jamk

3D-skannauksien tallennus OpenCV:lle

Riku Loponen

Opinnäytetyö, AMK
Maaliskuu 2023
Sähkö- ja automaatiotekniikan tutkinto-ohjelma

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Loponen Riku

3D-skannauksien tallennus OpenCV:lle

Jyväskylä: Jyväskylän ammattikorkeakoulu. Maaliskuu 2023, 22 sivua.

Sähkö- ja automaatiotekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

3D-skannerit ovat yleistyneet teollisuudessa. Toimeksiantaja oli hankkinut 3D-skannereita ja oli kiinnostunut koneoppimisen käyttämisestä 3D-skannauksen yhteydessä. OpenCV konenäkökirjastolla, pystyy kehittämään koneoppimis-algoritmeja konenäköä varten. 3D-skannauksen tuottama pistepilvi ja konenäkökameran ottama kuva eivät datarakenteeltaan eroa paljoa toisistaan, joten koneoppimista päätettiin testata OpenCV kirjastolla.

Koneoppimisen opettaminen ja testaaminen vaatii paljon mittausdataa. Sopivan materiaalin keräämiseen ei ollut valmiuksia, joten tavoitteeksi tuli tallennusohjelman toteutus, jolla pystyy tallentamaan 3D-skannauksia OpenCV:llä käsiteltäväksi. Ennen kuin ohjelma voitiin tehdä, piti tutkia 3D-skannauksien tallennus mahdollisuuksia, käytettyjen laitteiden tukemia tallennusmuotoja sekä niiden ohjelmointirajapintoja.

3D-skannereiden ohjelmistorajapintojen sekä OpenCV kirjaston tallennus mahdollisuudet selvitettiin. Valittiin yksinkertaisin tapa tallentaa mittausdata ja tehtiin ohjelma, jolla saadaan yhteys 3D-skanneriin ja mittaus tallennettua. Tallennusmuodoksi valikoitui OpenCV:n FileStorage funktion tuottama XML-tiedosto. Tiedosto on helppo käyttää OpenCV:llä ja kykenee tallentamaan kaiken 3D-skannerin tuottaman datan. Ohjelmointi tehtiin Visual Studiolla ja käytettiin C++ kieltä. Tallennusohjelmissa noudatettiin laitteistojen ohjelmointiesimerkkejä laitteisiin yhdistämisessä ja datan vastaanottamisessa. Vastaanotetusta datasta tallennettiin pistepilvi OpenCV:n XML-tiedostoon.

Toteutetuilla ohjelmilla pystyy tallentamaan dataa koneoppimisen tutkimista varten. Käytettyä ohjelman toteutustapaa voi käyttää pohjana muillakin 3D-skannereilla, jos niissä on vastaava ohjelmointirajapinta. Ohjelmia kannattaa jatkokehittää ja lisätä niihin pistepilven lisäksi myös muita 3D-skannauksen tietoja, kuten korkeuskartan tai pinnan normaalivektorit. Lisäksi ohjelmiin voisi lisätä valittavaksi muita yleisesti käytettyjä tallennusmuotoja.

Avainsanat (asiasanat)

3D-skannerit, 3D-mallinnus, pistepilvi, mittaustekniikka

Muut tiedot (salassa pidettävät liitteet)

-

jamk

Description

Loponen Riku

Collecting 3D-scans for OpenCV

Jyväskylä: JAMK University of Applied Sciences, March 2023, 22 pages.

Degree Programme in Electrical and Automation Engineering. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

3D scanners have become more common in industry. JAMK university had acquired 3D scanners and was interested in using machine learning with them. OpenCV machine vision library can be used to train machine learning algorithms for machine vision. 3D scanned point cloud and picture produced by a camera have similar data structure. It was decided to test machine learning with OpenCV library.

Machine learning requires a lot of data for training and testing it. There were no prerequisite to gather the data. So the target was set to produce a software to gather 3D scans and save them in format that can be accessed with OpenCV. Before the software could be made, 3D scanning file formats and 3D scanners programming interfaces needed to be investigated.

3D scanner's and OpenCV's data saving properties were investigated. It was found that the simplest way to save the data was with OpenCV's FileStorage function. It produces a XML file that can save all the data produced by the scanners. Software was programmed with Visual Studio using C++. Code examples provided by the manufacturer were followed when connecting to the scanner and collecting data. Point cloud was saved from the collected data as a XML file.

The produced programs were able to save data for OpenCV. The way the programs were made can be applied to other makes of 3D scanners if they have a similar programming interface. Programs could be further developed to include more data produced by the scanners such as height map and normal vectors. Also other common file formats could be added.

Keywords/tags (subjects)

Three-dimensional scanners, three-dimensional imaging, point cloud, measuring technology

Miscellaneous (Confidential information)

-

Sisältö

Kuviot

Taulukot

1 Johdanto

Opinnäytetyön aiheena on 3D-skannereilla tuotetun mittausdatan tallennus OpenCV konenäkökirjastolla käsiteltävään muotoon. 3D-skannereita käytetään automaatiossa kappaleiden laaduntarkastukseen, kappaleiden paikantamiseen ja robottien navigaatioon (What is 3D scanner used for? n.d.). Konenäkökameroita käytetään samoihin tarkoituksiin. Konenäön yhteydessä on pitkään käytetty koneoppimista. Koneoppimisessa hyödynnetään tekoälyä datan analysointiin. Koneoppimisen avulla esimerkiksi itseajavat autot, voivat havainnoida ympäristöään ja havaita liikennemerkkejä, muita tienkäyttäjiä ja navigoida tiellä. (What is computer vision? n.d.) Konenäkökamerat ja 3D-skannerit tuottavat saman tyylistä dataa, kamera kertoo näkemänsä värit ja skanneri etäisyyden näkemiinsä kohteisiin (Mizerák, Rosocha & Trebuňa 2018, 1). Tämän takia on herännyt kysymys: Miten koneoppimista voisi hyödyntää 3D-skannerien kanssa?

Toimeksiantajana oli Jyväskylän ammattikorkeakoulu. Toimeksiantaja on hankkinut 3D-skannauslaitteistoja. Laitteita on tarkoitus hyödyntää tutkimuksessa sekä koulutuksessa. Tällä hetkellä laitteita käytetään niiden mukana tulleiden ohjelmistojen sekä yhteen sopivien kaupallisten ohjelmien kanssa. Toimeksiantajaa on kiinnostanut koneoppimisen hyödyntäminen kappaleiden tunnistuksessa. OpenCV sisältää työkaluja koneoppimiseen sekä mahdollistaa omien työkalujen kehittämisen. Jotta koneoppimista voidaan tutkia, tarvitaan paljon mittausdataa sen opettamiseen ja testaamiseen. Siksi on tärkeää, että 3D-skannereilla saadaan helposti ja nopeasti tuotettua mittauksia sopivassa tiedostomuodossa.

Tavoitteena oli luoda ohjelma, jolla toimeksiantajan 3D-skannerien mittausdata saadaan tallennettua OpenCV:llä käsiteltävässä muodossa. Ohjelman luomisen edellytyksenä on valita tallennusmuoto ja ohjelman toteutustapa. Tämä edellyttää 3D-skannausten tallennusmuotojen tutkimista ja arviointia sekä laitteiston tukemien tallennusmuotojen ja kolmannen osapuolen ohjelmistojen liitäntämahdollisuuksien selvittämistä. Näiden tietojen perusteella valittiin yksinkertaisin tapa suorittaa tallennusohjelma. Ohjelman lisäksi sen lähdekoodi luovutettiin jatkokehitystä varten toimeksiantajalle ja se pyrittiin pitämään selkeänä ja helppokäyttöisenä. Itse koneoppimiseen ja OpenCV:n käyttämiseen ei työssä syvennytty. Testaamiseen tarvittavan materiaalin keräys sekä tutkiminen jää toimeksiantajalle tulevaisuuden haasteeksi. Lisäksi mittausympäristön suunnittelu ja valmistus jää toimeksiantajan vastuulle.

Opinnäytetyö toteutettiin kehittämistutkimuksena ja siinä käytettiin laadullisen tutkimuksen aineistonkeruumenetelmiä. Kanasen (2015) mukaan kehittämistutkimus tähtää muutoksen aikaan saamiseksi. Siinä kehitetään tuotetta, menetelmää tai muuta asiaa. Kehittämistutkimus voi yhdistellä määrällistä ja laadullista tutkimusta tai olla pelkästään laadullinen tutkimus. Kehittämistutkimukseen kuuluu kehittämissykli. Toistuvissa sykleissä etsitään keinoja ongelman poistamiseen ja seurataan keinojen vaikutusta. (Kananen 2015, 39-41.) Laadullisia aineistonkeruumenetelmiä ovat havainnointi, haastattelut, kyselyt ja dokumentit (mts. 12). Työssä käytettiin havainnointia ja dokumentteja aineiston keruussa. Pääasiallisina lähteinä käytettiin laitteiden ja ohjelmistojen käyttöohjeita sekä dokumentaatiota. Aineiston pohjalta kehiteltiin ratkaisuja, joita arvioitiin ensin teoreettisesti, ja sen jälkeen toimivuutta kokeiltiin ja tarkasteltiin käytännössä.

2 Opinnäytetyön keskeiset käsitteet ja teknologiat

Seuraavaksi on esitelty 3D-skannerien toimintaa sekä työssä käytettyjä laitteistoja ja ohjelmointikirjastoja. 3D-skannaukseen käytettiin Photoneo PhoXi 3D ja Gocator 2140 3D-skannereita, jotka oli hankittu ammattikorkeakoulun laboratorioon. Ohjelmointikirjastoissa ja skannereissa on paljon ominaisuuksia. Niistä esitellään vain työn kannalta tärkeimmät.

2.1 3D-skannaus automaatiossa

3D-skannaus on mittaus, joka muodostaa kohteesta kolmiulotteisen mallin, joka kuvaa kohteen ulkoisia muotoja. 3D-skannerit voivat kuvata vain osia, jotka ovat näkyvissä. 3D-skannauksen tulosta voi luonnehtia ”kuvaksi”, jossa esitetään kohteen etäisyys mittalaitteesta. Yleensä yksi skannaus ei riitä kohteen täydellisen mallin tuottamiseen, vaan kohdetta täytyy kuvata useasta eri kulmasta. (Mizerák ym. 2018, 1.) Mittaustulos esitetään yleisesti pistepilvenä. Pistepilvi on kokoelma avaruudessa olevia pisteitä, jotka muodostavat skannatun kohteen pinnan. (Mts. 4.) Pistepilven lisäksi 3D-skannauksen data voi sisältää: pinnan normaalivektorin, värin, tekstuurin, korkeuskartan ja mittauksen varmuusarvion. Lisätiedot esitetään jokaista pistepilven pistettä kohti. (Phoxi Control 1.7 2022, 24.) Lisätietojen tuki vaihtelee mittalaitteittain. 3D-skannaukseen on useita menetelmiä, ne voidaan jakaa mittaustekniikan perusteella eri ryhmiin (Mizerák ym. 2018, 1).

Koskettavat menetelmät

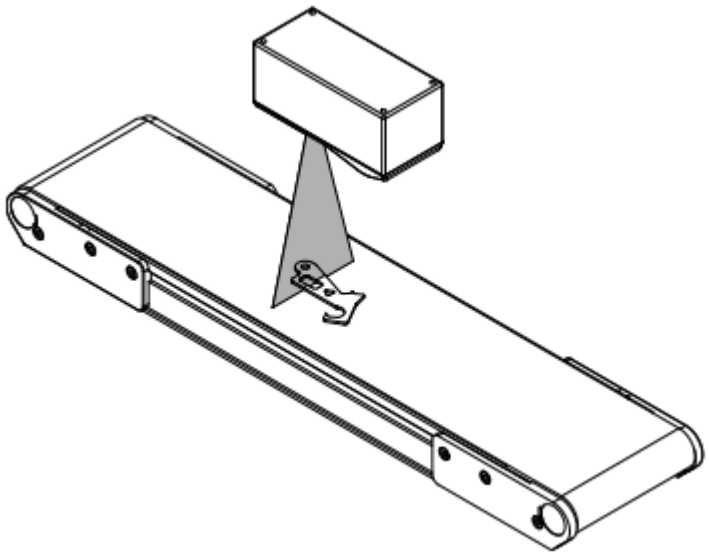
Koskettavissa menetelmissä kappale kiinnitetään tukevaan alustaan ja sen pintaa kosketetaan mittauspäällä. Mittauspää on kiinni akselistossa joko lineaariakselistossa, nivelletyssä varressa tai näiden yhdistelmässä. Akseliston ja nivelten paikkaa seurataan tarkasti kulma-antureilla ja siitä lasketaan mittapään kosketuspaikka. Mittapää voi olla käsin liikuteltava tai automaattisesti mittaava. Koskettavien menetelmien heikkous on koskettamisen tarve sekä hitaus. Koskettaminen voi aiheuttaa vahinkoa herkille kappaleille ja nopeimmatkin automaattiset laitteet pystyvät mittaamaan vain muutamia satoja pisteitä sekunnissa. (Mizerák ym. 2018, 1-2.)

Time-of-flight/laserpulssi

Laserpulssimenetelmässä kohteeseen heijastetaan lasersäde ja sen paluuaika mitataan. Takaisin heijastukseen kuluneesta ajasta voidaan laskea etäisyys. Laitteen tarkkuus riippuu ajanmittauksen tarkkuudesta. Valolla menee noin 3,3 pikosekuntia kulkea 1 millimetri. Laitteet käyttävät yleensä peiliskanneria lasersäteen tähtäämiseen. Peiliskannerit ovat todella nopeita ja tyypillinen laserpulssiskanneri mittaa noin 10 000 – 100 000 pistettä sekunnissa. (Mizerák ym. 2018, 2.) Laserpulssiskannerien vahvuus on pitkä kantama. Laitteiden kantama mitataan kilometreissä. Laserpulssin heikkous on huono tarkkuus, joka johtuu valon suuresta nopeudesta ja heijastusajan mittaamisen vaikeudesta. (Mts. 3.)

Laserkolmiomenetelmä

Laserkolmiomenetelmässä mittalaite lähettää lasersäteen kappaleeseen ja sen osumakohta kuvataan kameralla. Kamera on laserin lähetyskohdasta sivussa ja takaisin heijastuneen laserin paikka kamerassa muuttuu etäisyyden mukaan. Osumakohta lasketaan kolmiolaskennalla, kun laserin ja kameran paikka ja kulma ovat tiedossa. Mittauksessa joko mittalaitetta tai mitattavaa kappaletta liikutetaan niin että laser kulkee mitattavien alueiden yli. Lasersäde on usein viivamainen, jolloin mittaus nopeutuu verrattuna pistemäiseen laseriin (ks. kuvio 1). (Mizerák ym. 2018, 2-3.)



Kuvio 1. Yksinkertainen mittausympäristö (Gocator Line Profile Sensors 2022, 38)

Laserkolmiomenetelmän heikkous on rajallinen mittausetäisyys. Etäisyys on parhaillaan joitakin metrejä. Vahvuus on tarkkuus, joka voi olla jopa kymmeniä mikrometrejä. (Mizerák ym. 2018, 3.)

Valostruktuuri

Mittalaite lähettää valokuvion, jota kuvataan kameralla. Kamera on sivussa valokuvion lähettäjästä ja laite laskee kuvion vääristymästä kohteen etäisyyden, kuten laserkolmiomenetelmässä. Valostruktuuri menetelmällä koko mittausalue mallinnetaan kerralla. Valostruktuuri pystyy kuvaamaan laajan alueen kerralla, nopeasti ja tarkasti. Jotkin laitteet kykenevät skannaamaan kuvausalueen sekunnin murto-osassa. Tätä on hyödynnetty liikkuvien tai nopeasti muuttuvien kohteiden mallintamiseen. (Mizerák ym. 2018, 4.)

2.2 3D-skannauksien tallennusmuotoja

Pistepilvien tallentamiseen on hyvin paljon eri tiedostomuotoja. Useilla ohjelmistojen ja mittalaiteiden valmistajilla on omat tiedostomuotonsa ja lisäksi on valmistajasta riippumattomia tiedostomuotoja. Tiedostot voivat olla ASCII eli tekstimuotoisia tai binäärimuodossa tallennettuja, myös pistepilven lisäksi tallennettu sisältö vaihtelee. (Thomson 2018.)

XYZ

XYZ-tiedosto sisältää X,Y,Z koordinaatistoon sijoitettuja pisteitä. Tiedosto on ASCII-muodossa eikä sille ole standardia. Standardin puuttuminen heikentää tiedoston käytettävyyttä, mutta se on silti tuettu useissa ohjelmistoissa. (Thomson 2018.)

OBJ

OBJ on Wavefront Technologiesin kehittämä ASCII-tiedosto. Tiedosto suunniteltiin 3D-mallien tallennukseen ja siirtämiseen ohjelmien välillä. OBJ määrittelee kärkipisteet ja niistä koostuvia muotoja kuten: piste, tahko, jana, kaari. Lisäksi OBJ voi sisältää koordinaatiston värikartalle. OBJ ei standardin mukaan sisällä itse värejä, vaan ne tallennetaan Wavefront Material Template Library (MTL) tiedostoon. MTL-tuki vaihtelee ohjelmien välillä ja sen käytön vaihtoehdoksi on sovittu värin määrittäminen jokaiselle kärkipisteelle OBJ-tiedostossa. (Wavefront OBJ File Format 2020.)

PLY

Polygon file format (PLY) kehiteltiin Stanfordin yliopistolla 3D-skannausten tallennukseen. Sen tarkoitus oli olla yksinkertainen ja helppokäyttöinen, mutta silti pystyä monipuoliseen mallintamiseen. PLY voi olla joko ASCII- tai binäärimuodossa. Molempien muotojen sisältö on sama. Tiedoston perusrakenne on kärkipisteiden ja niistä muodostuvien tahkojen määrittely. Tahkoja ei ole pakko määritellä, jolloin tiedosto sisältää pelkästään pisteitä eli pistepilven. Tiedoston määritelmässä on annettu mahdollisuus projektikohtaisten ominaisuuksien määrittelyyn. Kärkipisteille voidaan esimerkiksi määritellä väri ja pinnan normaali. (Polygon File Format (PLY) Family 2019.)

PCD

Point Cloud Data (PCD) on Point Cloud Libraryn käyttämä tiedostomuoto. Point Cloud Library on avoimen koodin pistepilvien käsittelyyn tarkoitettu ohjelmointikirjasto. PCD-tiedostomuoto kehitettiin, koska tarvittiin n-ulottuvuuden histogrammia tukeva tiedostomuoto. PCD kykenee tallentamaan pistepilven ja histogrammin lisäksi muita tietoja, kuten normaalivektorit ja värin. Tiedosto voi olla ASCII- tai binäärimuodossa. (The PCD (Point Cloud Data) file format n.d.)

E57

E57 kehiteltiin ASTM E57 komiteassa ja se on määritelty ASTM E2807 -standardissa. E57 on pistepilvien, kuvien ja muun 3D-skannerien tuottaman datan tallentamiseen suunniteltu tiedostomuoto. Sitä ei ole kehitelty laitteisto- tai ohjelmistovalmistajan tarpeisiin, vaan sen tarkoituksena on toimia yleisenä 3D-skannauksien tallennusmuotona. E57 on pääosin binääritiedosto. Mittausdata on binäärinä ja osa metadatasta on ASCII-muodossa. (libE57: Software Tools for Managing E57 Files n.d.)

2.3 OpenCV konenäkökirjasto

OpenCV on avoimeen lähdekoodiin perustuva konenäkö- ja koneoppimiskirjasto (About n.d.). OpenCV sisältää perinteisen kuviin perustuvan konenäön lisäksi myös 3D-mallintamiseen ja pistepilvien käsittelyyn tarkoitettuja työkaluja. Datan analysointiin keskittyviä työkaluja ei tarkastella, vaan työssä keskityttiin tallentamiseen ja datan muotoon.

OpenCV:n natiivi datamuoto on Mat. Mat on n-ulotteinen, yksi- tai monikanavainen lista. Kanavien lukumäärä tarkoittaa montako asiaa jokaiseen listan kohteeseen on tallennettu. Listan sisältämä data voi olla boolean, integer tai float tyyppiä. (cv::Mat Class Reference n.d.) Tämä tarkoittaa sitä, että esimerkiksi harmaasävykuva on kaksiulotteinen ja yksikanavainen, jolloin jokaista pikseliä kohti on yksi väritieto. Värikuva on kaksiulotteinen ja kolmekanavainen, jolloin jokaisella pikselillä on kolme väriä. (Mat- The Basic Image Container n.d.) Pistepilven tilanteessa kolmen värin sijasta kanaviin kirjataan pisteen koordinaatit, tämä on esitelty tarkemmin kappaleessa 2.4. OpenCV tukee Mat tallentamista kuvana imwrite toiminnolla. Imwrite tallentaa Mat:in kuvatiedostona. Kuvaformaatit aiheuttavat rajoitteita, koska niillä pystyy tallentamaan vain kokonaislukumuuttujia. Tilanteissa, joissa tallennettava sisältö poikkeaa yleisistä kuvaformaateista, voi käyttää FileStorage toimintoa. (Image file reading and writing n.d.) FileStorage tallentaa datan luettavassa XML- tai YAML-muodossa. Samaan tiedostoon voi tallentaa useamman Mat:in sekä erillisiä muuttujia. FileStoragella tallennettaessa ja luettaessa Mat:in rakenne ja sisältö säilyy muuttumattomana, eikä erillisiä rajoitteita sisällölle ole. (File Input and Output using XML and YAML files n.d.)

Pistepilvien yleisiin tallennusmuotoihin ei OpenCV suoraan tarjoa tallennusmahdollisuutta, mutta OpenCV moduuli 3D Visualizer sisältää tallennuksen ja latauksen PLY-, XYZ- ja OBJ-formaateille (3D

Visualizer n.d.). Moduulin käyttäminen vaatii OpenCV:n kääntämistä lähdekoodista ja halutun moduulin lisäämistä siihen (Installation in Windows n.d.). Lisäksi moduuli vaatii VTK - visualization Toolkit asentamisen (Using the new highgui 3D visualization features n.d.). VTK on avoimen lähdekoodin ohjelmisto, jolla pystyy luomaan 3D grafiikoita (Overview n.d.).

2.4 Photoneo mittalaite

Phoxi 3D Scanner on paikallaan olevien esineiden skannaamiseen tarkoitettu laite. Laitteen toiminta perustuu valostruktuuritekniikkaan. PhoXi 3D-skanneri heijastaa valokuvioita kohteeseen ja laskee kaikkien näkyvien pintojen koordinaatit. Saatu pistepilvi lähetetään käyttäjälle ethernet-yhteydellä. (PhoXi 3D Scanner 2022, 7.) Työssä käyetty laite oli malliltaan PhoXi 3D Scanner M (Kuvio 2).

Kuvio 2. PhoXi 3D scanner M (PhoXi 3D Scanner 2022, 1)

Phoxi 3D-skannereita ohjataan PhoXi Control ohjelmistolla. Sillä voidaan säätää mittausasetukset ja tarkastella otettuja mittauksia. (Phoxi Control 1.7 2022, s.8.) PhoXi Control pystyy tallentamaan mittauksia Photoneo RAW data format (.praw), MotionCam-3D RAW data format (.pmraw), Stanfords PLY (.ply), Leica PTX (.ptx), Text file (.txt) ja Raw images in tif (.tif) tiedostomuodoissa. Photoneo RAW ja MotionCam-3D RAW ovat Photoneon natiiveja tiedostomuotoja, jotka voidaan avata PhoXi Control sovelluksella. (Phoxi Control 1.7 2022, s.24.)

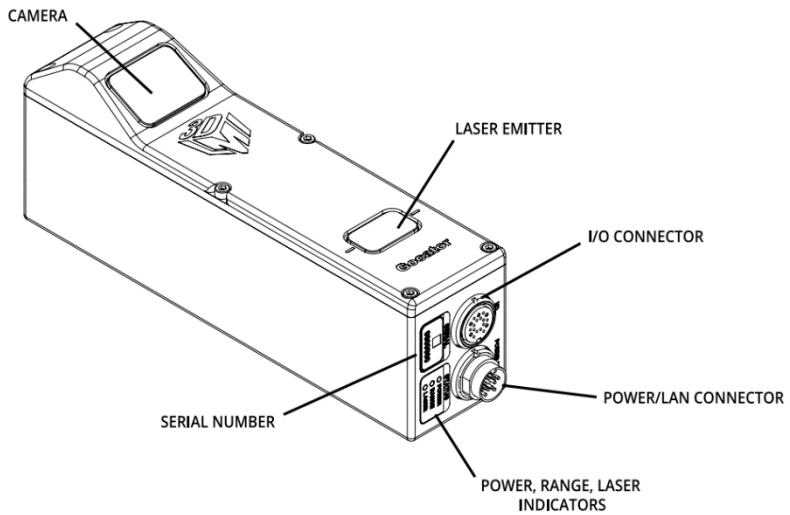
PhoXi Control sisältää ohjelmointirajapinnan PhoXi API, joka mahdollistaa sovellusten kehittämisen mittalaitteelle (PhoXi 3D Scanner 2022, 18). PhoXi API löytyy PhoXi Control asennuskansiosta. Se sisältää ohjelmointikirjaston, dokumentaation sekä ohjelmointiesimerkkejä c++ ja c# kielillä. PhoXi

Control täytyy olla käynnissä laitteella, koska PhoXi API ohjaa PhoXi Control sovellusta eikä suoraan mittalaitetta. (Phoxi Control 1.7 2022, 37.)

Työssä mainitut esimerkit ja PhoXi API dokumentaatio löytyy PhoXi Control asennussijainnin kansiosta ”API”. PhoXi API sisältää koodiesimerkin Minimal OpenCV, jossa esitellään datan muuttaminen OpenCV muotoon (mts. 38). Esimerkin MinimalOpenCV mukaan pistepilvi tallennetaan kaksiulotteisen ja kolmekanavaisen Mat-listan sisään. Tallennuksessa jokainen listan kohde vastaa laitteen mittaamaa kohtaa, ja siihen on tallennettuna kohdan sijainti xyz-koordinaatteina, yksikkönä millimetri. Lisäksi PhoXi API tukee tallentamista PLY-tiedostona, tämä on esitelty Full API Example:ssa. PLY-muotoista tallentamista on selvennetty API:n sisältämässä dokumentaatiossa tiedostossa a00161.html. Sen mukaan PhoXi API hyödyntää PLY:n projektikohtaista ominaisuuksien määrittelyä ja samassa PLY-tiedostossa voi olla tallennettuna: pistepilvi, pinnannormaali, korkeuskartta, tekstuuri ja luotettavuusarvo.

2.5 Gocator mittalaite

Työssä käytetty laite oli Gocator 2140 line profile sensor (kuvio 3). Gocator heijastaa laserviivan, jonka läpi mitattava kappale kulkee. Viivasta otetaan kuvia ja laite laskee viivan muodosta kappaleen pinnan muodon. Jokainen kuva tuottaa kapean siivun kappaleen muodosta. Laite yhdistää siivut ja tuottaa niistä 3D-mallin. Menetelmää kutsutaan laserkolmiomenetelmäksi. (Gocator Line Profile Sensors 2022, 61.)



Kuvio 3. Gocator 2140 (Gocator Line Profile Sensor 2022, 27)

Gocator toimii itsenäisenä mittalaitteena eikä tarvitse tietokonetta toimiakseen (mts. 33-34). Se pystyy suorittamaan kappaleen etsintää sekä mittauksia (mts. s.73-80). Gocatorin asetuksia säädetään sen web-käyttöliittymässä, joka aukeaa selaimella Gocatorin IP-osoitteessa. Käyttöliittymässä voi muokata laitteen mittausasetuksia sekä ulos lähetettävää dataa. (Mts. 90.) Käyttöliittymässä voi valita mittausten tallentamisen ”record” painikkeella. Tallennettu data voidaan avata Gocatorilla sekä ladata tietokoneelle kolmannen osapuolen työkaluilla avattavaksi (mts. 96). Manuaalista ei löytynyt mainintaa, millä työkaluilla data voidaan avata. Manuaalista ei myöskään löytynyt ohjeita 3D-mallien tai pistepilvien tallentamiseen.

Gocatoriin tarjotaan ohjelmistokehityksen kirjastot: Gocator Development Kit (GDK) ja Software Development Kit (GoSDK) (mts. 1064). GDK mahdollistaa laitteen sisäisten mittaustyökalujen kehittämisen ja lataamisen Gocatoriin (mts. 1075). GoSDK sisältää kirjaston, dokumentaation sekä ohjelmointiesimerkkejä, joiden avulla tietokonesovellus voi hallita Gocator -anturia (mts. 1064). GoSDK koodiesimerkeistä tai dokumentoinnista ei löytynyt ohjeita mittausten tallentamiseen.

3 Työn toteutus

Tallennusohjelma päätettiin toteuttaa niin, että se olisi mahdollisimman helppo ymmärtää ja jatkokehittää. Ohjelmointikieleksi valittiin C++, koska molempien laitteiden API sekä OpenCV tukee sitä. Ohjelma päätettiin rakentaa Visual Studio projektiin. Visual Studio on käytettävissä Jamkissa, ja sillä tehty projekti on helppo avata ja muokata. Toimeksiantaja halusi ohjelmaan visuaalisen käyttöliittymän. Käyttöliittymä tekee koodista hieman monimutkaisemman, mutta parantaa käytettävyyttä. Käyttöliittymän ohjelmointiin valittiin avoimen lähdekoodin wxWidgets kirjasto, koska sen käytöstä oli aiempaa kokemusta ja sen lisenssi sallii käytön ilmaiseksi. Käytetty wxWidgets versio oli 3.1.5. Käyttöliittymään ohjelmoitiin tarvittavat toiminnot laitteeseen yhdistämiseen ja tallennussijainnin valitsemiseksi. Tallennusohjelmalla suoritettiin testitallennus toiminnan todentamiseksi. Konenäön opettamiseen käytettävää dataa ei kerätty.

3.1 Tallennusmuodon valinta

Tallennusmuotona kannattaa käyttää teknologioille natiivimuotoa, koska sovellukselta halutaan yksinkertaisuutta. Valinta tehtiin vertailemalla tallennusmuotoja (Taulukko 1). Yleisistä pistepilvien tiedostomuodoista kaikki pystyvät tallentamaan mittalaitteiden pistepilven ja kaikki muut paitsi XYZ-tiedosto mittauksen lisätietoja. E57 vaikuttaa hyvältä, koska se on suunniteltu toimimaan yleisenä valmistajasta riippumattomana tallennusmuotona. Sille ei kuitenkaan ole suoraa tukea missään käytetyssä laitteessa eikä OpenCV kirjastossa. PLY tiedostoformaatti on natiivi sekä Photoneolle, sekä OpenCV 3D Visualizer moduulille. Gocator ei tue mitään tallennustapaa. 3D Visualizer moduulin asentaminen OpenCV projektiin on kuitenkin monimutkaista, koska se vaatii OpenCV:n ja haluttujen moduulien kääntämisen lähdekoodista. OpenCV FileStorage toiminto löytyy suoraan OpenCV:n valmiiksi käännetystä paketista ja sen käyttämiseen löytyy OpenCV:n nettisivuilta tutoriaali, jonka avulla tallennettu data on helppo lukea (File Input and Output using XML and YAML files n.d.).

	OpenCV tukee	Photoneo tukee	Gocator tukee	Tallentaa kaiken datan
XYZ	ei tue	ei tue	ei tue	ei
E57	ei tue	ei tue	ei tue	kyllä
PLY	vaatii lisä moduuleja	kyllä	ei tue	kyllä
OpenCV XML	kyllä	kyllä	ei tue	kyllä

Taulukko 1. Tallennusmuotojen vertailu

FileStoragen käyttö vaatii tallennettavan tiedon olevan Mat-luokassa, johon molempien laitteiden tuottama pistepilvi pitää tallentaa. PhoXi API tukee suoraan Mat-muotoon muuttamista. PhoXi API:n Minimal OpenCV esimerkin luoma Mat on kaksiulotteinen ja kolmekanavainen. Mittausdata on tallennettu 32 bit float muodossa. GoSDK:n sisältämässä esimerkissä ReceiveSurface esitellään pistepilven lukeminen laitteesta. Pistepilvi luetaan kaksiulotteista, kolmekanavaista, 64 bit float Mat:ia muistuttavaan luokkaan. Koodin luokka voidaan miltei suoraan korvata Mat luokalla (ks. kuvio 4). Koordinaatit kirjataan Point3d OpenCV muuttujaan, joka tallennetaan Mat listaan oikealle paikalle.

```
106
107         cv::Mat MatData;
108
109         MatData = cv::Mat::zeros(cv::Size(GoSurfaceMsg_Width(surfaceMsg), GoSurfaceMsg_Length(surfaceMsg)), CV_64FC3);
110
111
112
113         for (rowIdx = 0; rowIdx < GoSurfaceMsg_Length(surfaceMsg); rowIdx++)
114         {
115             k16s* data = GoSurfaceMsg_RowAt(surfaceMsg, rowIdx);
116
117             for (colIdx = 0; colIdx < GoSurfaceMsg_Width(surfaceMsg); colIdx++)
118             {
119                 // gocator transmits range data as 16-bit signed integers
120                 // to translate 16-bit range data to engineering units, the calculation for each point is:
121                 //          X: XOffset + columnIndex * XResolution
122                 //          Y: YOffset + rowIndex * YResolution
123                 //          Z: ZOffset + height_map[rowIndex][columnIndex] * ZResolution
124
125                 //surfaceBuffer[rowIdx][colIdx].x = XOffset + XResolution * colIdx;
126                 //surfaceBuffer[rowIdx][colIdx].y = YOffset + YResolution * rowIdx;
127                 cv::Point3d tempPoint;
128                 tempPoint.x = XOffset + XResolution * colIdx;
129                 tempPoint.y = YOffset + YResolution * rowIdx;
130
131
132                 datapoints++;
133                 if (data[colIdx] != INVALID_RANGE_16BIT)
134                 {
135                     //surfaceBuffer[rowIdx][colIdx].z = ZOffset + ZResolution * data[colIdx];
136                     tempPoint.z = ZOffset + ZResolution * data[colIdx];
137                 }
138                 else
139                 {
140                     //surfaceBuffer[rowIdx][colIdx].z = INVALID_RANGE_DOUBLE;
141                     tempPoint.z = INVALID_RANGE_DOUBLE;
142                 }
143
144                 MatData.at<cv::Point3d>(rowIdx, colIdx) = tempPoint;
145
146             }
147         }
```

Kuvio 4. Koodimuutokset OpenCV käyttöä varten

Koska molempien laitteiden muodostamat pistepilvet voidaan muuttaa OpenCV:n Mat luokkaan, tallennus muodoksi valittiin OpenCV:n filestorage toiminto. Se on yksinkertaisin tapa tallentaa pistepilvi OpenCV:llä avattavaksi. Lisäksi samaan tiedostoon voi tallentaa muitakin mittalaitteen tuottamia tietoja, kuten värikartan tai mittauksen varmuusarvion, kunhan ne ensin muutetaan Mat -muotoon.

3.2 Sovelluksen ohjelmointi

Aluksi luotiin Visual Studio projekti, joka sisältää OpenCV kirjaston sekä wxWidgets käyttöliittymäkirjaston. OpenCV kirjaston versioksi valittiin 3.10, koska photoneon Minimal OpenCV esimerkissä sanottiin sen toimivan versiolla 3.10. Molemmille laitteille luotiin omat ohjelmat, mutta käyttöliittymä ja ohjelmarakenne pidettiin mahdollisimman samana molempien ohjelmien välillä. Ohjelman perustoiminnot ovat: Laitteeseen yhdistäminen, datan vastaan ottaminen, datan muuttaminen Mat -muotoon ja tallennus. API:t mahdollistavat mittalaitteiden asetusten säätämisen, mutta sen pystyy tekemään helposti myös laitteiden omilla käyttöliittymillä, joten sitä ei toteutettu ohjelmaan. Ohjelmoinnissa noudatettiin soveltuvin osin API:n mukana tulleita ohjelmointiesimerkkejä.

Toteutettu ohjelma poikkesi rakenteeltaan esimerkeistä, koska esimerkit olivat konsoliohjelmia, eli niissä ei ole visuaalista käyttöliittymää.

3.3 Photoneo ohjelma

PhoXi Control API lisätään Visual Studio projektiin. Ohjelmassa noudatettiin API:n Minimal OpenCV esimerkin mukaista laitteeseen yhdistämistä ja kuvan kaappausta. Ohjelman rakenne poikkeaa esimerkistä, koska sen toiminnot on siirretty käyttöliittymään (Liite 1).

Laitteeseen yhdistämiseen luotiin oma ikkuna. Ikkunaan päivittyy lista saatavilla olevista laitteista. Käyttäjä valitsee listasta haluamansa Photoneo laitteen ja siihen yhdistetään Minimal OpenCV esimerkin mukaisella tavalla.

Mittaus suoritetaan Minimal OpenCV esimerkin mukaisella ohjelmallisella liipaisulla. Erona esimerkkiin on kuitenkin, että liipaisu suoritetaan, kun käyttäjä painaa painiketta ja otetaan vain yksi mittaus. Mittauksella saatu pistepilvi muutetaan Mat -muotoon esimerkin mukaisesti. Saatu Mat tallennetaan FileStorage funktiolla XML-tiedostoksi, käyttäjän valitsemaan paikkaan. Lisäksi ohjelmaan lisättiin valinta PLY-tiedoston tallentamisesta, koska API:n omalla tallennusmetodilla sen lisääminen oli helppoa.

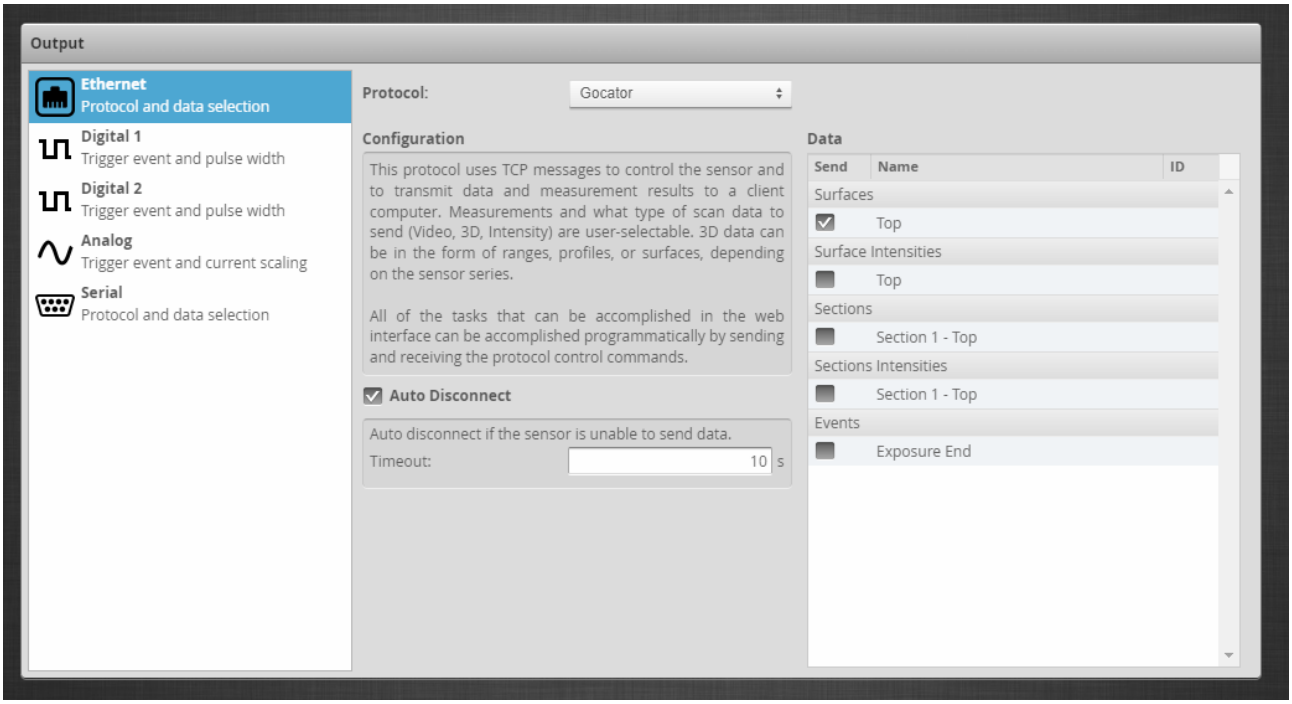
3.4 Gocator ohjelma

Gocatorin GoSDK ja kAPI lisättiin Visual Studio projektiin. Ohjelmassa noudatettiin GoSDK:n esimerkkejä ja seurattiin Photoneoa varten tehdyn ohjelman rakennetta, poikkeuksena laitteiden hakutoiminnon puuttuminen ja mittauksien odottaminen rinnakkaisessa säikeessä. Rakennetta on selvennetty liitteessä 2.

Gocatorin Discover esimerkki esittelee laitteiden etsimisen, mutta esimerkin ohjelma listaa vain jo yhdistetyt laitteet. Esimerkissä ei käytetä GoSDK:n dokumentoinnissa löytyviä GoSystem StartDiscovery- tai GoSystem EnableDiscoveryCompatibility -käskyjä. Dokumentoinnista ei selviä kuinka näitä tulisi käyttää, joten yhdistämisikkunaan ei saatu listaa saatavilla olevista laitteista. Listasta valinnan sijaan yhdistäminen laitteeseen suoritetaan GoSystem_FindSensorById -metodilla, jonka

käyttö esitellään ConsoleExample esimerkissä. Metodin vaatima ”id” on GoCator laitteen sarjanumero.

Kuvan kaappaus suoritetaan ReceiveAsync esimerkin mukaan. GoAPI luo säikeen, joka kuuntelee Gocator laitetta ja vastaanottaa sen lähettämiä viestejä. Gocatorissa pitää olla päällä ohjelma, jossa on joku mittauksen laukaiseva toiminto ja Output Ethernet asetuksista valittuna: Protocol = Gocator ja Surfaces Top (ks. kuvio 5).



Kuvio 5. Kuvan kaappaus GoCator asetuksista

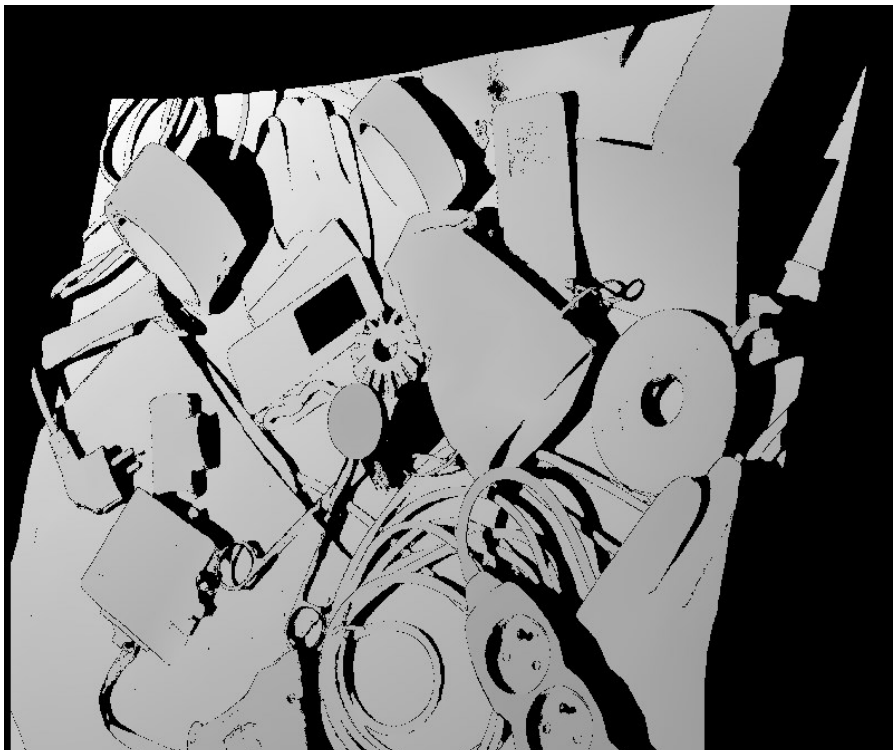
Gocatorin lähettämistä viesteistä odotetaan GoUniformSurfaceMsg -viestiä. Se sisältää pistepilven valitusta mittausalueesta. Pistepilvi otetaan vastaan ReceiveSurface esimerkin mukaisesti, mutta pisteet tallennetaan Mat luokkaan luvussa 3.1 esitellyllä tavalla. Mat:iin tallennettu pistepilvi tallennetaan FileStorage toiminnolla XML-tiedostoksi käyttäjän valitsemaan paikkaan.

4 Tulokset

Molemmille toimeksiantajan 3D-skannereille tuotettiin ohjelma, joka tallentaa pistepilven OpenCV:llä käsiteltävään muotoon. Ohjelmien toimivuus testattiin ottamalla niillä yhteys 3D-skanneriin ja tallentamalla mittaustulos. Ohjelmien lisäksi niiden lähdekoodi luovutettiin Jamk:lle.

Vaikka tallennusmuotoja pistepilven tallennukseen on paljon, valikoitui käytettäväksi muodoksi OpenCV:n oma tiedoston tallennustoiminto. FileStorage toiminnolla tiedoston tallennus ja lukeminen onnistuu helposti suoraan OpenCV:n Mat luokkaan, johon kaikki OpenCV:llä käsiteltävä data ladataan käsittelyä varten. Pistepilven lisäksi FileStorage mahdollistaa myös mittauksessa syntyvän lisädatan, kuten kohteen valokuvan, tallennuksen samaan tiedostoon. Muilla tiedostomuodoilla kuten PLY:llä voidaan saavuttaa sama lopputulos. PLY:n vapaan standardin antama mahdollisuus lisätä omia tietoja tekee siitä yhtä joustavan kuin FileStorage. PLY:n käyttäminen OpenCV:n kanssa kuitenkin vaatii lisäosien liittämistä ohjelmaan, koska perusmuotoinen OpenCV ei tue PLY-tiedostoja. Tämän takia sitä ei valittu tiedostomuodoksi.

Ohjelmat testattiin Jamkin laboratoriossa. Laitteet yhdistettiin verkkokaapelilla tietokoneeseen ja tietokoneeseen asetettiin samassa avaruudessa oleva IP-osoite. Photoneo laitteeseen yhdistäminen sujui helposti valitsemalla laite listasta. Sen jälkeen mittaus laukaistiin ja ohjelma loi XML tiedoston. Gocator vaati enemmän järjestelyä, koska sen sarjanumero piti selvittää ja käynnistää sen web-käyttöliittymässä sopiva ohjelma. Ohjelmaan valittiin mittauksen laukaisuksi, että laite havaitsee yli 5mm korkean kappaleen. Yhdistämisen jälkeen mittausten ottaminen oli helppoa. Laite otti mittauksen kappaleesta ja ohjelma tallensi sen automaattisesti. Molempien ohjelmien tuottamia tiedostoja tarkasteltiin avaamalla ne. Tiedostot sisälsivät pistepilven.



Kuvio 6. Tiedoston sisältö esitettynä kuvana.

Kuviossa 6 näkyy pistepilven korkeus eli z koordinaatti esitettynä harmaasävykuvana. Kuviossa oleva data on tallennettu PhoXi controllin offline testaukseen tarkoitetusta mittalaite simulaattorista. Tallennusohjelman luoma XML tiedosto avattiin OpenCV:llä ja se muutettiin kuva formaattiin. Datan alussa ja lopussa on mustaa, koska siinä mittalaite ei havaitse kappaletta säädetyn mittausalueen sisällä. Samoin kappaleet jättävä ”varjon” paikkoihin joihin skannerin lähettämä valokuvio ei osu. Nämä paikat ovat tiedostossa koordinaatteina 0,0,0 (liite 3).

5 Jatkokehitys

Ohjelmien tarkoitus on datan keruu koneoppimista varten, luonnollinen jatkokehityksen kohde on koneoppimisen testaaminen. Testaamista varten pitää valmistella mittalaitteille hyvät mittausolosuhteet ja kerätä riittävä määrä mittauksia. Koska molemmissa ohjelmissa kerätty mittausdata muutetaan suoraan OpenCV natiiviin formaattiin, on koneoppimisella tuotetut työkalut mahdollista liittää suoraan ohjelmaan tosielämän testausta varten.

Ohjelmat tallentavat vain pistepilven tiedostoon. Tallennukseen voisi lisätä toiminnallisuuden myös muiden tietojen, kuten pinnan normaalin ja värin tallennukselle. Gocator-ohjelmaan olisi myös kannattavaa lisätä tallennus jollekin yleisesti käytössä olevalle tiedostoformaatille, kuten PLY. Tämä mahdollistaisi kerätyn datan käyttämisen muissa ohjelmistoissa.

OpenCV ei ole ainoa pistepilvien käsittelyyn kykenevä kirjasto. Esimerkiksi Point Cloud Library (PCL) on avoin pistepilviin ja kolmeulotteiseen dataan keskittyvä kirjasto. PCL testausta varten voisi sen liittää nykyisiin ohjelmiin ja tallentaa kerätyn datan samalla myös PCL-tiedostona.

6 Pohdinta

Tuotettu ohjelma antaa pohjan vapaan lähdekoodin kirjastojen kuten OpenCV:n ja Point Cloud Libraryn käyttämiseen. Tämä mahdollistaa skannausten tutkimisen opiskelija ystävälliseen hintaan, koska monet kolmannen osapuolen ohjelmat ovat hintavia. Lisäksi se mahdollistaa työkalujen kehittämisen tilanteisiin, joihin valmista ratkaisua ei vielä ole. Jos esimerkiksi mittausdata on huonoa, johtuen vaikka materiaalin heijastuksista tai liike nopeudesta, voidaan yrittää käyttää koneoppimista tunnistamaan kappaleita huonosta datasta.

Työn aikana nousi esille useita huomioita laitteista sekä niiden ohjelmistoista ja API käytännöistä. Gocator ei käytännössä sisällä mittausten tallennusmahdollisuutta. Tämä ei ole selvästi esillä Gocatorin dokumentaatiossa ja voi tulla yllätyksenä. Mittausten tallennus vaatii vähintään työn mukaisen ohjelman kirjoittamisen. Laite on suunniteltu toimimaan kokonaan itsenäisesti ja on hyvä siinä käytössä, jos sen valmiit mittaustyökalut riittävät tai käyttäjällä on tarpeeksi ohjelmointiosaamista hyödyntää sen kehitystyökaluja. Molempien laitteiden API:n dokumentointi oli heikkoa. API:sta löytyy toimintoja, joille ei ole dokumentaatiota tai ne eivät yksinkertaisesti toimi. API:n lisääminen Visual Studio projektiin tai edes esimerkkikoodien ajaminen oli vaikeaa automaatiotekniikan opintojen pohjalta.

Pistepilvien tallentamiseen käytettävistä tiedostomuodoista oli vaikea löytää tietoa. Monesta yleisesti käytössä olevasta muodosta ei ollut saatavilla selkeää dokumentointia. Tämä voi olla yksi syy siihen, että ohjelmistovalmistajat ovat päättäneet kehittää omat tiedostomuotonsa. Lisäksi useat tiedostomuodot ovat hyvin avoimia ja niihin voi lisätä elementtejä haluamallaan tavalla. Tämä voi aiheuttaa ongelmia, koska yleistä nimeämiskäytäntöä ei ole.

Koen työssä saatujen tulosten olevan erittäin toistettavia. Ne perustuvat vahvasti laitteiden ja OpenCV kirjaston omiin esimerkkeihin. Koska OpenCV on laitteistosta riippumaton, voidaan työtä yleistää muihinkin 3D-skannereihin, kuin mitä työssä käytettiin. Kun laitteessa on ohjelmointirajapinta ja siihen löytyy vastaavat esimerkit yhdistämisestä ja mittauksen vastaanottamisesta, siihen voi tehdä vastaavan ohjelman.

Lähteet

3D Visualizer. N.d. OpenCV dokumentaatio. Viitattu 4.1.2023. https://docs.opencv.org/3.1.0/d1/d19/group__viz.html.

About. N.d. OpenCV verkkosivujen tietoja-sivu. Viitattu 4.1.2023. https://opencv.org/about/.

cv::Mat Class Reference. N.d. OpenCV dokumentaatio. Viitattu 4.1.2023. https://docs.opencv.org/3.1.0/d3/d63/classcv_1_1Mat.html.

File Input and Output using XML and YAML files. N.d. OpenCV tutoriaali. https://docs.opencv.org/3.1.0/dd/d74/tutorial_file_input_output_with_xml_yml.html.

Gocator Line Profile Sensors. 2022. Korjattu painos D. Käyttöopas LMI Technologies verkkosivulla. Viitattu 4.1.2023. https://lmi3d.com/wp-content/uploads/2021/10/15159-6.1.32.12_MANUAL_User_Gocator_Line_Profile_Sensors_EN-US-5.pdf.

Image file reading and writing. N.d. OpenCV dokumentaatio. Viitattu 4.1.2023. https://docs.opencv.org/3.1.0/d4/da8/group__imgcodecs.html.

Installation in Windows. OpenCV dokumentaatio. Viitattu 4.1.2023. https://docs.opencv.org/4.x/d3/d52/tutorial_windows_install.html#tutorial_windows_git-bash_build.

Kananen, J. 2015. Kehittämistutkimuksen kirjoittamisen käytännön opas. Jyväskylän: Jyväskylän ammattikorkeakoulu.

libE57: Software Tools for Managing E57 Files. N.d. Viitattu 4.1.2023. http://www.libe57.org/.

Mat- The Basic Image Container. N.d. OpenCV dokumentaatio. Viitattu 16.1.2023. https://docs.opencv.org/3.1.0/d6/d6d/tutorial_mat_the_basic_image_container.html.

Mizerák, M., Rosocha, L. & Trebuňa, P., 2018. 3D Scaning - Technology And Reconstruction. Acta Simulatio. Viitattu 4.1.2023. https://www.actasimulatio.eu/issues/2018/III_2018_01_Trebuna_Mizerak_Rosocha.pdf.

Overview. N.d. Visualization Toolkit (VTK) verkkosivut. Viitattu 4.1.2023. https://vtk.org/about/#overview.

PhoXi 3D Scanner. 2022. Käyttöopas. Viitattu 4.1.2023. https://www.photoneo.com/files/manuals/PhoXi3DScanner_UserManual.pdf.

Phoxi Control 1.7. 2022. Käyttöopas. Viitattu 4.1.2023. https://www.photoneo.com/files/dw/dw/pxc/1.7/PhoXiControl1.7-UserManual-02-2022.pdf.

Polygon File Format (PLY) Family. 2019. Tallennusformaatin kuvaus Library of Congress verkkosivuilla. Viitattu 4.1.2023. https://www.loc.gov/preservation/digital/formats/fdd/fdd000501.shtml.

The PCD (Point Cloud Data) file format. N.d. Point Cloud Library dokumentaatio. Viitattu 4.1.2023. https://pointclouds.org/documentation/tutorials/pcd_file_format.html.

Thomson, C., 2018. Common 3D point cloud file formats & solving interoperability issues. Viitattu 4.1.2023. https://info.vercator.com/blog/what-are-the-most-common-3d-point-cloud-file-formats-and-how-to-solve-interoperability-issues

Using the new highgui 3D visualization features. N.d. OpenCV verkkosivut. Viitattu 4.1.2023. https://opencv.org/using-the-new-highgui-3d-visualization-features/.

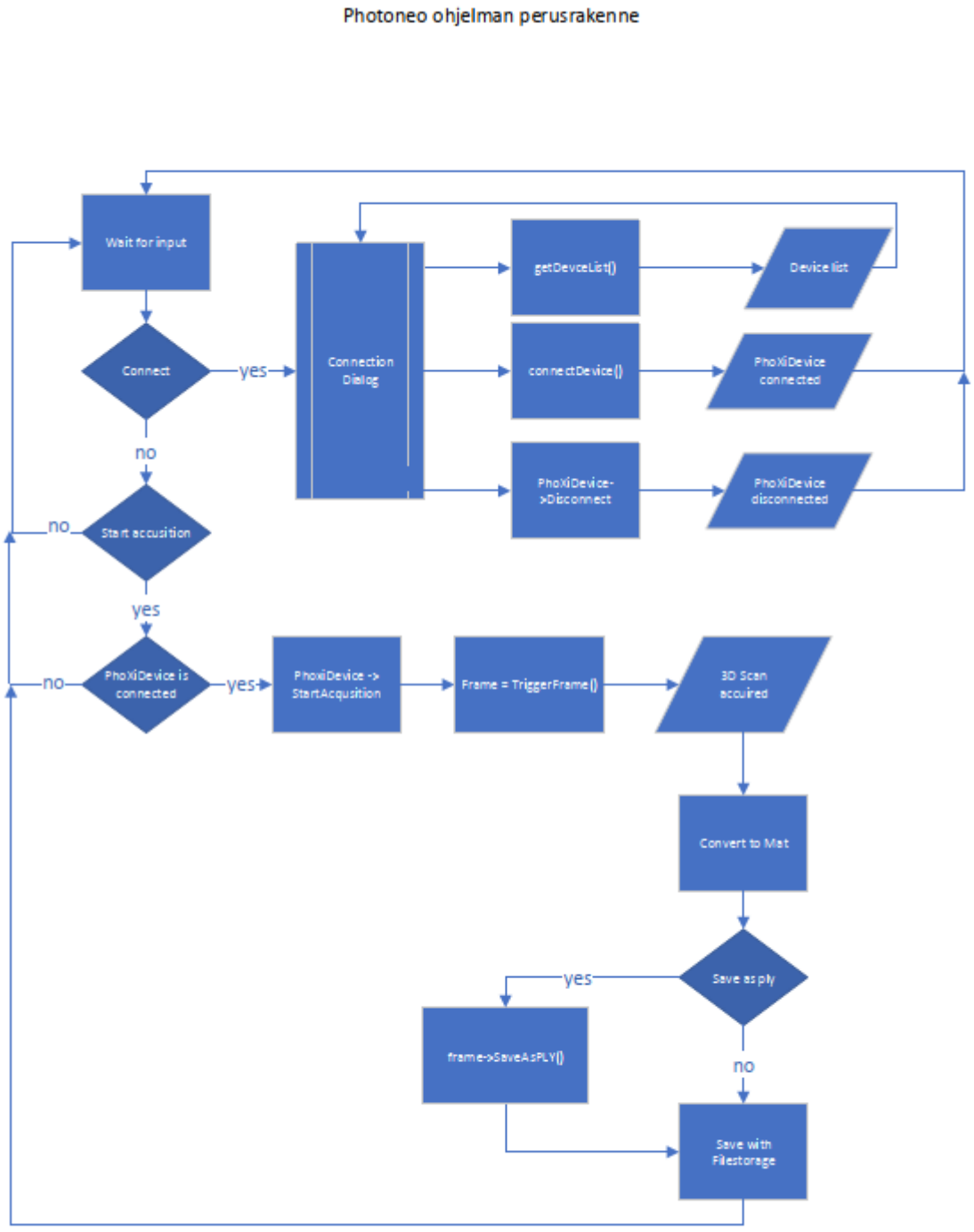
What is 3D scanner used for? N.d. Polyga verkkosivuilla. Viitattu 22.1.2023. https://www.polyga.com/3d-scanning-101/what-is-3d-scanner-used-for/.

What is computer vision? N.d. Artikkeli IBM verkkosivuilla. Viitattu 22.1.2023. https://www.ibm.com/topics/computer-vision.

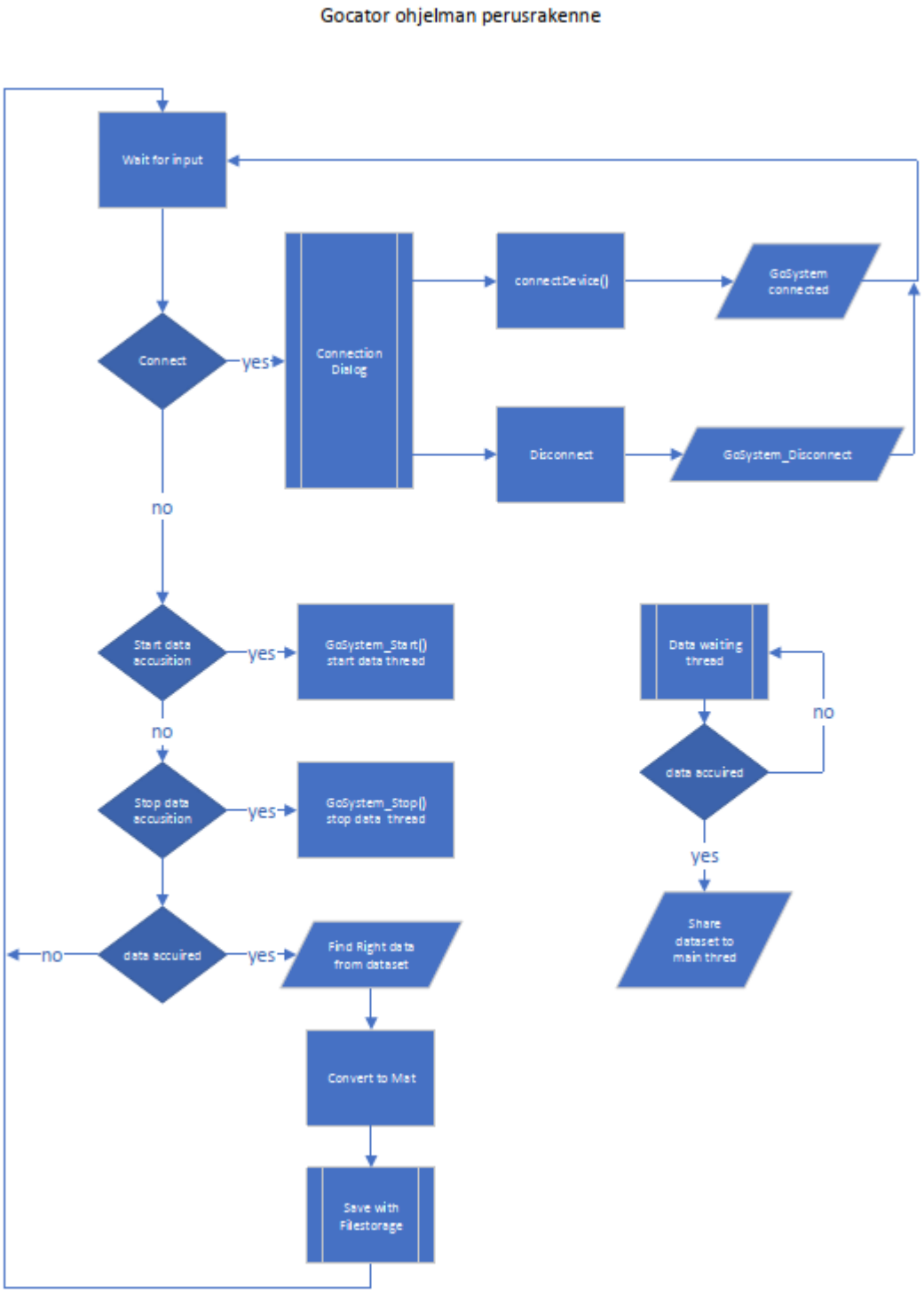
Wavefront OBJ File Format. 2020. Tallennusformaatin kuvaus Library of Congress verkkosivuilla. Viitattu 4.1.2023. https://www.loc.gov/preservation/digital/formats/fdd/fdd000507.shtml.

Liitteet

Liite 1. Photoneo ohjelman rakenne



Liite 2. Gocator ohjelman rakenne



Liite 3. Tallennetun tiedoston sisältö

Leikkeitä OpenCV:n luoman XML tiedoston sisällöstä

test1.xml


```
<?xml version="1.0"?>
<opencv_storage>
<OpenCV type_id="opencv-matrix">
  <rows>1544</rows>
  <cols>2064</cols>
  <dt>"3f"</dt>
  <data>
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

    -2.92797333e+02 2.15717812e+01 7.55231628e+02 -2.92379059e+02
    2.15662956e+01 7.55058350e+02 -2.92003113e+02 2.15639229e+01
    7.54994019e+02 -2.91643219e+02 2.15627308e+01 7.54971069e+02
    -2.91194641e+02 2.15549679e+01 7.54718079e+02 -2.90826508e+02
    2.15531502e+01 7.54673218e+02 -2.90405701e+02 2.15474205e+01
    7.54491333e+02 -2.90034515e+02 2.15453644e+01 7.54438110e+02
    -2.89628204e+02 2.15406895e+01 7.54293152e+02 -2.89276337e+02
    2.15400562e+01 7.54289795e+02 -2.88895538e+02 2.15372620e+01
    7.54210754e+02 -2.88506195e+02 2.15338268e+01 7.54109192e+02
    -2.88146942e+02 2.15326290e+01 7.54086060e+02 -2.87753571e+02
    2.15288773e+01 7.53973450e+02 -2.87317993e+02 2.15219574e+01
    7.53749817e+02 -2.86925720e+02 2.15182629e+01 7.53639221e+02
    -2.86588806e+02 2.15187168e+01 7.53673889e+02 -2.86182800e+02
    2.15139809e+01 7.53526733e+02 -2.85818359e+02 2.15123596e+01
    7.53488708e+02 -2.85418549e+02 2.15080681e+01 7.53357178e+02
    -2.85069824e+02 2.15076199e+01 7.53360291e+02 -2.84674072e+02
    2.15036201e+01 7.53238953e+02 -2.84291840e+02 2.15006313e+01
    7.53152954e+02 -2.83865692e+02 2.14943123e+01 7.52950378e+02
    -2.83520874e+02 2.14941387e+01 7.52963074e+02 -2.83153137e+02
    2.14922276e+01 7.52914917e+02 -2.82727722e+02 2.14859295e+01
    7.52712952e+02 -2.82316010e+02 2.14806538e+01 7.52546936e+02
    -2.81984131e+02 2.14814510e+01 7.52593628e+02 -2.81601593e+02

    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
    0. 0.</data></OpenCV>
</opencv_storage>
```