Dlg::OnBnClickedButton2()
221 {
222     // TODO: Add your control notification handler code here
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 4

```
223     setStage("selectlayer");
224     CButton *nappi3 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON3));
225     CCheckListBox *led_a = reinterpret_cast<CCheckListBox *>(GetDlgItem(IDC_CHECK1));
226     CCheckListBox *led_b = reinterpret_cast<CCheckListBox *>(GetDlgItem(IDC_CHECK2));
227     CCheckListBox *led_c = reinterpret_cast<CCheckListBox *>(GetDlgItem(IDC_CHECK3));
228     CEdit *nimi = reinterpret_cast<CEdit *>(GetDlgItem(IDC_EDIT1));
229
230     CString oma;
231     nimi->GetWindowTextW(oma);
232     setName(oma);
233     if(!getName().IsEmpty())
234     {
235         if(CreateDirectory(getName(),0))        // Try to create the folder.
236         {
237             nappi3->EnableWindow(true);         // Activation of step 2 controls
238             led_a->EnableWindow(true);
239             led_b->EnableWindow(true);
240             led_c->EnableWindow(true);
241             nimi->EnableWindow(false);          // Disable step 1 textbox
242
243         }
244         else
245         {
246             LPCWSTR viesti = L"Directory already exists";   // CreateDirectory function return false,
247             ::MessageBox(NULL,viesti,NULL,MB_OK);           // if creation fails, eg. Directory already
248         }                                                   // exists.
249     }
250
251 }
252
253
254 void CMFCApplication4Dlg::OnBnClickedButton3()
255 {
256     setStage("focus");
257
258     this->setActionCommand("startEVF");
259     this->fireEvent();
260
261
262     // TODO: Add your control notification handler code here
263     CButton *nappi1 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON1));
264     CButton *nappi4 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON4));
265     CButton *nappi5 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON5));
266     CButton *nappi6 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON6));
267     CButton *nappi7 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON7));
268     CActionButton *nappi8 = reinterpret_cast<CActionButton *>(GetDlgItem(IDC_BUTTON8));
269
270     CButton *led_a = reinterpret_cast<CButton *>(GetDlgItem(IDC_CHECK1));
271     CButton *led_b = reinterpret_cast<CButton *>(GetDlgItem(IDC_CHECK2));
272     CButton *led_c = reinterpret_cast<CButton *>(GetDlgItem(IDC_CHECK3));
273
274     const int piccountadd = 4; // 4 jos vain RAW, 8 jos RAW + jpeg
275
276     if(led_a->GetCheck() == BST_CHECKED)
277     {
278         setPicCount(getPicCount() + piccountadd);
279         setPicCountA(4);
280     }
281     if(led_b->GetCheck() == BST_CHECKED)
282     {
283         setPicCount(getPicCount() + piccountadd);
284         setPicCountB(4);
285     }
286     if(led_c->GetCheck() == BST_CHECKED)
287     {
288         setPicCount(getPicCount() + piccountadd);
289         setPicCountC(4);
290     }
291
292     int testtotal = getPicCount();
293     int testa = getPicCountA();
294     int testb = getPicCountB();
295     int testc = getPicCountC();
296
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 5

```
297     nappi1->EnableWindow(true);
298     nappi4->EnableWindow(true);
299     nappi5->EnableWindow(true);
300     nappi6->EnableWindow(true);
301     nappi7->EnableWindow(true);
302     nappi8->EnableWindow(true);
303
304     ledControllerWrite("$FSON!");
305
306 }
307 void CMFCApplication4Dlg::setupListener(ActionListener* listener)
308 {
309     addActionListener(listener);
310
311     testi.setActionCommand("TakePicture");
312     testi.addActionListener(listener);
313     evfon.setActionCommand("startEVF");
314     evfon.addActionListener(listener);
315     evfoff.setActionCommand("endEVF");
316     evfoff.addActionListener(listener);
317
318     _pictureBox.setActionCommand("downloadEVF");
319     _pictureBox.addActionListener(listener);
320 }
321
322 void CMFCApplication4Dlg::setupObserver(Observable* ob)
323 {
324
325     ob->addObserver(static_cast<Observer*>(&_pictureBox));
326
327 }
328 void CMFCApplication4Dlg::update(Observable* from, CameraEvent *e)
329 {
330     std::string event = e->getEvent();
331
332     //End of download of image
333     if(event == "DownloadComplete")
334     {
335         //The update processing can be executed from another thread.
336         ::PostMessage(this->m_hWnd, WM_USER_DOWNLOAD_COMPLETE, NULL, NULL);
337     }
338
339     //Progress of download of image
340     if(event == "ProgressReport")
341     {
342         EdsInt32 percent = *static_cast<EdsInt32 *>(e->getArg());
343
344         //The update processing can be executed from another thread.
345         ::PostMessage(this->m_hWnd, WM_USER_PROGRESS_REPORT, percent, NULL);
346     }
347
348     //shutdown event
349     if(event == "shutDown")
350     {
351         ::PostMessage(this->m_hWnd, WM_CLOSE, 0, NULL);
352     }
353 }
354 LRESULT CMFCApplication4Dlg::OnDownloadComplete(WPARAM wParam, LPARAM lParam)
355 {
356
357     //End of download of image
358     setPictindex(getPictindex() + 1);
359
360     std::string stage = getStage();
361     if(stage == "run")
362     {
363         int position = 100 - (double)((((double)getPicCountA() + (double)getPicCountB() + (double)
    getPicCountC()) / (double)getPicCount()) *100);
364         CProgressCtrl *progbar = reinterpret_cast<CProgressCtrl *>(GetDlgItem(IDC_PROGRESS1));
365         progbar->SetRange(0,100);
366         progbar->SetPos(position);
367         std::string temp;
368
369         CT2CA conversion(getName());
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 6

```
370         std::string temp2(conversion);
371         std::string temp3;
372
373         if(getPictindex() < 10)
374         {
375             temp = "IMG_000" + std::to_string(getPictindex()) + ".CR2";
376         }
377         else
378         {
379             temp = "IMG_00" + std::to_string(getPictindex()) + ".CR2";
380         }
381
382         temp3 = const_cast<char*>(temp2.c_str());
383         temp3 = temp3 + "\\" + const_cast<char*>(temp.c_str());
384         rename(const_cast<char*>(temp.c_str()),const_cast<char*>(temp3.c_str()));
385         if(getPicCountA() + getPicCountB() + getPicCountC() == 0)
386         {
387             std::ofstream *kirjuri = new std::ofstream();
388             kirjuri->open(getName()+(CString)"\\meta.xml",std::ofstream::app);
389             if(kirjuri->is_open())
390             {
391                 kirjuri->write("</image_data>",13);
392                 kirjuri->close();
393             }
394             if(kirjuri != NULL)
395             {
396                 delete kirjuri;
397                 kirjuri = NULL;
398             }
399         }
400         if(getPicCountA() > 0)
401         {
402             kuvaaA();
403         }
404         else if(getPicCountB() > 0)
405         {
406             kuvaaB();
407         }
408         else if(getPicCountC() > 0)
409         {
410             kuvaaC();
411         }
412     }
413     else if(stage == "focus")
414     {
415         std::string temp;
416         if(getPictindex() < 10)
417         {
418             temp = "IMG_000" + std::to_string(getPictindex()) + ".CR2";
419         }
420         else
421         {
422             temp = "IMG_00" + std::to_string(getPictindex()) + ".CR2";
423         }
424
425         CT2CA conversion(getName());
426         std::string temp2(conversion);
427         std::string temp3;
428
429         temp3 = const_cast<char*>(temp2.c_str());
430         temp3 = temp3 + "\\focuscalib.CR2";
431
432         rename(const_cast<char*>(temp.c_str()),const_cast<char*>(temp3.c_str()));
433
434
435         setStage("measurement");
436     }
437     else if(stage == "measurement")
438     {
439         std::string temp;
440         if(getPictindex() < 10)
441         {
442             temp = "IMG_000" + std::to_string(getPictindex()) + ".CR2";
443         }
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 7

```
444         else
445         {
446             temp = "IMG_00" + std::to_string(getPictindex()) + ".CR2";
447         }
448
449         CT2CA conversion(getName());
450         std::string temp2(conversion);
451         std::string temp3;
452
453         temp3 = const_cast<char*>(temp2.c_str());
454         temp3 = temp3 + "\\measurecalib.CR2";
455
456         rename(const_cast<char*>(temp.c_str()),const_cast<char*>(temp3.c_str()));
457
458
459         setStage("lightning");
460     }
461     else if(stage == "lightning")
462     {
463         std::string temp;
464         if(getPictindex() < 10)
465         {
466             temp = "IMG_000" + std::to_string(getPictindex()) + ".CR2";
467         }
468         else
469         {
470             temp = "IMG_00" + std::to_string(getPictindex()) + ".CR2";
471         }
472
473         CT2CA conversion(getName());
474         std::string temp2(conversion);
475         std::string temp3;
476
477         temp3 = const_cast<char*>(temp2.c_str());
478         temp3 = temp3 + "\\lightcalib.CR2";
479
480         rename(const_cast<char*>(temp.c_str()),const_cast<char*>(temp3.c_str()));
481
482
483         setStage("run");
484     }
485
486     valmis=true;
487
488     return 0;
489
490
491 }
492
493 LRESULT CMFCApplication4Dlg::OnProgressReport(WPARAM wParam, LPARAM lParam)
494 {
495     return 0;
496 }
497
498 void CMFCApplication4Dlg::OnBnClickedButton5()
499 {
500     // TODO: Add your control notification handler code here
501     // Focus --
502     this->setActionCommand("focus_Near3");
503     this->fireEvent();
504 }
505
506
507 void CMFCApplication4Dlg::OnBnClickedButton6()
508 {
509     // TODO: Add your control notification handler code here
510     // Focus -
511     this->setActionCommand("focus_Near1");
512     this->fireEvent();
513 }
514
515
516 void CMFCApplication4Dlg::OnBnClickedButton7()
517 {
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 8

```
518     // TODO: Add your control notification handler code here
519     // Focus +
520     this->setActionCommand("focus_Far1");
521     this->fireEvent();
522 }
523
524
525 void CMFCApplication4Dlg::OnBnClickedButton8()
526 {
527     // TODO: Add your control notification handler code here
528     // Focus ++
529     this->setActionCommand("focus_Far3");
530     this->fireEvent();
531 }
532
533 void CMFCApplication4Dlg::setNoCamDet(bool uusi)
534 {
535     nocamdet_ = uusi;
536 }
537 bool CMFCApplication4Dlg::getNoCamDet() const
538 {
539     return nocamdet_;
540 }
541 void CMFCApplication4Dlg::setPicCount(int uusi)
542 {
543     piccount_ = uusi;
544 }
545 int CMFCApplication4Dlg::getPicCount() const
546 {
547     return piccount_;
548 }
549 void CMFCApplication4Dlg::setName(CString name)
550 {
551     name_ = name;
552 }
553 CString CMFCApplication4Dlg::getName() const
554 {
555     return name_;
556 }
557 void CMFCApplication4Dlg::setPicCountA(int uusi)
558 {
559     piccounta_ = uusi;
560 }
561 int CMFCApplication4Dlg::getPicCountA() const
562 {
563     return piccounta_;
564 }
565 void CMFCApplication4Dlg::setPicCountB(int uusi)
566 {
567     piccountb_ = uusi;
568 }
569 int CMFCApplication4Dlg::getPicCountB() const
570 {
571     return piccountb_;
572 }
573 void CMFCApplication4Dlg::setPicCountC(int uusi)
574 {
575     piccountc_ = uusi;
576 }
577 int CMFCApplication4Dlg::getPicCountC() const
578 {
579     return piccountc_;
580 }
581 void CMFCApplication4Dlg::setComport(std::string uusi)
582 {
583     comport_ = uusi;
584 }
585 std::string CMFCApplication4Dlg::getComport() const
586 {
587     return comport_;
588 }
589 void CMFCApplication4Dlg::setStage(std::string uusi)
590 {
591     stage_ = uusi;
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 9

```
592 }
593 std::string CMFCApplication4Dlg::getStage() const
594 {
595     return stage_;
596 }
597 std::string CMFCApplication4Dlg::findPic()
598 {
599     picpath_ = "notfound";
600     int index = 1;
601     return picpath_;
602 }
603 bool CMFCApplication4Dlg::scanCom()
604 {
605     int index = 1;
606     SerialPort *omaportti = new SerialPort();
607     std::wstringstream wss;
608     std::string tempstr = "COM1";
609
610     while(index < 30)
611     {
612         unsigned char reply[5];
613
614         wss.str(L"");
615         wss.ignore();
616         wss.flush();
617         wss.clear();
618
619         wss << tempstr.c_str();
620         wss.str().c_str();
621
622         omaportti->connect((LPWSTR)wss.str().c_str());
623         omaportti->sendArray((unsigned char*)"$SSAA!",6);
624         omaportti->getArray(reply,5);
625         omaportti->clear();
626         omaportti->disconnect();
627         if(reply[0] == 'A' && reply[1] == 'C' && reply[2] == 'K')
628         {
629             setComport(tempstr);
630             return true;
631         }
632
633         index++;
634         if(index == 10)
635         {
636             tempstr.at(3) = 49;
637             tempstr.push_back(index + 38);
638         }
639         else if (index > 10)
640         {
641             tempstr.at(4) = index + 38;
642         }
643         else
644             tempstr.at(3) = index + 48;
645
646     }
647
648     delete omaportti;
649     omaportti = NULL;
650
651
652
653     return false;
654 }
655
656 void CMFCApplication4Dlg::ledControllerWrite(std::string command)
657 {
658     std::wstringstream wss;
659     unsigned char commandchar[6];
660
661     for(int i = 0; i < 6; i++)                  // A loop to copy the command string to a commandchar
    array
662     {
663         commandchar[i] = command.at(i);
664     }
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 10

```
665
666     wss << getComport().c_str();                // Reading the desired comport to wss with getComport
667     wss.str().c_str();                          // getter. Comport must be set earlier with setComport.
668
669     SerialPort *omaportti = new SerialPort();   // Creation of SerialPort object
670     omaportti->connect((LPWSTR)wss.str().c_str());  // Open the desired comport with connect function
671     omaportti->sendArray(commandchar,6);        // Sending commandchar array
672     omaportti->disconnect();                    // Closing the comport
673     if(omaportti != NULL)
674     {
675         delete omaportti;                       // Deleting the SerialPort object
676         omaportti = NULL;                       // and setting the pointer to NULL
677     }
678 }
679
680 void CMFCApplication4Dlg::kuvaaA()
681 {
682     std::ofstream *kirjuri = new std::ofstream();
683
684     kirjuri->open(getName()+(CString)"\\meta.xml",std::ofstream::app);
685     if(kirjuri->is_open())
686     {
687         kirjuri->write("\t<image>\n",9);
688         kirjuri->write("\t\t<X value=\"aaa\"/>\n",19);
689         kirjuri->write("\t\t<Y value=\"aaa\"/>\n",19);
690         kirjuri->write("\t\t<Z value=\"aaa\"/>\n",19);
691         kirjuri->write("\t</image>\n",10);
692     }
693
694     if(kirjuri != NULL)
695     {
696         delete kirjuri;
697         kirjuri = NULL;
698     }
699
700     setPicCountA(getPicCountA() - 1);
701     this->setActionCommand("TakePicture");
702     this->fireEvent();
703
704
705
706 }
707 void CMFCApplication4Dlg::kuvaaB()
708 {
709     std::ofstream *kirjuri = new std::ofstream();
710
711     kirjuri->open(getName()+(CString)"\\meta.xml",std::ofstream::app);
712     if(kirjuri->is_open())
713     {
714         kirjuri->write("\t<image>\n",9);
715         kirjuri->write("\t\t<X value=\"bbb\"/>\n",19);
716         kirjuri->write("\t\t<Y value=\"bbb\"/>\n",19);
717         kirjuri->write("\t\t<Z value=\"bbb\"/>\n",19);
718         kirjuri->write("\t</image>\n",10);
719     }
720
721     if(kirjuri != NULL)
722     {
723         delete kirjuri;
724         kirjuri = NULL;
725     }
726
727     setPicCountB(getPicCountB() - 1);
728     this->setActionCommand("TakePicture");
729     this->fireEvent();
730 }
731 void CMFCApplication4Dlg::kuvaaC()
732 {
733     std::ofstream *kirjuri = new std::ofstream();
734
735     kirjuri->open(getName()+(CString)"\\meta.xml",std::ofstream::app);
736     if(kirjuri->is_open())
737     {
738         kirjuri->write("\t<image>\n",9);
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 11

```
739         kirjuri->write("\t\t<X value=\"ccc\"/>\n",19);
740         kirjuri->write("\t\t<Y value=\"ccc\"/>\n",19);
741         kirjuri->write("\t\t<Z value=\"ccc\"/>\n",19);
742         kirjuri->write("\t</image>\n",10);
743     }
744
745     if(kirjuri != NULL)
746     {
747         delete kirjuri;
748         kirjuri = NULL;
749     }
750
751     setPicCountC(getPicCountC() - 1);
752     this->setActionCommand("TakePicture");
753     this->fireEvent();
754 }
755
756 void CMFCApplication4Dlg::setPictindex(int uusi)
757 {
758     pictindex_ = uusi;
759 }
760
761 int CMFCApplication4Dlg::getPictindex() const
762 {
763     return pictindex_;
764 }
765
766 void CMFCApplication4Dlg::OnBnClickedCheck1()
767 {
768     // TODO: Add your control notification handler code here
769     if (aled == false)
770     {
771         aled = true;
772     }
773     else
774     {
775         aled = false;
776     }
777 }
778
779
780 void CMFCApplication4Dlg::OnBnClickedCheck2()
781 {
782     // TODO: Add your control notification handler code here
783     if (bled == false)
784     {
785         bled = true;
786     }
787     else
788     {
789         bled = false;
790     }
791 }
792
793
794 void CMFCApplication4Dlg::OnBnClickedCheck3()
795 {
796     // TODO: Add your control notification handler code here
797     if (cled == false)
798     {
799         cled = true;
800     }
801     else
802     {
803         cled = false;
804     }
805 }
806
807
808 void CMFCApplication4Dlg::OnBnClickedButton9()
809 {
810     // TODO: Add your control notification handler code here
811
812     if(getStage() == "run")
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 12

```
813     {
814         std::ofstream *kirjuri = new std::ofstream();   // Create a pointer to new ofstream object
815         kirjuri->open(getName()+(CString)"\\meta.xml"); // Create the meta file in requested location
816 
817         if(kirjuri->is_open())                          // Testing if the file creation was succesfull
818         {
819             kirjuri->write("<?xml version=\"1.0\" encoding=\"utf-8\"?>\n",39); // Writing the first line
820             kirjuri->write("<image_data>\n",13);        // Writing the second line
821             kirjuri->close();                           // Closing the file for now
822         }
823 
824         if(kirjuri != NULL)
825         {
826             delete kirjuri;                             // Delete the ofstream object
827             kirjuri = NULL;                             // and set the pointer to NULL.
828         }
829 
830         unsigned char puskuri[6];
831         unsigned char * char_ptr;
832         std::string buffer = "$SSAC!";
833         char_ptr = &puskuri[0];
834 
835         puskuri[0] = '$';
836         puskuri[1] = 'S';
837         puskuri[2] = 'S';
838         puskuri[3] = 'A';
839         puskuri[4] = 'C';
840         puskuri[5] = '!';
841 
842         if(getPicCountA() == 4)
843         {
844             puskuri[3] = 'A';
845             buffer.at(3) = 'A';
846         }
847         else if(getPicCountB() == 4)
848         {
849             puskuri[3] = 'B';
850             buffer.at(3) = 'B';
851         }
852         else
853         {
854             puskuri[3] = 'C';
855             buffer.at(3) = 'C';
856         }
857         if(getPicCountC() == 4)
858         {
859             puskuri[4] = 'C';
860             buffer.at(4) = 'C';
861         }
862         else if(getPicCountB() == 4)
863         {
864             puskuri[4] = 'B';
865             buffer.at(4) = 'B';
866         }
867         else
868         {
869             puskuri[4] = 'A';
870             buffer.at(4) = 'A';
871         }
872 
873         ledControllerWrite(buffer);
874 
875         int kuvienlkm = getPicCount();
876 
877         if(getPicCountA() == 4)
878         {
879             kuvaaA();
880         }
881         else if(getPicCountB() == 4)
882         {
883             kuvaaB();
884         }
885         else if(getPicCountC() == 4)
886         {
```

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 13

```
887             kuvaaC();
888         }
889     }
890 }
891
892 void CMFCApplication4Dlg::OnLbnSelchangeList1()
893 {
894     // TODO: Add your control notification handler code here
895 }
896
897 void CMFCApplication4Dlg::OnLbnSelchangeList2()
898 {
899     // TODO: Add your control notification handler code here
900 }
901
902 void CMFCApplication4Dlg::OnBnClickedButton4()
903 {
904     // TODO: Add your control notification handler code here
905     this->setActionCommand("TakePicture");
906     this->fireEvent();
907
908
909     ledControllerWrite("$FOFF!");
910
911     CButton *nappi13 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON13));
912     nappi13->EnableWindow(true);
913
914     ::MessageBox(m_hWnd,L"Before clicking next, please insert millimeter calibration image to the ---",L
    "Attention!",NULL);
915 }
916
917
918 void CMFCApplication4Dlg::OnBnClickedButton13()
919 {
920     // TODO: Add your control notification handler code here
921
922     CButton *nappi14 = reinterpret_cast<CButton *>(GetDlgItem(IDC_BUTTON14));   // Create a pointer to
    the
923     nappi14->EnableWindow(true);                                                // button with ID
    "IDC_BUTTON14
924
925     this->setActionCommand("TakePicture");      // Before calling fireEvent function, proper
    ActionCommand
926     this->fireEvent();                          //  must be set. "TakePicture" makes the camera take
927                                                 //  a picture. FireEvent excecutes the set command.
928
929     ::MessageBox(m_hWnd,L"Before clicking next, please insert lightning calibration image to the ---",L
    "Attention!",NULL);
930 }
931
932 void CMFCApplication4Dlg::OnBnClickedButton14()
933 {
934     // TODO: Add your control notification handler code here
935     this->setActionCommand("endEVF");
936     this->fireEvent();
937
938     this->setActionCommand("TakePicture");
939     this->fireEvent();
940
941     ::MessageBox(m_hWnd,L"Before clicking next, please insert the sample to the ---",L"Attention!",NULL);
942 }
943
944 void CMFCApplication4Dlg::OnBnClickedButton15()         //New-painike
945 {
946     // TODO: Add your control notification handler code here
947
948 }
949
950 void CMFCApplication4Dlg::OnBnClickedButton1()          // Auto Focus -painike
951 {
952     // TODO: Add your control notification handler code here
953     this->setActionCommand("evfAFon");
954     this->fireEvent();
955 }
```

14 (14)

C:\Users\Aleksi\Desktop\Syksy\Opinnäytetyö\MFCApplication4\MFCApplication4\MFCApplication4Dlg.cpp 14

```
956
957 void CMFCApplication4Dlg::OnBnClickedButton12()         //EVF off -painike
958 {
959
960     // TODO: Add your control notification handler code here
961     this->setActionCommand("endEVF");
962     this->fireEvent();
963 }
964
965 void CMFCApplication4Dlg::OnBnClickedOk()
966 {
967
968 }
969
```