eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Proiektua

3D ikusmen-sistemen softwareen konfigurazioa eta konparazioa

Egilea

Josu Arruarte

informatika fakultatea
facultad de informática

2017

Laburpena

Dokumentu honetan aurkezten den gradu amaierako proiektua *robotika* alorrean kokatzen da, zehazki *ikusmen artifiziala* izeneko esparruan. Proiektua kamera eta proiektore independente batzuekin osaturiko 3D eskaner baten softwarearen bilaketan eta azterketan zentraturik dago. Hiru fase nagusitan banatu da proiektua. Lehenik eta behin, sistemaren azterketa eta softwareen bilaketari ekiten zaio, ondoren aukeratutako softwareen instalazioa, konfigurazioa eta kalibrazioa egiteko (SLStudio, TI3D eta DAVID). Azkenik, software horien arteko konparazioa egingo da, ikusmen artifizialaren alorrean erabiltzen diren hainbat ezaugarri kontuan izanik. Konparazioan, hainbat proba ezberdin burutuko dira software ezberdinen bitartez lorturiko puntu-hodeiekin: prozesatu gabeko puntu-hodeien konparazioa, lerrokatzeen arteko konparazioa, berreraikitze osoen arteko konparazioa eta puntu-hodeien analisi zuzena. Proiektuaren garapen osoan, eta ahal den neurrian, software librea erabiliko da.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera eta helburuak

Atal honetan, lehenik eta behin, proiektuari buruzko sarrera labur bat egingo da, proiektuaren sorkuntzarako eman zen motibazioari buruz hitz eginez eta, horrez gain, proiektuak bete beharko lituzkeen helburuak ere aipatuko dira. Gainera, proiektuaren irismenari buruzko informazioa ere aurkituko duzue: Lanaren deskonposaketa-egitura, emangarriak...

1.1 Motibazioa

Esan dezakegu, gaur egun, beste iraultza teknologiko berri bat bizitzen ari garela robotikan ematen ari diren aurrerapausoak direla eta. Industrian soilik erabiltzetik, gure eguneroko bizitzan gero eta presentzia gehiago izatera pasatzen ari da. Duela urte batzuk sinesgaitz egingo litzaizkiguken teknologiak jada erabiltzen hasiak dira eta, zenbait urte barru, ohikoak izango dira gure artean.

Robotikari buruz hitz egiten ari garenean, mota askotako makinak etor lekizkiguke burura; beso robotikoetatik hasi eta gizaki itxura duten androideetaraino. Robotak ikusten ditugunean, gehienetan, egiten dituzten mugimendu eta ekintzek bereganatzen dute gure arreta. Baina, ezin dugu ahaztu, mugimendu gehienen atzean dagoen funtsezko teknologia bat: *ikusmen artifiziala*. Bai robotaren eremua ezagutzeko, segurtasuna bermatzeko, landu behar dituen objektuak lokalizatzeko eta baita piezen gainean kalitate-kontrola burutzeko ere, 3D ikusmenaren presentzia behar beharrezkoa izaten da, gehienetan.

Aipatu bezala, asko dira 3D ikusmenaren aplikazioak eta, horri esker, asko dira teknologia

hori aurrera eramateko aurkitu ditzakegun teknika eta baliabideak. Zenbaitetan robotak edo sistemak irudi azkar baten beharra izaten du, inguruneko aldaketetara azkar erantzun ahal izateko. Beste batzuetan berriz, irudiaren kalitatea izango da robot edo sistemari interesatuko zaiona, eta kasu horretan, ez da hain garrantzitsua izango irudia lortzeko behar den denbora, bai ordea lorturiko berreraikitzea errealitatetik ahalik eta gertuen egotea. Berreraikitzeak, industrian, oso erabiliak dira; piezen lokalizaziorako, objektuen 3D modeloak (CAD) sortzeko eta, batez ere, objektuen kalitate-kontrola egiteko.

Bestalde, proiektuaren motibazioa ondo ulertzeko, hilabete batzuk atzera egin beharra daukagu. Izan ere, 2016 udan borondatezko praktikak burutu nituen *Tecnalia, Research & Innovation* korporazio teknologikoan. Bertan, ikusmen artifizialeko 2D eta 3D sistemekin lan egin nuen robot batek pieza batean jarritako sigilatzailearen kalitate-kontrola burutuz. Horretarako, ikusmen artifiziala aurrera eraman ahal izateko erabiltzen diren tekniketako bat erabili nuen, *argi egituratua* hain zuzen ere. DAVID sistema komertziala erabili nuen hain zuzen ere, eta software horren bitartez kalitate handiko berreraikitzeak lortu nituen. Baina, DAVID softwareak, arazo nagusi bat zuen: sistema itxia eta komertziala zen.

Alde batetik, komertziala izatetik, sistema hori erabili nahi dugun bakoitzean lizentzia baten beharra izango genuke, gastu ekonomiko bat suposatuz; eta bestetik, sistema itxia denez, ez ginateke gai izango bere funtzionamendua barrutik aztertu eta guk nahi genituzkeen aldaketa guztiak burutzeko, ezta sistema osotasunean ulertzeko ere.

Hortaz, arazo horiek konpontzeko asmoz, proiektu berri hau sortzen da, berreraikitzeak lortzeko sistema bat konfiguratzeko asmoarekin, baina software librea soilik erabiliz.

1.2 Helburuak

Proiektu honen helburu nagusiak honakoak dira:

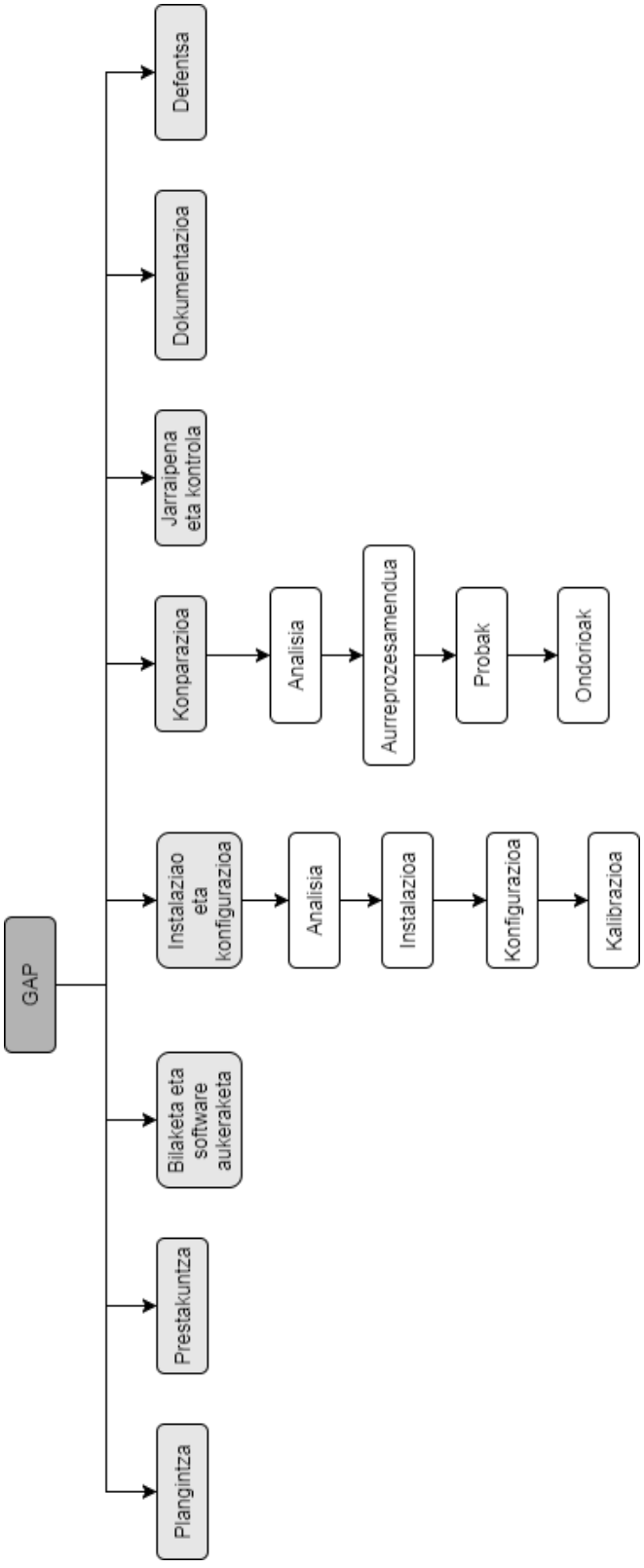
- 3D ikusmen-sistema batean erabiltzeko software libreen bilaketa, instalazio eta azterketa burutzea.
- Softwareen arteko konparazioa egitea, ikusmen artifizialaren industrian ohikoak diren hainbat teknika erabiliz.
- Proiektu osoaren garapena software librearen bitartez egitea.

1.3 Proiektuaren irismena

Jarraian datorren atal honetan, proiektuaren irismenari buruz hitz egingo dugu. Bertan aipatuko diren atalak proiektuaren betekizunak, lanaren-deskonposaketa egitura, emangarriak eta mugarriak izango dira.

1.3.1 Lanaren deskonposaketa-egitura

Hurrengo orrialdean, lanaren deskonposaketa-egitura aurkitzen da (ikus 1.1 irudia):



1.1 Irudia: LDE diagrama

- **Plangintza:** Burutu behar dugun proiektua aurrera modu egokian eraman ahal izateko, egin beharreko lanaren analisi bat egingo da, lana antolatuz. Egingo diren ekintzak zehaztuko dira, arriskuak identifikatu, betekizunak zehaztu eta horietan oinarritutako irizpide batzuk ezarri, eta azkenik, denboran zehar lana antolatu.

- **Prestakuntza:** Gure proiektua modu egokian aurrera eraman ahal izateko, burutu beharreko fase bakoitzaren aurretik, prestakuntza-fase bat behar dugu. Lehenik eta behin, ikusmen artifizialaren munduan murgilduz, gaur egun eskaintzen diren aukera ezberdinak ezagutu beharko dira. Behin argi egituratuaren eremuan sarturik, aurki ditzakegun eredu eta teknikak ezagutu beharko dira eta baita hauek instalatu eta kalibratzeko metodoak ere. Azkenik, konparazioa egin ahal izateko lortutako emaitzak prozesatu eta aztertzeko beharrezko metodoak ezagutu beharko ditugu.

- **Bilaketa:** Behin aurreko fasea gainditua dugunean, nahiz eta proiektu guztian gure prestakuntza zabalduz joango den, bilaketa-fasean sartuko gara. Gure helburua aurrez erabiltzen den produktu komertzial bat software librean oinarrituriko tresnen bidez ordezkatzea denez, bilaketa bat gauzatu beharko dugu lehen aukeraketa bat eginez, gehien komeni zaizkigunak aukeratuz.

- **Instalazioa eta konfigurazioa:** Atal hau bost fase nagusitan banatuko da, izan ere, aurrez aurrera eramango dugun bilaketa gauzatu ostean, aukeratutako software bakoitzarekin honako pausu hauek eman beharko ditugu:

  - Analisia
  - Instalazioa
  - Konfigurazioa
  - Kalibrazioa
  - Patroien hautapena

- Konparazioa: Softwareen azterketarekin bukatzeko, konparazio bat burutuko dugu. Aukeratutako softwareekin egiteaz gain, jada ezaguna dugun DAVID software komertzialarekin ere egingo dugu konparazioa.

- **Jarraipena eta kontrola:** Ataza honetan, proiektuaren hasieran ezarritako helburu eta epeak lortutakoekin alderatuko dira, plangintza modu egokian betetzen ari den ikusiz. Horrela, arazoren bat dagoela atzeman ezkero, berandu izan aurretik konponduko da.

- **Dokumentazioa:** Proiektuaren memoria idatziko da ataza honetan.
- **Defentsa:** Ataza honetan proiektuaren defentsa modu egokian aurrera eramateko beharrezko lana egingo da.

1.3.2 Emangarriak

Proiektu honen emangarriei dagokienez, bi talde nagusitan banatuko dira. Alde batetik, produktuarekin lortuko ditugun emangarriak daude eta bestetik, proiektuarekin eta horren dokumentazioarekin lortuko ditugunak.

Produktuarekin lortutakoak

- Gure sistema, software libre batekin konfiguratuta eta kalibratuta, berreraikitzeak burutzeko prest.
- Instalazioak nola egin azaltzen duen gida.
- Sistema ezberdinak konparatu eta aztertzen dituen dokumentua.

Proiektuarekin lortutakoak

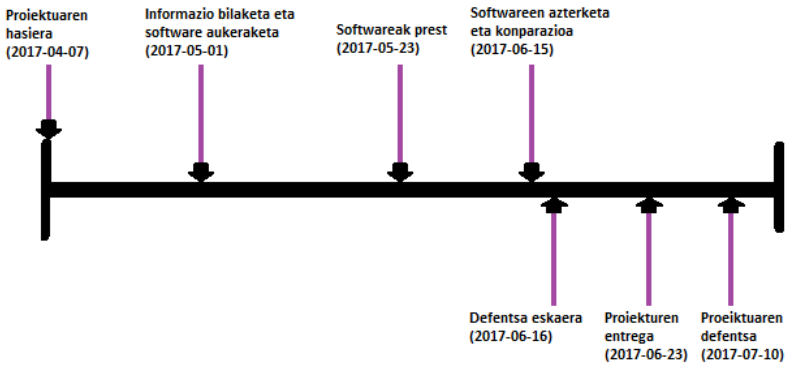
- Memoria: Proiektuari buruzko informazio gehiena egongo da bertan, horien artean, proiektuaren helburu-dokumentua, proiektuaren garapenaren jarraipena, eta proiektuaren bukaeran ateratako ondorioak. Baita, proiektuari lotutako xehetasun garrantzitsuenak ere.
- Aurkezpena: Proiektuaren defentsan erabiliko den dokumentua izango da honakoa.

1.3.3 Mugarriak

Gure proiektua modu ordenatu batean aurrera eramateko eta denboraren kontrola izaten saiatzeko hainbat epe ezarri genituen. Izan ere, epe bakoitza modu zehatzean ezartzeko aukerarik ez badago ere, modu horretan atzerapen edo aurrerapenak erreferentzia baten bitartez kontrolatzeko aukera izango dugu.

1.2 irudian ikus dezakezue gure proiektuko Gant diagrama eta 1.3 irudian berriz, mugarri guztiak denboran zehar antolaturik ikus ditzakezue.

1.2 Irudia: Gant diagrama



1.3 Irudia: Mugarrien grafikoa

Barne-mugarriak

- Informazio bilaketa eta software aukeraketa (2017/05/01)
  Egun horretarako, proiektua garatzeko beharrezkoa den informazioa bilduko da eta instalatuko diren softwareak aukeratuko dira.

- Softwareen instalazio, konfigurazio eta kalibrazioa (2017/05/23)
  Egun horretarako, aurreko fasean aukeraturiko softwareak instalatuta, konfiguratuta eta kalibratuta egon beharko dute.

- Softwareen azterketa eta konparazioa (2017/06/15)
  Egun horretarako, aurreko fasean gure sisteman martxan jarri ditugun softwareen arteko konparazioak amaiturik egon beharko du.

Kanpo-mugarriak

- Defentsa-eskaera (2017/06/16)
  Egun honetarako, GAURen proiektuaren matrikula egin beharko da eta defentsa eskaera egin.

- Proiektuaren entrega (2017/06/23)
  Egun honetarako proiektuak bukatua egon beharko du eta ADDI plataformara igota.

- Proiektuaren defentsa (2017/07/10-13)
  Egun horietako batean proiektuaren defentsa izango da eta beraz, ordurako, aurkezpenak prestatua egon beharko du.

Ikus daitekeen moduan, mugarriak bi taldetan banatu ditugu, izan ere ezaugarri ezberdin nagusi bat dute beraien artean. Barne-mugarriak guk ezarriak dira proiektua aurrera modu egokian eraman ahal izateko eta kanpo-mugarriak, berriz, kanpotik ezarriak daude. Hori dela eta, barne-mugarriak ez betetzeko aukera daukagu, proiektuaren beharrek hala eskatzen badigute. Kanpo-mugarriak berriz, ez dute aldatzeko aukerarik eta beraz, epeak nahitaez errespetatu beharko ditugu, defentsa-epea mantendu nahi bada, noski.

1.3.4 Arriskuen analisia

Soberan dakigu proiektu baten garapenena eta erresoluzioa baldintzan jarri dezaketen arriskuak daudela. Hori dela, gure proiektuan arrisku hauek izan dezaketen eragina aztertu eta kalteak minimizatzeko, analisi bat burutu da.

Identifikatutako arriskuak

- Erabilitako hardwareak edonolako matxura bat izatea.
- Egin beharko ditugun instalazio-, konfigurazio- edo kalibrazio-faseren bat gainditzeko gai ez izatea.
- Edozein informazioaren galera izatea.
- Planifikazioan ezarri ditugun epeak ezin bete ahal izatea sorturiko lan-kargagatik edo proiektuan zehar aurkitu ditzakegun arazoengatik.

Arriskuei aurre egiteko planak

- Badakigu garatu behar dugun proiektuak hardware espezifiko baten menpekotasuna duela. Hori dela eta, proiektua ekin aurretik, hardware horren aldaketa burutzeko elementuak ditugula ziurtatzea komeniko litzateke. Instalazioaren kasuan, edozein matxura gertatuz gero kalteak minimizatzeko, jarraipen zehatz bat eman beharko dugu.
- Aukeraketa-fasean hainbat software aukeratuko dira sisteman instalatuak izan daitezen. Horietariko batekin arazoak egonez gero, aukera gehiago izango ditugu erabilgarri. Software guztietan arazoak egonez gero, bilaketa zabaltzen saiatu beharko genuke.
- Proiektua Tecnalia enpresako ordenagailu batean eta nire ordenagailu pertsonalean garatuko da. Hori dela eta, informazioa periodikoki sarera igoko da edozein arazoren aurrean informazio galerarik egon ez dadin. Erabiliko den plataforma *Google Drive* izango da.
- Proiektua planifikatu bezala baldin badoa, kanpo-faktoreek ez dute eragina izango proiektuari eskaini beharreko denboran. Hala ere, proiektuan denbora-galerak eman

ditzaketen arriskuak daudenez, ekainerako proiektua amaitzea lortzen ez bada, irailean aurkeztu beharko da.

2. KAPITULUA

Aurrekariak

Atal honi hasiera emateko, lehenik eta behin, motibazioan aipatu berri dugun udako praktikei buruz hitz egingo dugu, hau izan baitzen 3D ikusmenarekin izan genuen lehen kontaktua. Ondoren, 3D ikusmenari helduko diogu zuzen-zuzenean; teknologia aztertu, dituen aukera ezberdinak ikusi eta gure aukeraketa azaldu. Behin hori eginik, udan erabili genuen sistemari buruz hitz egingo dugu, bere xehetasunak gainetik azalduz, eta jarraian, proiektu honetan erabiliko den hardwarea zein den azalduko dugu.

2.1 Udako praktikak

Esan dudan bezala, proiektu honen aurretiko gisa, 2016ko maiatzak 23tik 2016ko abuztuaren 2ra *Tecnalia, Innovation & Research*-en 3D sistemen arloan borondatezko praktikak egin nituen. Praktika horiek elkarlanerako robotikako taldean egin nituen, ikusmen artifizialeko eremuan hain zuzen ere.

Lehenik eta behin, taldearen lan egiteko modua ezagutu eta bertako proiektuak ikusi eta ezagutu nituen, 3D ikusmenean zentraturiko proiektu batean lanean aritzeko. Proiektua kalitate-kontrolean zentraturik zegoen, automatikoki egitera saltoa eman nahi zuen prozesu batean, zehazki. Prozesuari dagokionez, beso robotiko batek sigilatzailea aplikatzen zuen zenbait junta eta torlojuren gainean. Sigilatzailea ondo aplikatua geratu zen konprobatu beharra zegoen eta hori burutzea, pertsonentzat ere prozesu nekeza eta arriskutsua izan zitekeen. Hor sartzen zen jokoan 3D ikusmena, sigilatzailedun piezak berreraikiz eta

horien kalitatea aztertuz. Pieza horiek berreraikitzeko, momentu horretan ikusmen artifizialeko taldea erabiltzen ari zen DAVID 3D sistema itxi eta komertziala erabili genuen.

Software horrek berreraiki nahi genuen objektuaren puntu-hodei bat lortzen zigun, hau da, pieza hiru dimentsiotan irudikatzen zuen puntuak erabiliz.

Behin hori lorturik, puntu-hodeien tratamendurako erabiltzen diren *Point Cloud Library (PCL)*[1] librurutegiak erabili genituen, sigilatzailearen kalitatea neurtu nahian. Automatizatzen saiatu ginen ekintzen artean, honakoak ditugu:

- Segmentazioa: Algoritmo ezberdinen bitartez elementu ezberdinak identifikatu genituen: planoak, zirkunferentziak...
- Puntu-hodeia jatorrira mugitu teknika ezberdinak erabiliz, zentroidearen bitartez adibidez.
- Puntu-hodeian neurketak egin.
- Puntu-hodeiaren normalak kalkulatu eta hauek aztertu.
- Sigilatzailearen inguruko hainbat neurketa egin metodo ezberdinak erabiliz.
- Aztertzera gindoazen sigilatzailea plano baten gainean baldin bazegoen ere, teknika matematikoak erabili genituen planoa zilindro bihurtu eta azterketa bertan aurrera eramateko. Izan ere, etorkizunean sigilatzailea zilindro baten gainean aplikatuko zen.

2.2 Aurrekariak

Egungo proiektuaren bilaketari ekiteko, lehenik eta behin, jada merkatuan zeuden zenbait 3D sistema aztertu genituen, horiek nola jokatzen zuten ikusiz eta gure aukeraketa egokia izan zedin informazioa bilduz. Azterketa egiteko, oinarritzat, DAVID 3D eskanerra erabili genuen, izan ere, hori izan zen udan eginiko lanetan erabili genuen eskanerra eta, horrez gain, laborategian edukitzeak eta zuzenean berarekin lanean aritzeak beste gertutasun eta ezagutza maila bat ahalbidetzen zigun. Merkatuan hainbat eta hainbat eskaner aurki baditzakegu ere, kontuan izan behar dugu gure sistemak robot bati egokitua egoteko prest egon behar duela eta hori dela, sistema ugari baztertu behar izan ditugu, ez baitute malgutasun hori eskaintzen.

[1] http://pointclouds.org/

2.2.1 DAVID 3D SCANNER

DAVID 3D SCANNER[2], gaur egun, HP-k erosi duen ikusmen artifizialeko enpresa bat da. Hainbat produktu kaleratu ditu bere sorreraz geroztik. Bere produktuak ugariak baldin badira ere, ikusmen artifizialean, eta zehazkiago esanda 3D eskanerren artean zeresana eman duen DAVID SLS-3 sistema izan da. Produktu hori 60-500 mm-ko objektuak eskaneatzeko gai da, 360º-tako 3D modelo zehatzak sortuz. Argi egituratuan oinarritzen den sistema eramangarri eta malgua da, gure beharretara egokitzeko gai dena. DAVID SLS-3 proiektore eta kamera batez osaturik dago (bi kameretara egokitzeko aukerarekin). Bi osagaiak lotzeko, tripode bat erabiltzen du (ikus 2.1 irudia). Kalibrazio-taula bat ere badauka, kalibrazio automatiko bat ahalbidetzen duena, hain zuzen ere. Konfigurazio egoki batekin kameraren bereizmena eskaneatu beharreko objektuaren %0.5-ekoa izatera iritsi daiteke (0.05 mm-koa gehienez ere) eta, denborari dagokionez, ezarpen guztiak modu optimoan baldin baditugu, 2 segundokoa izatera murriztu dezakegu (hala ere 10 segundotik gorakoa ere izan daiteke konfigurazio eta gure hardwarearen ezaugarrien arabera). Prozesu guztia modu sistematiko eta erraz batean burutu ahal izateko lizentzia baten beharra programa propio bat ekartzen du berarekin. Horrez gain, lorturiko emaitzaren manipulaziorako zenbait tresna intuitibo eskaintzen dizkigu.

Xehetasunez hitz egiten hasi behar baldin badugu, aipatu beharra dago Windows plataformarako soilik dagoela eskuragarri. Hau puntu negatibotzat hartu beharra daukagu, izan ere badakigu industria munduan Linux sistema eragilea ere oso erabilia dela eta bateragarritasun hori ez izateak arazoak ekar dakizkiguke, bai hainbat prozesu sinkronizatu eta lotzeko orduan eta baita gure aplikazioak optimizatzeko orduan ere. Hori alde batera utzita, eta produktuaren errendimendua pausoz pauso aztertzeari ekiten badiogu, bere errendimendua kontuan hartzeko modukoa dela esan genezake:

- **Kalibrazioa:** 90º-ko angeluarekin perpendikularki elkarturiko bi panel erabiltzen ditu sistema honek prozesu hau burutu ahal izateko. Panelean, sistemarentzat irakurgarri diren hainbat irudi agertzen dira eta horien neurriak ezagunak dira sistemarentzat. DAVID SLS-3-ren abantailetako bat sistemaren eskalatzeko ahalmena da, izan ere tamaina ezberdinetako irudiak eskaintzen dizkigu eta erabiltzaileak eskaneatu beharreko objektuari ondoen egokitzen zaiona erabiltzeko aukera izango du sistema kalibratzeko.

- **Eskaneatze-prozesua:** Fase honetan hainbat aukera eskaintzen zaizkigu gure eskaneatze-

[2] www.david-3d.com/en/support/david4/introduction

prozesuaren xehetasunak aldatzeko. Testura aldatzeko eta lortutako emaitza iragazteko hainbat tresna eskaintzen dizkigu. Horrez gain, kamerara iristen den argi kopurua errazago kontrola dezagun hainbat laguntza grafiko ere ematen ditu.

- **Eskaneatu ondorengo fasea:** Behin eskaneatzea eginda, hainbat tresna interesgarri eskaintzen dizkigu programa honek. Hala nola, lerrokatzerako aukerak, hainbat eskaneo bakar batean elkartzekoa, zenbait emaitzaren elkarren artean alderatzeko aukera eta baita, distantziak neurtzeko hainbat tresna ere. Azkenik, lorturiko berreraikitzea formatu ezberdinetan esportzeko gaitasuna ere eskaintzen du.

2.1 Irudia: David sistemak erabiltzen duen hardwarea

2.2.2 Microsoft Kinect

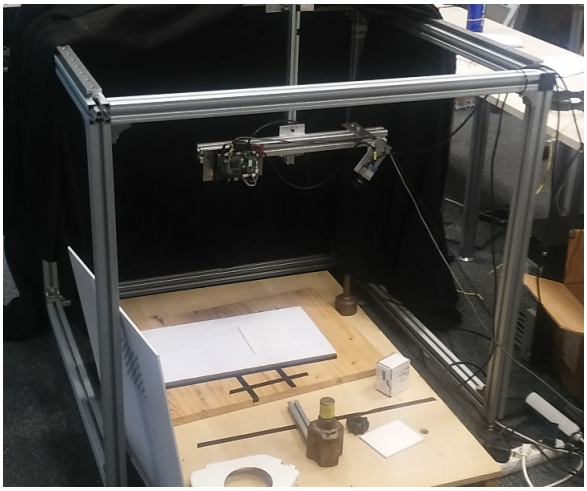
Microsoft-ek jokoak kontrolatzeko modu berritzaile bat bilatuz kaleratu zuen gailua da[3], egun oso ezaguna. Sakonera neurtzeko sentsoreak, RGB kamera, mikrofono array bat eta infragorriak emititu eta hartzen dituen produktu hau, giza eskeletoa atzeman eta plano batean posizionatzeko gai da. Hala ere, jokoetarako erabiltzeaz gain, jada sarean dauden hainbat eta hainbat programekin gure proiektu propioetara egokitzeko aukera eskaintzen du. Berreraikitzeak burutzeko elementu guztiak bere barnean dituenez, mugikortasun handia eskaintzen digu eta, horrez gain, jokoak programatzeko prestaturik dagoenez, hainbat tresna interesgarri ditu lorturiko irudiekin lan egiteko ere. Hala ere, sortzen ari garen sistema objektuak atzeman eta hauen kalitate-kontrola burutzeko erabiliko dugunez, bere bereizmena eta zehaztasun ezak arazoak emango lizkiguke. Erabiltzen duen kamera

[3] https://developer.microsoft.com/es-es/windows/kinect

640x480-koa da eta, softwareari dagokionez, asko izan dira 3D kamera honekin lan egiteko sarean sorturiko programak. Hori ikusita, Microsoft berak ere, bere aplikazio propioa kaleratu du horretarako, *3D Scan* [4].

2.2.3 Gure sistema, hardwarea

Ikusmen artifizialeko sistema bat garatzeko, funtsezko elementuetako bat ordenagailuaren izan behar duen ikusteko gaitasuna da, hau da, argazkiak ateratzeko gaitasuna. Horrez gain, gure kasuan behintzat argi egituratuan oinarrituriko sistema bat izango da, eta beraz proiektore baten beharra ere izango du. Esan bezala, proiektuaren helburua ikusmen sistema baten software aukeraketa eta konfigurazioa burutzea zen, eta jarraian sortuko dugun sistemaren hardwarea zein den erakutsiko dugu (ikus 2.2 irudia).



2.2 Irudia: *Gure sistema, kanpotik ikusita*

Esan bezala, kamera eta proiektore batez osaturik dago eta horiek barra metaliko batzuen bitartez eginiko egitura baten bitartez loturik daude. Barra metaliko horien bitartez kamera eta proiektorearen altuera egokitzeko aukera dago, modu horretan sistema tamaina ezberdineko piezak eskaneatzeko egokituz. Horrez gain, atzealdean, gure nahietara egokitu dezakegun tela beltz baten ikus dezakegu. Telaren bitartez sistemara kanpotik sartuko den argia erregulatu dezakegu.

[4] https://developer.microsoft.com/en-us/windows/hardware/3d-print/scanning-with-kinect

IDS uEye CP USB 3

IDS konpiniak sorturiko kamera industrial hau [5] (ikus 2.3 irudia) egokia da 3D ikusmen sistemak sortzeko, bai bere tamainagatik, eta baita eskaintzen dituen zehaztapenengatik ere.

2.3 Irudia: *IDS kamera industriala*

Kamera honek hainbat abantaila eskaintzen dizkigu gure sistema eraikitzeko orduan, eta horri esker, gure proiektuan bere aukera guztiak erabili ez baditugu ere, sisteman integratzeko eta gure bere errendimendua optimizatzeko aukera egokiak eskaintzen dizkigu:

- Software propio bat dauka bere ezarpenak egokitu eta gure beharretara egokitzeko. Probak egiteko ere oso ondo etorriko zaigu.
- API bat eskaintzen du gure aplikazioekin integratu dezakeguna.
- USB 3.0 erabiltzeko aukera du, fotograma-tasa asko igotzen duena.
- Memoria propio bat dauka, bere ezarpenen multzo bat gordetzeko gaitasunarekin.
- *Trigger* konexioa dauka, hau da, kamerarekin zuzenean kable baten bitartez konektatzeko gaitasuna, sinkronizazioa modu nabarmenean optimizatuz.

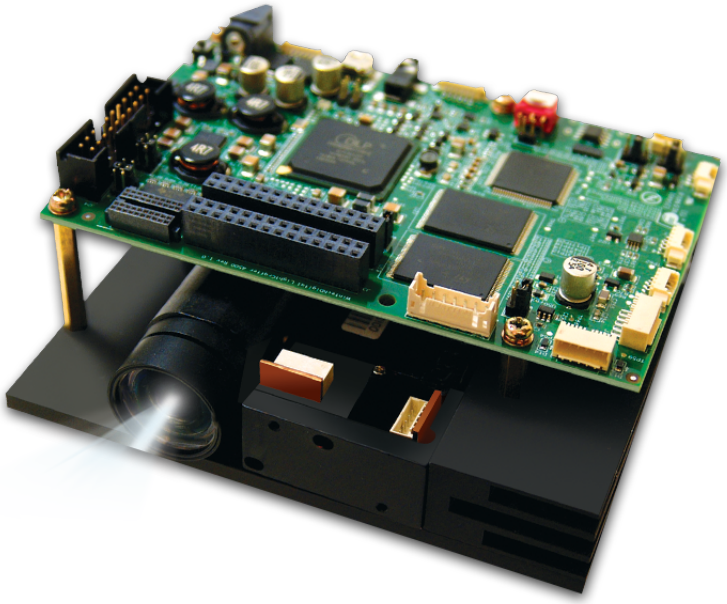
Light Crafter 4500

Texas Instruments konpainiak eskaintzen digun DLP Light Crafter 4500 irudia) proiektorea[6] (ikus 2.4 izango da gure sistema erabiliko duguna. Produktu honek, bere tamaina txikia izanda ere, distira eta bereizmen altuak eskaintzen dizkigu.

Proiektore honen ezaugarrietako asko oso egokiak dira sortzera goazen sistemarako:

[5] https://en.ids-imaging.com/store/products/cameras/usb-3-0-cameras/ueye-cp/show/all.html

[6] http://www.ti.com/tool/DLPLCR4500EVM



2.4 Irudia: *Light Crafter 4500 proiektorea*

- Programa propio bat eskaintzen du bere ezarpenak aldatu eta gure aplikazioaren beharretara egokitzeko.
- API bat eskaintzen du, tresna grafikoaren egin ditzazkegun aldaketak zuzenean gure programatik burutu ahal izateko.
- Patroi-sekuentziak sortu, bere memoria propioan gorde eta azkar bistaratzeko aukera ematen du, bistaratze-frekuentzia altuak lortuz.
- Bi *trigger* portu eskaintzen ditu, kamerarekin konektatu eta bien arteko sinkronizazioa optimizatzeko.
- Mini HDMI portu bat du eskaintzen du, erabiliko den ordenagailuarekin konektatzeko egokia.
- Proiektorearen tamaina trinkoa: 122mm x 115mm x 48mm

3. KAPITULUA

Analisia eta software aukeraketa

Jada orain arte azaldutakoarekin garbi uzten saiatu bagara ere, berriro ere azpimarratu nahi dugu gure proiektua ikusmen artifizialari buruzkoa dela. Eremu zientifiko hori mundu errealeko irudiak atzeman, prozesatu, analizatu eta ulertzen ahalegintzen da, ordenagailu bidez tratatzeko moduko informazioa sortuz. Hala ere, oso eremu zabala da eta hau zertxobait murriztu nahian, gure proiektua 3D ikusmenean zentratuko dela esango dugu. 3D ikusmenak ordenagailu batean giza ikusmena emulatzean datza, hau da, ordenagailuari aurrean daukan eszena edo objektuaren hiru dimentsioko modelo bat sortzeko gaitasuna ematea izango da gure helburua. Horretarako, bi dimentsiotako irudi bat edo gehiagotatik abiatuko da ikusten duenaren sakonera lortuz. Prozesu hori burutu ahal izateko, hainbat modu ezagutzen dira. Hona hemen egun erabiltzen diren teknika ezagunenak:

- Zuzeneko 3D berreraikitzea

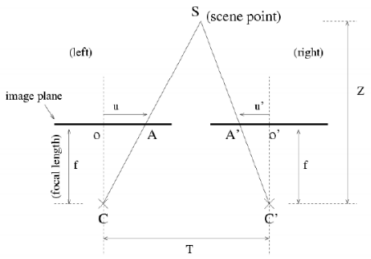
  Teknika honek 2D irudi batetik zuzenean 3D irudi bat berreraikitzea suposatzen du. Esan bezala, arazo ugari ekartzen ditu prozedura honek, objektu berdina oso modu ezberdinetara agertu baitaiteke anbiguotasun maila oso altua izanik. Hala nola, hori burutzeko honakoa izan ohi da prozesu ohikoena. Lehenik eta behin, berreraiki nahi dugun irudi bakoitzarentzat, haren antza duen 3D modelo bat beharko dugu. Modelo horiek erabiliz, sakonera-mapa ezberdinak sortuko ditugu eta horietan sistemak gainberitatutako ikasketa-prozesu bat burutuko du mapa hauetariko bakoitza dagokion 3D modeloarekin lotu dezan. Teorian, irudi berri bat jasotzen duen heinean, dagokion 3D modeloarekin lotzeko ahalmena izan beharko luke. Hala ere, prozesua errazteko hainbat teknika lagungarri erabiltzen dira:

  - Ertz eta eskualde ezberdinen detektatzea
  - Ezaugarrien erauzketa
  - Behin objektua detektatuta, objektua espazioan kokatzeko gaitasuna
  - Interpolazio-metodoak, informazio falta osatzeko

- Sistema estereoskopikoak (Munro and Gerdelan, 2006)

  3D ikusmeneko sistemak gizakiok daukagun ikusmen-sistema birsortzen saiatzen dira, horretarako eszena berberaren bi dimentsiotako bi ikuspegi edo gehiago erabiliz, eszenaren sakonerari buruzko informazioa lortuz. Hainbat eta hainbat aplikaziotarako erabiltzen dira hiru dimentsiotako ikusmena bermatzen duten sistemak, eta kasu gehienetan, bi kameren bitartez burutzen da.

  Sistemak, bi kameren bitartez, bi irudi lortzen ditu eta triangulazioaren bitartez, aurrean daukan eszenaren sakonera kalkulatzeko gai da. Oinarrizko triangulazioak distantzia jakin batera dauden eta toki berberera begiratzen dauden bi puntu erabiltzen ditu. Parametro horietatik abiatuz, gai da eszenako puntura dagoen distantzia kalkulatzeko. Ideia hauxe erabiltzen da bi irudietan agertzen den sakonera kalkulatzeko.



3.1 Irudia: *Estereo ikusmen-sistemen eskema*

3.1 irudian ikus dezakezuenez, bi kamerek (C,C') puntu berdina ukitzen dute (S). Kamera bakoitzean irudietan puntu horrek duen posizioa A et A'-k ematen dute. Bi kamerak elkarrengandik T distantziara aurkitzen direnean, A eta A' puntuak irudiaren normaletik U eta U' distantziara egongo dira, hurrenez hurren. Parametro guztiak jakinik, kameretatik eszenako puntura dagoen distantzia jakin dezakegu honako formularekin bitartez:

$$Z = f\frac{T}{U - U'}$$

Ikus dezakegunez, formula konplikatua ez bada ere, U eta U'-ren balioak jakin beharrak dauzkagu sakonera kalkulatu ahal izateko eta, horretarako, irudi-analisirako zenbait teknika erabiltzen da. Izan ere, parekotasun hau bilatzea konplikatua izan daiteke kolore edo intentsitate homogeneoa duten eremuetan; posible da lorturiko emaitzaren azpimultzo batek fidagarritasunik ez ematea. Teknika honen abantailatzat eszena prestakuntza eza nabarmenduko genuke, ez baitu eszenaren argiztapen bereziaren moduko inongo baldintza berezik behar bere helburua bete ahal izateko.

- Hegaldi-denboran oinarrituriko sistemak

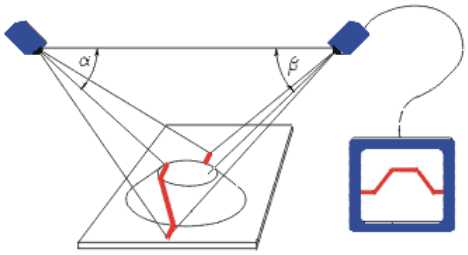
  Hiru dimentsiotako irudia lortzeko sistema hauek hegaldi-denboran oinarrituriko teknikak erabiltzen dituzte radar baten antzeko funtzionamendua dutelarik. Eszena osoan argi-izpi batek duen portaera atzematen da, emititzen denetik kamerara itzultzen den arte igarotzen den denbora kalkulatuz. Eguneratze-abiadura altuak eskaintzen baldin badituzte ere zarata ugari duten irudiak lortzen dituzte eta hori dela eta, ez dira egokiak zehaztasuna eskatzen duten aplikazioetarako.

- Argi egituratuan oinarrituriko sistemak

  Hiru dimentsiotan eskaneatzeko erabiltzen diren sistema hauek, kamera eta proiektore batez osaturik daude. Kamera eta proiektorea software edo hardware bidez sinkronizaturik egongo dira eta proiektoreak, jada sistemarentzat ezaguna izango den patroi bat proiektatzen duen bakoitzean, kamera edo kamera multzoak proiektaturikoa atzemango dute.

  Proiektoretik hiru dimentsiotako eszena baten gainean patroi bat proiektatzen dugun bakoitzean, proiektorea bera ez den beste edozein perspektibatatik begiratuta distortsio bat atzemango dugu. Deformazio hori, jada jakina den ikuspuntu batetik aztertzen baldin badugu, eszenaren berreraikitzen zehatz bat burutzeko gai izango ginateke (ikus 3.2 irudia).

  Sistema horiek patroi bakarra edo patroi bat baino gehiago proiektatuko dute atzeman beharreko eszenaren gainean eta bertako pixel multzo bakoitza kode baten bidez identifikatuko dute. Funtsean, identifikatu beharreko pixel bakoitzak bere kode propioa izango du proiektatu beharreko unean. Hau da, guk kameraren bidez irudia jasotzen dugunean, kodetze baten beharra izango dugu bertan proiektoreko pixela identifikatu eta patroian duen posizioarekin mapatzeko. Hori dela eta, identifikatu beharreko puntuen kopurua handitzen den heinean, orduan eta zailagoa izango da puntu horiek kodetzeko gai den estrategia bat bilatzea.



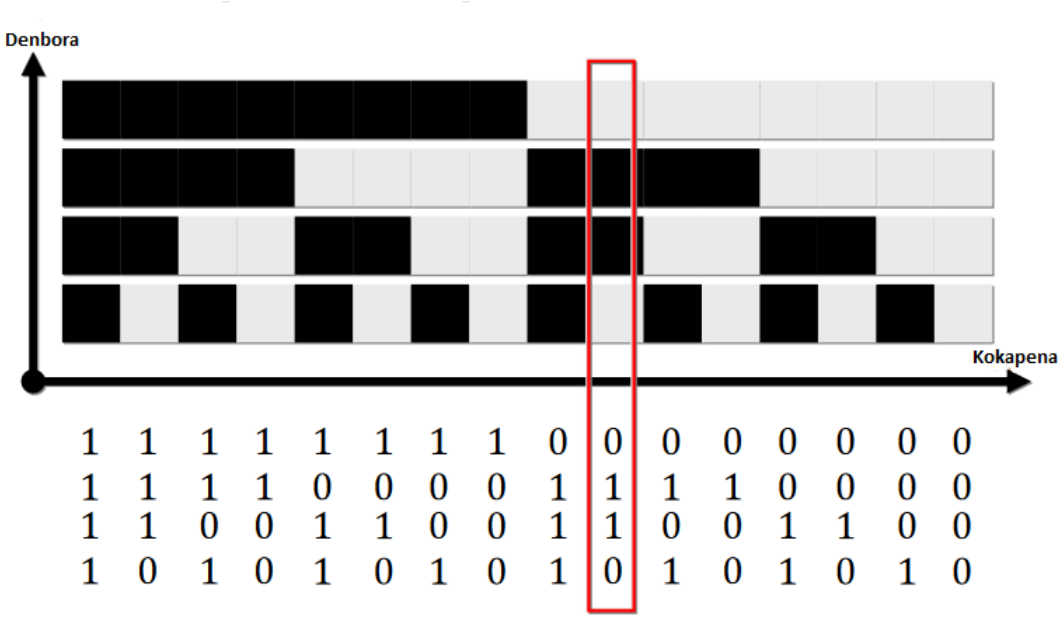
3.2 Irudia: Argi egituratuan oinarrituriko sistema baten eskema

Behin mapatzea burutu ondoren, sistemak kamerako pixel bakoitza proiektoreak proiektatu duen patroian duen koordenatuekin lotuko du; hau izango da funtsean kodetze-sistema guztiek izango duten helburua, eta behin hori lorturik, triangulazioa sartuko da jokoan, mapatzetik abiatu eta pixel bakoitzaren 3D koordenatuak lortuz kameraren espazioan.

Hau ulertuta, sistema horien gakoa proiektatuko diren patroietan dagoela ikus dezakegu, izan ere, mapatze hori teknika ezberdinak erabiliz lor dezakegu. Patroi ezberdinak proiektatzeko ahalmenak oso polifazetikoak bihurtzen ditu argi egituratuan oinarritutako sistemak, izan ere, patroi hauek aldatuz zehaztasunean, egonkortasunean edo azkartasunean zentratu gaitezke.

Hori dela eta, proiektatuko diren patroien kodifikazioaren inguruan ikerketa ugari burutu dira kodetze azkarragoak bilatu nahian, patroi kopurua murriztu nahian eta identifikazioan eman daitezkeen erroreak minimizatzeko asmotan. Teknika horien sailkapenak egiteko modu ugari baldin badago ere, hiru multzo nagusi bereiziko ditugu.

- Denbora-multiplexazioan oinarrituriko teknikak: Teknika hauek patroi bat baina gehiago proiektatzen dute denboran zehar eta, hori dela eta, patroi bakoitzaren kodetzea modu sinplean egiteko aukera dago. Identifikazioa, beraz, asko errazten da eta erraztasuna, fidagarritasuna eta zehaztasuna eskaintzen dizkigu. Hala ere, berreraikitzeak burutzeko behar duen denbora beste teknikena baina handiagoa da eta horrek muga batzuk ezartzen dizkigu. 3.3 irudian ikus dezakezuen bezala, posizio bakoitza kodeturik gelditzen da.
- Bizilagunetan oinarrituriko kodetzeak (Salvi et al., 2004): Teknika hauek kodifikazio eskema osoa patroi bakar batean kontzentratzen ahalegintzen dira.



3.3 Irudia: Kodeketa bitarra patroi bidez, lerro bakoitza proiekzio bat da

Patroiko puntu bakoitzaren kodetzea zehazteko bere inguruko puntuen informazioa erabiltzen da. Horrek patroi kopurua murrizten baldin badu ere, arazoak izan ditzakegu deskodetze fasean, puntu batzuen ingurunea identifikatzeko gai ez garenez erroreak egon daitezkeelako. Patroien artean teknika ezberdinak aurki ditzakegu, kodifikazio ez-formalean oinarrituak, ausazko De Bruijn sekuentzietan ala M-array sekuentzietan.

– Kodetze zuzena: Kodifikazio mota honetan pixel bakoitzarentzat kodetze zuzen bat egiten da; hau bere kolore (ikus 3.4 irudia) edo gris mailaren berdina izango da. Horretarako, bai emisio eta baita argiaren atzemateak ere guztiz sinkronizaturik egon behar dute eta prozesamenduak ere oso optimizatua egon behar du. Sistema gutxi batzuk modu honetan lan egiten badute ere, guk dakigula inplementazio hauek ez dira publikoak.

Printzipioz behintzat, eraikitzera goazen sistemak zehaztasun handia eskatzen duen aplikazioetara bideratuko dugu, beraz, kodetzea sailkapen honetako lehen multzoan finkatuko gara, denbora-multiplexazioan oinarritutako tekniketan hain zuzen ere. Denbora gehiago behar badute ere, teknika hauek dira zehaztasun eta fidagarritasun maila handienak eskaintzen dituztenak, baita azken urteetan ikertuenak izan direnak ere.

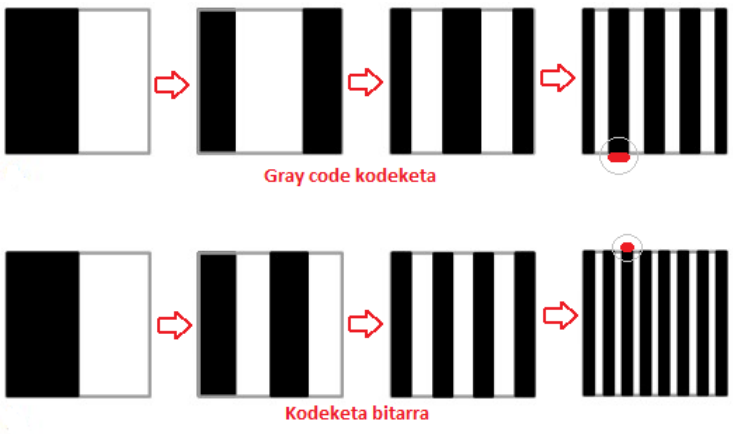
Denbora multiplexazioan oinarrituriko tekniketan, pixel bakoitzaren kodetzea pixe-

3.4 Irudia: Kodeketa zuzena: puntu bakoitzaren kodea bere kolorea soilik da

lak proiektaturiko patroietan zehar hartzen duen argiztapen sekuentziak zehazten du. Zenbait abantaila eskaintzen dizkigu horrek, izan ere, patroi bat baino gehiago proiektatzeak asko sinplifikatzen du gure kodetzea eta, horrez gain, proiekzioa patroi sinpleenetik konplexuenera burutzen denez, pixel bakoitzaren zehaztasuna iterazioz iterazio hobetzen joaten da.

Multzo honetan ere, erabiltzen diren tekniken artean, aldakuntza ezberdinak aurki ditzakegu baina gu bi ezagunenetan zentratuko gara:

– Kode bitarrak: Bi argiztapen maila soilik erabiltzen dira patroiak osatzeko eta hauek 0 eta 1 balioen bidez kodetzen dira. Hau horrela izanik, patroiko pixel bakoitza 0 eta 1 balioen sekuentzia baten bidez kodetua dago. Hala ere, kode bitar hauen artean *gray code* metodoa azpimarratu dezakegu, patroi kopuru berdina erabilita ere zabalera handiagoa erabiltzen baitu bere marretan kameraren atzematea erraztuz (ikus 3.5 irudia). Horrez gain, *gray code* kodeketan dagokionez patroi batetik bestera bit bakarra aldatzen da.



3.5 Irudia: Ohiko kode bitarra eta *gray code*-en alderaketa

Teknika hauen azpimultzotzat har ditzakegu n-ary kodeak. Kodetze mota hau proiektatu beharreko patroi kopurua murrizten ahalegintzen da argiztapenaren intentsitatean maila ezberdinak erabiliz eta, modu horretan, patroi bakoitzean informazio gehiago kodetuz.

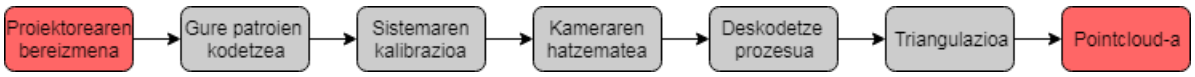
– *Phase shifting* metodoak: Aurrez aipaturiko kodetzeetan, pixel bakoitzaren kodetzea besteekiko independentea da eta bizilagunak ez dira kontuan hartu behar kodetzea burutu ahal izateko. Hala ere, kodetze mota hau zehatza baldin bada ere, zenbait muga ezartzen dizkigu bereizmenari dagokionen. Bereizmena hobetu nahian sortzen dira *phase shifting* metodoak. Teknika horiek intentsitate periodikoa duen patroi bat hainbat aldiz proiektatzen dute, intentsitatea proiekzio bakoitzean ezberdina delarik. Hala ere, patroiek duten izaera periodikoak anbiguotasuna sortzen du kameraren irudietan seinalearen periodoa determinatzeko unean. Hori saihesteko, ohikoa da *phase shifting* metodoa aipatu berri dugun teknika bitarrekin konbinatzea anbiguotasuna ezabatu eta bereizmen egokia duen kodifikazio sendo bat lortzeko. 3.6 irudian ikus ditzakezue soilik 3 patroiez osaturiko kodetze batean erabilitako irudiak.

3.6 Irudia: Hiru patroiez osaturiko *phase shifting* kodetzea

Kodetze-fasea garrantzitsua baldin bada ere, sistema hauek funtsezkoa den kalibrazio-fase bat eskatzen dute. Kalibrazioak bai proiektore bai kameraren lenteen distortsioak konpentsatu behar ditu, baita kamera eta proiektoreak espazioan duten posizio eta orientazioa zehaztu ere.

Kalibrazio egokia, kodetze-sistema eraginkor bat eta horrekin batera deskodetze egoki bat duen sistema bat bilatzen ahaleginduko gara gure sistema osatu eta berreraikitze zehatzak burutu ahal izateko.

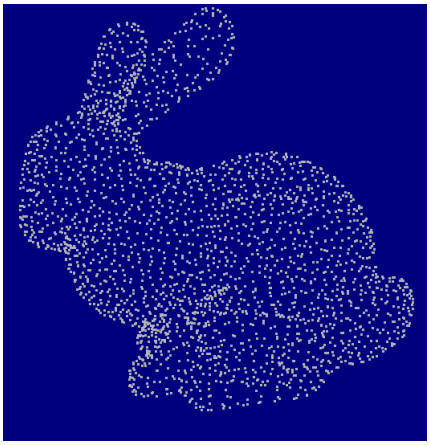
Ideia hauek garbiago azaltzeko asmoz jarraian argi egituratuan oinarriturik dauden sistema gehienek izan ohi dituzten pausu guztiak finkaturik daude, edo garbiago esanda, gure berreraikitzean lortuko dugun puntu-hodeiean eragina duten fase guztiak.



3.7 Irudia: Argi egituratuaren eskema

Gure sistemak lortuko duen emaitzak errorea minimoa izatea nahi baldin badugu, 3.7 irudian ikus ditzakezuen fase bakoitzean burutu beharko dugun prozedurak egokia izan beharko du eta aurreko fasean lorturiko bereizmen eta zehaztasuna ahalik eta gehien errespetatzen saiatu beharko du. Behin fase guztiak gainditua, sistema puntu-hodei (point cloud ingelesez) itzuliko digu.

Puntu-hodeia koordinatu sistema batean aurkitzen diren puntuen multzo bat da (ikus 3.8 irudia). Gehienetan, eta baita gure proiektuan ere, puntu hauek hiru dimentsioko koordinatu sistema batean finkaturik egoten dira eta objektu bat errepresentatzen dute. Formatu ezberdin ugari erabil daitezke hauek tratatzeko eta hauetariko batzuk puntu bakoitzaren kolorea (RGB balioa) edo normala izan dezakete.



3.8 Irudia: Untxi bat irudikatzen duen puntu-hodei sinple bat

Puntu-hodei hauetatik abiatuz zenbait helburutarako erabilgarriago diren triangelu-sareak eraiki ditzakegu. Hau da, lehenik eta behin puntu-hodeiko puntu bakoitzaren normalak kalkulatu ditzakegu eta ondoren, hemendik abiatuz, gertu dauden puntuak ertzen bitartez lotu ditzakegu triangelu sare bat osatuz.

Puntu-hodeiak hainbat helburutarako erabil ditzakegu, gure kasuan kalitate analisi bat burutuko dugu baina hori modu zehatzagoan azalduta aurki dezakezue *Konparaketa* atalean.

3.1 Gure sistemaren softwarearen aukeraketa

Zehaztu berri ditugu zein diren gure 3D ikusmen-sistemak edukitzea gustatuko litzaizkigukeen ezaugarriak. Batez ere, zehaztasun handiko lanetarako erabilia izango denez, denborari baina garrantzi handiagoa emango diogu lortutako berreraikitzeen kalitate eta zehaztasunari. Hori dela, esan dugun bezala, denbora multiplexazioan oinarritutako teknikak bilatuko ditugu. Horien barnean, gaur egun argi egituraren munduan erabilienak diren teknika bitar eta *phase shifting* metodoak bilatuko ditugu eta gure lanerako zer nolako emaitzak ematen dituzten ikusiko dugu.

Gure bilaketa egiterako orduan, software komertzialak alde batera utzi ditugu eta software libreetan zentratu gara. Sustatzaile independenteengandik sortuak egon ohi dira eta, gehienetan, hardware jakin batentzat soilik prestatuak egoten dira, hau da, egilearen hardwarerako. Hori dela eta, hainbat konfiguratze eta egokitze-prozesu burutu behar izaten dira gure sisteman funtzionatu dezaten. Horrek hainbat atzerapen eta arazo ekar baditzakegu ere, software horiek erabiltzeko ez da lizentziarik behar izaten eta, gainera, hobekuntzak burutzeko atea irekia uzten du. Programa aldatu eta berriz konpilatzeko aukera eskaintzen digu, oso erabilgarri gerta dakigukeena horretarako beharra sortzen bazaigu.

Gure bilaketa ekin bezain laster konturatu ginen aurki genitzakeen softwareak ez zirela asko izango, izan ere, gaur egun 3D ikusmena punta-puntan baldin badago ere aukera zabala sistema komertzialetan baino ez dago. Software libreari zegokionean, oso programa gutxi aurkitu genituen, zehazki hiru: SLStudio, Texas Instruments 3D Scanner, eta SLS Underworld 3D. Jarraian bakoitzaren ezaugarrien deskribapen labur bat eta gure hautapena ikus dezakezue.

3.1.1 SLStudio

Dinamarkako Unibertsitate Teknikoko eta Kopenageko Unibertsitate Ospitaleko hainbat ikerlarik sorturiko kode irekiko framework-a da honakoa (Wilm et al., 2014). Bere helburua, denbora errealean 3D eszenen atzematea burutzea da medikuntzako aplikazioetan erabili ahal izateko, bai mugimendua detektatu, lokalizatu eta hau zuzentzeko. Softwareak duen izaera modular eta hedagarriak ateak irekiak uzten ditu softwarearen kodean algoritmo berriak integratzeko; bai kodetze/deskodetzerako, triangulaziorako eta hardware ezberdinen integraziorako.

Framework hau C++ lengoaian idatzirik dago eta bere konputazio gehienetarako OpenCV

eta Point Cloud Library (PCL) liburutegiak erabiltzen ditu, erabiltzailearen interfaze grafikorako zein nukleo anitzeko prozesamendurako berriz Qt framework-a erabiltzen du.

Hardwareari dagokionean berriz, erabilitakoa proiektorea guk erabilitakoarekin bat datorrela ikusi dugu, eta erabilitako kamerari dagokionean berriz, bateragarritasun osoa ez baldin badu ere, IDS kameraren API-aren hainbat aukera barneraturik dituela ikusi dugu.

Esan bezala, patroiak kodetzeko erabiltzen dituen teknikei dagokionean, ohiko teknika binario eta *phase shifting* teknikez gain beste hainbat teknika eskaintzen dizkigu, hala ere kontuan izanik gehienak oraindik ikerketa fasean daudela. Hori dela eta, nahiz eta guri interesatzen zitzaiguna software honek eskaintzen zigun denbora errealeko atzematea ez izan, patroi ezberdinak probatzea interesgarria izan zitekeela iruditu zitzaigun.

Kalibrazioa burutzeko, ikerlari hauek proposatzen duten teknika bat erabiltzen du proiektu honek; horretarako aurrez inprimatutako kalibrazio taula bat erabiltzen du eta phase shifting-ean oinarrituriko patroi batzuk proiektatzen ditu haren gainean. Kalibrazioa optimotik gertu egon dadin 30-40 posizio ezberdin eskatzen ditu orientazio, posizio eta distantzia ezberdinetan baina gutxiagorekin ere emaitza onak lor ditzake. (Wilm et al., 2014a)

Horrez gain, abiadura handiko eskaneatzea ere eskaintzen zigunez, horretarako beharrezkoa den hardware *trigger* konfigurazioa ere eskaintzen zigun software honek. Hau kamera eta proiektorearen artean ezartzen den konexio bat da eta modu honetan patroi proiekzio/-kamera atzemate sinkronizazioa asko azkartzen da. Ohikoa den bezala software bidez burutzen denean, seinalea ordenagailuaren bitartez bi gailuetara bidaltzen da eta itxaron denborak kontuan hartzekoak izan ohi dira. Hardware bidez burutzen baldin badugu berriz, proiektoreak patroi bat proiektatzen duen bakoitzean, zuenean seinale bat bidaliko du kamerara sinkronizazio prozesua asko azkartuz.

3.1.2 3D UNDERWORLD SLS

Immersive & Creative Technologies Lab laborategiko ikerlariek, arkeologoek ur azpian burutu beharreko 3D eskaneatze prozesuak azartzeko asmoarekin sortu zuten jarraian aztertzera goazen softwarea [1]. Aurreko bi softwareen moduan C++ lengoaian idatzirik zegoen, Visual Studioko proiektu egitura erabiliz hain zuzen ere.

Lehenik eta behin azpimarratuko duguna, gure aukeraketan ezaugarri hau garrantzitsua baita, software hau bi kamera eta proiektore baten bitartez osaturiko sistema batentzat

[1] https://arxiv.org/pdf/1406.6595.pdf

prestaturik dagoela da. Hau da, behin proiektaturiko patroien irudiak kameraren bitartez atzeman ondoren, puntuak triangulazio bitartez lortzeko, ikuspuntu ezberdinetatik lorturiko bi kamera erabiltzen ditu. Bi kamera izateaz gain, hauek bi *Canon* kamera dira eta softwarea hauen SDK-rekin erabiltzeko prestaturik dago.

Softwareak bere kodeketa burutzeko erabiltzen dituen patroiei dagokienez, denbora multiplexazioan ohikoen bi aldaerak erabiltzen ditu: alde batetik teknika bitarra, *gray code* metodoa erabiltzen duelarik, eta bestetik *phase shifting* metodoa, puntu-hodeiaren puntu kopurua handitu ohi duena.

Sistemaren kalibrazioari dagokionean, ohikoa den moduan lehendik inprimatua izan beharko dugun xake taula moduko egitura bat inprimatu behar da eta taula lau batean itsatsirik kalibrazioa egiteko erabili. 10-20 irudi beharko ditu kalibrazio egoki bat lortzeko eta distantzia orientazio eta posizio ezberdinetan lortu beharko ditugu irudi hauek.

Software honek eskaintzen duen puntu interesgarri bat, behin berreraikitzea egitean softwareak berak burutzen duen formatu aldaketa da. puntu-hodeia emateaz gain, triangelusare bat sortuko du bertatik abiatuz, hau da Puntu-hodeiko puntuak ertz batzuen bitartez konektatuko ditu hirukiak osatuz. Modu horretan erabilgarriagoa den datu-egitura bat lortuko dugu. Gainera, prozesu hau programak berak burutzen duenez, kanpo programa batek baina informazio gehiago dauka, hodeiko puntu bakoitzari dagokion proiektoreko pixela zein den badakielako, eta arrazoi horrengatik puntuen arteko loturak modu egokiagoan burutu ditzake. Horrez gain, dokumentazioan erakusten duenez kolorea mantentzeko gai da irudiko rgb balioak erabiliz eta hau interesgarria izan daiteke objektuari buruzko informazio hori interesatzen baldin bazaigu; baina ez da gure kasua.

3.1.3 Texas Instruments 3D Application (TI3D)

Hirugarren aukeratzat hartu genuena Texas Instruments enpresak sorturiko erabilera libreko softwarea da. Industria elektronikoan TI siglekin ezaguna den enpresa hau erdieroaleak ordenagailuetan erabiltzeko teknologia ezberdinak komertzializatzeagatik da, horien artean proiektore industrialak. Gure proiektuan erabili dugun proiektorea enpresa honek eraikia da eta horrek bultzatu gintuen software honi begiratu bat ematera. Izan ere, proiektoreaz gain, 3D berreraikitzeak burutzeko software bat eskaintzen zigun enpresa honek. Hainbat erabilera-gida eta proba-kasu ere eskaintzen zizkigun eta erabilgarri suertatuko zitzaizkigulakoan geunden. Software hau ere, C++ lengoaian idatzirik zegoen eta bere kodea modulu jakin batzuetan banaturik zegoen.

Gu eraikitzera gindoazen moduan kamera eta proiektore baten bitartez egiten zuen lan software honek. Proiektoreari zegokionean bagenekien gurearen berdina zela eta alde horretatik arazorik izango ez genuela. Kamerari zegokionean berriz, Point Grey kamera bat erabiltzeko prestaturik zegoen, baina hala ere beste kamera batzuekin lan egoteko tresnak ere eskaintzen zizkigun erabilgarritasun guztiarekin ez bazen ere.

Erabilitako patroiei zegokienean, denbora multiplexazioan ohikoenak ziren bi kodeketak erabiltzen zituen software honek ere, alde batetik *gray code* metodoa eta *phase shifting* bestetik. Bagenekien biak ere emaitza onak emateko gai zirela, proiektaturiko patroi kopurua kontuan hartzekoa bazen ere.

Kalibrazioari zegokionean, bi fasetan burutzen zuela ikusi genuen aurrez ikusi ditugun bi softwareekin alderatuz. Lehenik eta behin, kameraren kalibrazioa burutzen zuen honen parametroak zehazteko. Behin hori eginda, sistema osoa kalibratzeko, bigarren fase batean proiektorearen parametroak eta kamera eta proiektorearen arteko posizioa zehazten zuen.

Horrez gain, espero bezala eta SLStudioren moduan, kamera eta proiektorearen arteko sinkronizazioa azkarragoaa izan dadin hardware triggering-a eskaintzen digu software honek.

3.1.4 Aukeraketa

Hiru softwareen egitura eta ezaugarriak aztertu ondoren, aukeraketa fase bati ekin genion, gure sisteman instalatuko genituen softwareak aukeratuz. Horretarako, lehenik eta softwarearen bakoitzaren nondik norakoak aztertu genituen.

SLStudio eta 3D Underworld SLS ikertzaile talde batzuk garatuak izan zirela ikusi genuen, eta batez ere, atentzioa deitu zigun SLStudio-ko kideek bere argitalpenetan erakusten zuten jarrera irekia. TI3D-ren kasuan, konpainia batek egindako softwarea zela ikusita ere, kode irekia edukitzeak, eta erabiltzaileen dudak argitzeko foro bat izateak be alde egin zuen.

Erabilitako lengoaiari zegokionean, hiru softwareek C++ lengoaia erabiltzen zuten, eta beraz, hiru kasuetan gure beharrak asetzen zituen, C++ lehendik landua geneukan lengoaia bat baitzen, hala nola, udaran Tecnalian bertan egindako praktiketan.

Erabilitako hardware aztertzerakoan jada, gure aukeraketa zehazten hasi zen, izan ere, itxura batean behintzat SLStudio eta TI3D softwareak gure sistemara, aldaketa askorik

gabe egokitu zitezkeela zirudien. 3D Underworld-n kasuan berriz, 2 kameren bitartez osaturik egoteak, eta gainera hauek Canon SDK-ren bitartez integratuak izateak, bere instalazioa konplikatu, eta gainera bigarren kamera bat bilatu beharko genuke horretarako. Gainera, Tecnalian erabili nuen sistemak (ikus 2.2 irudia), inongo mugimendurik gauzatzen ez bazuen ere, etorkizunean robot edo sistema konplexuagoetan integratua izan zitekeen. Beraz, bigarren kamera horrek mugikortasuna mugatuko zukeen.

Bestalde, esan beharra dago 3D Underworld-en dokumentazioan ikusi dugunez, kalitate handiko berreraikitzeak burutzeko gai dela. Baina, hala ere kontuan eduki beharra daukagu berreraikitze horiek bi Canon kameren bitartez gauzatuak izan direla, eta beraz, lorturiko bereizmen-mailak oso altuak izango direla.

Software hauek erabiltzen dituzten patroien kodifikazio sistemei dagokienez, alor honetan gure atentzioa piztu zuena SLStudio softwarea izan zen. Aztertutako hiru softwareek, ohikoak diren *gray code* eta *phase shifting* metodoak eskaintzen dizkigute, baina aipatu berri dugun softwareak garapen-mailan dauden beste hainbat kodetze-sistema ere eskaintzen dizkigu.

Kalibrazioaren alorrean, SLStudio eta Underworld softwareak fase bakar batean kalibratzeko gai zirela ikusi genuen, 20-30 irudi erabiliz. Gainera, SLStudioren kasuan, bere garatzaileek sorturiko algoritmo interesgarri bat erabiltzen zuen sistema kalibratzeko. TI3D-ri dagokionez berriz, bi fasetan banaturiko kalibrazioa proposatzen zuen: lehenik kamera soilik kalibratu, eta behin hori eginda sistema osoa kalibratze zuen.

SLStudio eta TI3D-ren kasuan, *trigger* bidez konfiguratzeko aukera eskaintzen zutela kontuan izan genuen, hau da kamera eta proiektorearen arteko hardware bidezko konfigurazioa ahalbidetzen zuen. Bilatzen ari ginen ezaugarri nagusia ez bazen ere, honek fidagarritasuna eta optimizazioa ekarriko zizkion gure sistemari.

3D Underworld softwareak beste biek eskaintzen ez zuten ezaugarri bat eskaintzen. Sistemaren emaitza puntu-hodeia izateaz, triangelu-sare bihurtzeko aukera ematen zigun. Hala ere, konparazioa egiten ari ginen unean ez genion garrantzia gehiegirik eman, ez baitzen bilatzen ari ginen ezaugarrietako bat.

Azaldutako arrazoi eta ezaugarri hauek aztertuz, gure sisteman instalatzeko SLStudio eta TI3D softwareak aukeratu genituen. Arrazoi nagusia hardwarean genuen bateraezintasuna izan zen, eta horrez gain, beste bi softwareak, 3D Underworld-ek eskaintzen ziguna gainditzeko gai izango zirela iruditu zitzaigun. Hala ere, lehen konparazio honetan baztertu izanak ez du esan nahi etorkizunean gure sisteman probatzen saiatuko ez garenik, bere proba kasuetan lorturiko emaitzak kontuan hartzekoak baitira.

4. KAPITULUA

SLStudio

Softwarearen azterketarekin amaitu ondoren, hau izan zen gure 3D sisteman martxan jarri nahi izan genuen lehen softwarea. Honako hau, konfigurazio egokiarekin denbora errealean 3D berreraikitzeko gai izango litzatekeen software bat da. Bere izaera irekia eta hedagarriak gure atentzioa piztu zuen, bai eskaintzen zituen algoritmoengatik eta baita guk bertan ekarpenak egiteko zuen erraztasunagatik. Argi egituratuaren erabilera ahalbidetzen duten software libreak ez dira anitzak eta hau proiektu sendoenetakoa iruditu zitzaigun.

Software honek, argi egituratuan oinarriturik dauden beste asko bezala, lehenik eta behin, kalibrazio-prozesu bat eskatzen du, bai kamera, bai proiektorearen lenteen distortsioak zehazteko; kamera, proiektore eta eszenaren arteko neurriak kalkulatzeko, eta baita benetako neurriak sartzeko ere, berreraikitze metrikoak egin ahal izateko. Hau egin ondoren honako pausuak bereizten dira sakonera-mapa bat lortzeko unean:

1. Eszenaren kodifikazioa, non proiektoreko koordenatuak patroi-sekuentzia batean kodetuko diren. Metodo gehienetan ardatz bakarra kodetzearekin nahikoa izan ohi da, ondoren informazio horrekin guk beste ardatza zatituz puntuak atera baititzakegu.

2. Patroi horien atzematea kameraren bitartez egingo da, horretarako modu egokian sinkronizatuak egon behar dutelarik.

3. Atzemandako irudi horietan deskodetze-prozesua emango da, kodetze-prozesuaren alderantzizkoa izango dena. Hemen, kameratik jasotako irudiko pixel bakoitzak proiektorearen koordenatuetan izango duen posizioa kalkulatuko da.

4. Behin aurrekoa kalkulatuta, proiektoreko pixel bakoitzak espazioan duen posizioa kalkulatuko dugu triangulazioa erabiliz.

Software hau, C++ lengoaian idatzirik dago eta bere konputazioetarako gehien erabiltzen dituen liburutegiak OpenCV eta PCL dira. Interfaze grafikoa eta nukleo anitzeko prozesamendua ahalbidetzeko berriz Qt framework-a erabiltzen du.

Bere inplementazioa hainbat modulutan banaturik dago eta guztiak C++-eko klase abstraktuak dira.

- *Calibrator*: Hemen zehazten da kalibraziorako erabiliko den interfazea. Bertan, kodeaz gain kamera eta proiektorearen aldagai intrintsekoak eta hauen arteko erlazio geometrikoa kalkulatzeko erabiliko diren hainbat patroi aurki ditzakegu. Kalibrazioa burutzeko xake taula itxura duen plano bat atzeman beharko du eta PSP teknika bat erabiliko du horretarako.

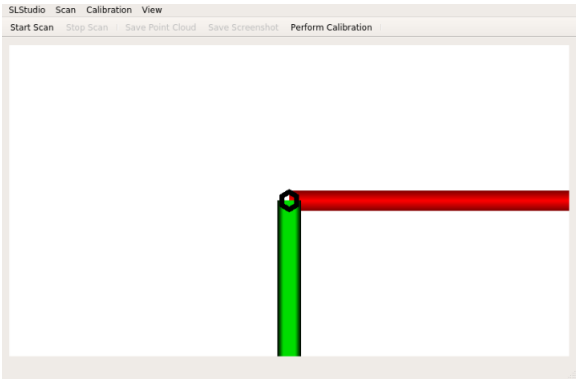
- *Projector*: Patroi proiekziorako interfazea da honakoa. Metodo honek zuzenean atzituko du patroia, memoria bloke bat bezala harturik, inongo konputaziorik burutu gabe. Linuxen gutxienez, X.org fitxategian aldaketak burutu beharko ditugu bigarren pantailan proiektuko diren patroiek lehen pantailan eraginik izan ez dezaten. Proiektoreak aukera gehiago eskaintzen baditu ere, datu transferentzia HDMI kablearen bitartez egingo da. Proiektorearen ezarpenak aldatu nahi baditugu berriz USB kablea erabili beharko da.

- *Camera*: Hemen burutzen da proiektore eta kameraren arteko komunikazioa. Bi modu nagusi beraeizten dira; alde bateik, software *trigger*-a, non irudia atzemateko agindua kodetik bertatik, kamerara eta proiektorera bidaliko den. Kasu honetan, denboran atzerapenak izango ditugu, izan ere, kamerari denbora eman beharko zaio patroia atzeman dezan. Bestetik, hardware *trigger*-a daukagu; kasu honetan, seinalea zuzenean joango da proiektoretik kamerara eta beraz, denboran azkarragoa izateaz gain, sinkronizazioa ia perfektua izango da.

- *Codec*: Hemen inplementaturik daude aurrez azaldu ditugun kodetze eta deskodetze-klaseak. Patroi-estrategia bakoitzarentzat klase hauek ezberdinak izango dira baina guztiek u_p edo v_p mapa bat itzuliko dute, non kamerako pixel bakoitza proiektoreko koordenatu horizontal edo bertikal batekin lotuko den. Patroi ugari aurkezten zaizkigu eta bakoitzaren errendimendua aurrerago aztertuko dugu.

- *Triangulator*: Triangulazioa gauzatzen duen klasea da honakoa, horretarako aurreko klaseak itzultzen duen mapa, kamera eta proiektorearen parametroekin lotuz. Prozesu hau burutu ondoren, PCL liburutegietan erabiltzen den formatuan errepresentatuko du puntu-hodeia.

Programaren exekuzioari dagokionez, hainbat hari ezberdin erabiltzen ditu. Lehenengoa gui-a (ikus 4.1 irudia) eta grafikoentzat, bigarren bat kamera eta proiektorearen deietarako, hirugarrena deskodetze-prozesurako eta azken hari bat, pointcloud-aren eraikuntza burutzeko.



4.1 Irudia: SLStudio softwarearen interfazea

Gure hardwarearekin konfigurazio optimo bat lortu ezkero, eta horrez gain programan bertan aukera egokiak ezartzen baldin baditugu, 300.000 puntuko eszena bat 20 Hz-ko frekuentzian berreraikitzeko gai izan beharko luke.

4.1 Instalazioa

Programa hau instalatzeari ekin genionean, hain kontziente ez baginen ere, software libre batekin ari ginen, ikertzaile edo lan talde batek bere hardware propioan eraikitako programa batekin hain zuzen ere. Hori zela eta, bere hardwareari egokiturik zegoen eta gure sisteman funtziona zezan hainbat aldaketa eta konfigurazio burutu beharko genituen. Hemen azalduko dut prozesu hori, bizi izan nituen ezusteko, arazo eta konponbideak azalduz.

- Dependentziak: Softwarearen nondik norakoak aztertu ondoren, iturburu-kodea deskargatu eta hau instalatzeari ekin genion. SLStudio programak dependentzia ugari

zituen eta, hori zela eta, lehenik eta behin hauek modu egokian instalatzea izan zen. Software honen funtzionamendua, hasiera batean behintzat, hainbat bertsio eta konfigurazio jakinetara murrizturik dago eta horren ildora, hainbat moldaketa eta konpilazioa burutu behar dira gure sisteman funtziona dezan. Aipatutako dependentzia horiek honakoak lirateke:

  - Qt 5.X
  - OpenCV 2.9.X
  - PCL 1.7.X
  - VTK 6.2, QVTK barne
  - Boost
  - Eigen
  - FLANN
  - GLEW
  - Kamerari dagozkion liburutegiak, gure kasuan IDS uEye kamerei dagozkien liburutegiak

- Lehenengo arazoa sistema eragilearekin etorri zitzaigun. Izan ere, softwarearen egileak Ubuntu 16.04 erabili zuen softwarearen garapenerako eta beraz, bertako liburutegiak erabili zituen. Niri Tecnalian utzitako ordenagailuak berriz Ubuntu 14.04 zuen instalatua, eta beste hainbat programekin ere konfiguratua zegoenez, hasiera batean behintzat, instalazioa bertsio honetan burutzea erabaki genuen. Softwarearen egileak, hainbat aldaketa eginez linux-en bertsio honetan instalatzea posible zela azaltzen zuen eta honek ere erabaki hau hartzera bultzatu gintuen.

- Paketeak instalatzeari ekin genionean gehienak arazorik gabe instalatu genituen baina Qt erabiliz konpilaturiko VTK-ren paketea ez zen eskuragarri aurkitzen gure linux bertsiorako. Hau ikusirik, arazoa konpontzeko lehenik Qt instalatu genuen eta ondoren VTK-ko iturburu-kodea deskargatu. Azken hau konpilatzerako orduan, *cmake* fitxategia aldatu behar izan genuen Qt erabiliz konpilatzeko eta horrez gain, Qt-ren liburutegiekin lotura guztiak burutu genituen.

- Konpilazio hori errorerik gabe burutu ostean, SLStudio konpilatzeari ekin genion, liburutegi berriak erabiliz. Hala ere, ez zen gai izan VTK-rekin zituen dependentziak bere kabuz zehazteko eta hori zela eta, berriro ere konpilazio fitxategia aldatuz lotura hauek eskuz burutu genituen.

- Hau egin ondoren ere, konpilazio orduan, VTK-rekin arazoak zeudela ikusi genuen. Orduan konturatu ginen VTK liburutegi dinamikoak erabiliz konpilaturik zegoela eta SLStudiok ez zuela hori onartzen. Horren ildora, berriro konpilatu behar izan genuen VTK liburutegi estatikoak erabiliz eta hori egin ondoren, berriro ekin genion SLStudioren konpilazioari.

- Behin exekutagarria lorturik, softwarea abiatzea lortu genuen baina, oraingoan, ez zen gai kamera eta proiektorera konektatzeko. Arazo hau nondik zetorren aztertzen aritu ondoren, gure ordenagailuak bi txartel grafiko zituela ikusi genuen eta HDMI-a konektaturik zegoen portua aktibatu gabe zegoela. Hori konpontzeko, BIOS-eko ezarpenak aldatu behar izan genituen txartel grafikoa aktibatuz.

- Jada HDMI-a aktibatuta, lortu genuen proeiktoreak funtziona zezan, bere programa propiotik hala egiteko agintzen baldin bagenion eta berdina gertatzen zen kamerarekin. Bi periferiko hauei SLStudioren gui-etik deitzen bagenien berriz programa erori egiten zen. Hainbat aukera aztertu ondoren, aurrerapenik lortzen ez genuenez, debugeatzea erabaki genuen. Hori egin ahala izateko, konpilazio fitxategia aldatu eta berriz konpilatu genuen SLStudio softwarea, kasu horretan debugeatzeko beharrezko fitxategiak sortuz. Debugeatzen hasi ondoren, Qt liburutegiekin arazoak zeudela ikusi genuen.

- Hori ikusita, arazoa sakonago aztertzeari ekin genion eta konturatu ginen SLStudiok Qt5 erabiltzen bazuen ere, PCL-ren liburutegiak Qt4 erabiltzen zutela, eta hori zela eta, programa ez zela funtzionatzeko gai. Hau konpontzeko asmotan guztia Qt4 erabiliz konpilatzen saiatu ginen, bai VTK eta baita SLStudio ere. Horrez gain, liburutegiak aurkitu zitzan ld.so.conf fitxategia ere aldatu genuen.

  Hau konpondu ondoren OpenGL-k ere erroreren bat eman zuen, baina VTK konpilatzerako orduan OpenGL-rekin lotzea falta zela ohartu nintzen eta arazoa honako kode lerro hauekin konpondu genuen:

  ```
  #include <vtkAutoInit.h>
  VTK_MODULE_INIT(vtkRenderingOpenGL);
  ```

- Oraindik ere programaren exekuzio egokia lortzen ez genuenez, aztertzen jarraitu eta PCL-k vtk5.8 erabiltzen zuela ohartu ginen. Horrek gure konpilazio guztiak hankaz gora jartzen zituen, izan ere berriro instalazio osoa burutu beharko genuen eta gainera vtk5.8 konpilatu Qt4-rekin. Horretarako, beharrezko baliabide eta jakintzak ez genituenez, eta denboran atzerapausu handiegia emango genuenez, erabaki

bat hartu genuen: gure sisteman Ubuntu 16.04 ezarri sistema eragile gisa eta bertan instalatu SLStudio.

- Gure sisteman jada Ubuntu 14.04 eta Ubuntu 12.04 instalaturik zeuden eta ez genuen aukerarik hauek ezabatzeko. Hori zela eta, ahalik eta modu egokienean partizio berriarentzat toki bat egin nuen disko gogorrean eta bertan Ubuntu 16.04 instalatu nuen.

- Hau egin ondoren, arazorik gabe instalatu genituen SLStudiorentzat beharrezkoak ziren dependentzia guztiak (SLStudio Ubuntu 16.04-en garaturik zegoenez paketeak jada modu egokian konpilaturik zeuden lan hori nire kabuz burutu beharrean). Behin instalazio guztiak burututa, SLStudio konpilatu eta funtzionatzeari ekin zion arazorik gabe.

- Programaren barneko arazo batzuk zirela eta kodea begiratzen aritu ondoren programa martxan jartzea lortu genuen. Bestetik, gure kameraren API-a programak erabitzen zuen bertsioa baina berriagoa zenez hainbat funtzioren deiak eguneratu behar izan genituen.

4.2 Konfigurazioa

Behin programa instalatuta, hainbat konfigurazioa burutu behar izan genituen, modu egokian funtziona zezan. Bigarren pantailaren konfigurazioa izan zen lan gehien eman zigun konfiguratze-fasea.

4.3 Pantaila independenteak

Ezarpenetan patroiak proiektatuko ziren pantaila hautatu genuen eta softwareak patroiak proiektatzeari ekin zion, baina honekin batera beste arazo bat etorri zitzaigun. Ubuntun bigarren pantaila bat HDMI bidez konektatzen dugunean pantaila hedatu gisa konfiguratzen da. Hori zela eta, SLStudio softwareak patroiak proiektatzeari ekiten zionean bigarren pantailan soilik proiektatu beharrean pantaila hedatu osoan zehar proiektatzen zuen. Hori zela eta gure interfazea oztopatu eta patroiak proiektatzen ziren bitartean edozein ekintza egitea galarazten zuen.

Hori konpontzeko aukera bakarrenetakoa bigarren pantaila hori pantaila independente gisa konfiguratzea litzateke. Modu horretan, SLStudio-ren ezarpenetan patroiak non proiektatu aukeratu beharko genukeenean, bigarren pantaila hori (proiektorea) aukeratzeko ahalmena izango genuke. Ubuntuk pantailak kontrolatzeko eskaintzen dizkigun tresna grafikoek ez digute bi pantaila independente konfiguratzeko aukera ematen. Linuxen hori egin ahal izateko bi aukera nagusi daude:

- nVidiako driverrak erabili konfigurazioa burutzeko
- xorg.conf fitxategia sortu eta bertan burutu pantailen konfigurazioa

nVidiako driverrak erabiliz prozesua grafikoagoa zenez, lehen probak horiekin egin genituen. Horretarako, nVidiako driverrak aktibatu genituen periferikoen kontrola hauen esku utziz. Hala ere, driver horiek aktibatu eta ordenagailua berrabiarazi genuenean, sistema bigarren pantaila soilik detektatzera igaro zen, gure monitore nagusia beltzean agertzen zelarik. Orduan ohartu ginen arazoaz: Gure monitore nagusia txartel grafiko integratuaren bidez kontrolatua zegoen eta proiektorea berriz nVidiaren txartel grafikoaren bidez. Hori zela eta, nVidiaren driverrak aktibatzen genituenean proiektoreko grafikoak soilik jartzen ziren martxan. Konponbidea pantaila nagusia ere nVidiaren sarreretara konektatzea litzateke, baina hemen ez genuen VGA konexiorik. Hori zela eta, grafikorik gabeko sesio batetik berriro ere Ubunturen ohiko driverra instalatu genuen txartel grafikoen kontrolerako. Hau egin ondoren, proposaturiko bigarren aukerari ekin genion xorg.conf fitxategia konfiguratuz (ikus https://wiki.archlinux.org/index.php/Xorg_(Espa%C3%B1ol)). Hau, linuxek pantailen konfiguraziorako erabiltzen duen aplikazio publikoa da.

Interneten sakon bilatzen aritu ondoren, beharrezko konfigurazioa burutzeko xorg.conf.d direktorioan fitxategi bat gehitu behar genuela konturatu ginen. Lehengo ordenagailuetan periferikoen konfigurazioak xorg.conf fitxategiaren bidez burutzen baziren ere, gaur egungo sistema gehienetan xorg.conf.d direktorioan fitxategi ezberdinak gehitzen dira XX-izena.conf formatua dutelarik eta sistema abiatzen denean, berak sortzen xorg.conf fitxategia. XX balio hori zenbaki bat da eta honek fitxategi bakoitzak prozesu horretan izango duen lehentasuna adierazten du. Aipatutako direktorioaren bide osoa /usr/share/X11/xorg.conf.d da.

Beraz, gure X pantaila independenteak sortzeko 10-intel.conf izan da bertara gehitutako fitxategia. Bertan beti egin ohi den eskema egin dugu eta ondoren txartel grafikoa bi aldiz definitu dugu aukera ezberdin bat emanez pantaila bakoitzarentzat, eta azkenik bi pantailak definitu ditugu.

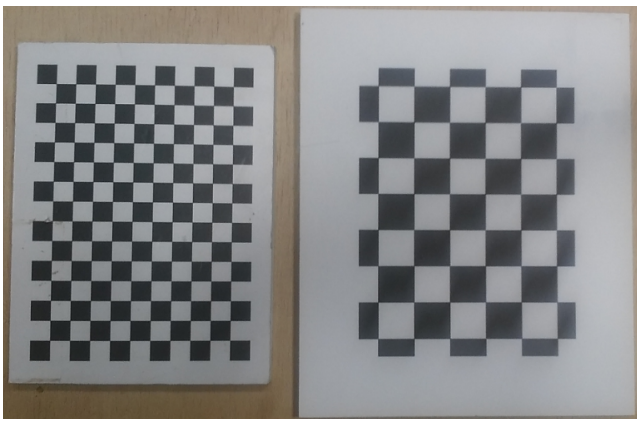
Fitxategi honekin, sistema exekutatzen dugunean pantaila bakoitza bere aldetik abiatzen da, guk lehenengoan lan egiten dugularik. Baina programa exekutatzerako orduan patroiak bigarren pantailan proiektatzeko agintzen diogu eta hala egiten du, gure lanean oztopatu gabe.

4.4 Kalibrazioa

Sistema konfiguratzearekin amaitu genuenean kalibrazioari ekin genion. Programa honek, kalibrazioa burutzerako orduan, interfaze berezi bat kargatzen du eta bertan zenbait ezarpen sartzeko aukera ematen digu:

- Kalibrazio-taularen zutabe kopurua
- Kalibrazio-taularen errenkada kopurua

Bertan guk erabiliko genuen kalibrazio taularen neurriak sartu genituen eta kalibrazioari ekin genion 8x6 eta 15mm-ko aldeak zituen taula erabiliz. Errendimendua hobetzen zen ikusteko taula ezberdin gehiago ere erabili genituen (ikus 4.2 irudia).



4.2 Irudia: Erabilitakoen artean, bi kalibrazio taula

Argi-egituratuan oinarrituriko sistema gehienetan, kalibrazioa ongi joan den jakiteko, **birproiekzio** errore bat kalkulatzen da. Errore hau kalkulatzeko, proiektatze den puntuaren arteko eta kameraren bitartez identifikatu puntuaren artean dagoen distantzia euklidearra kalkulatzen da. Guk ere, lehen estimazioa egiteko behintzat, balio hau erabili genuen kalibrazioa ona izan zen jakiteko.

Hasiera batean behintzat kalibrazioa ez zen batere ona eta programa ez zen ezer berreraikitzeko gai. Ezarpen ezberdinak probatuz aritu ginen eta baita kalibrazio taula ezberdinak ere baina ez genuen itxurazko emaitzarik lortzen. Lortutako puntu-hodeiak 4.3, 4.4, 4.5 irudietakoak baina are okerragoak ziren.

Soluziorik bilatzen ez genuenez, softwarearen egileari korreo bat bidali genion laguntza eske, gure arazoa azaldu genion eta konponbide edo ideiaren bat eskatu genion kalibrazioarekin aurrera egin ahal izateko. Erantzuna jaso genuenean bi arazo posible azaldu zizkigun.

Lehenik eta behin proiekzioen denbora egokia zen ziurtatzeko iradoki zuen eta hori zela eta, badaezpada ere denbora hau 150ms-ra igo nuen; hala ere, arazoaren jatorria hau ez zelakoan nengoen. Horrez gain, errorerik ohikoena zein zen azaldu zuen: Kamera eta proiektorearen artean erantzun linear baten beharra zegoen gamma balioetan. *Phase shifting* metodo guztietan, argi intentsitatea inkrementatzen zen heinean kamerak atzematen duen distiran eman beharko zen gorakadak ere berdina izan beharko lukeela aipatu zuen programaren exekuzioa egokia izan zedin. Horretarako, gure iturburu-kodean aurki genezakeen determineGammaResponse.m script-a erabil genezakeela ere aipatu zuen.

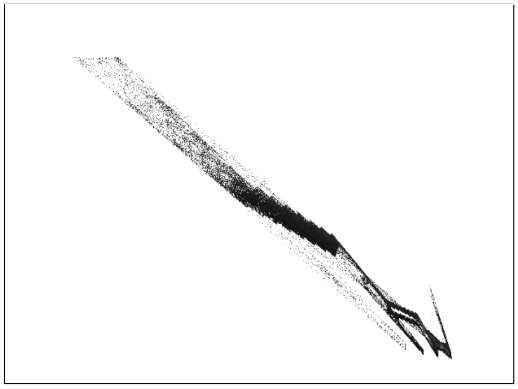
Bere aholkuak jarraituz, Matlab instalatu eta script-a exekutatu genuen baina, gure kamerarako prestaturik ez zegoenez, gure API-arekin integratu beharko genukeen, eta hori zela eta, aukera hau alde batera utzi genuen.

Hau ikusirik, Gamma balio ezberdinekin probak egitea erabaki genuen programaren portaera aztertuz eta emaitza onenak ematen zituen balioa aukeratuz. Baina, gamma balioa aldatzeko orduan, proiektorearen ezarpenetara jo genuenean honek aukera hau eskaintzen ez zigula ikusi genuen. Hori zela eta, konponbide gisa bigarren pantailaren gamma balioa aldatzea pentsatu genuen, hau da proiektorea konektaturik genuen pantailarena. Gure sistema bi pantaila independenterekin konfiguratuta genuenez ezin genituen ohiko ezarpenak erabili, gure monitore nagusian soilik atzematen baitziren aldaketak. Hori zela eta, zuzenean pantailen konfiguraziorako erabili genuen *Xorg* futxategia aldatu genuen eta gamma aldatzea lortu genuen.

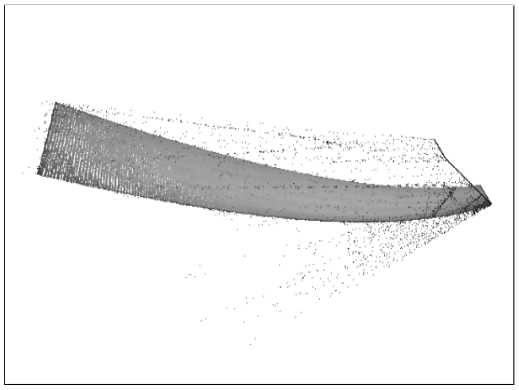
Hau horrela, lehenik eta behin diafragmaren irekiera eta esposizio-denboraren arteko konbinazio egoki bat bilatu genuen eta hori erabiliz gamma balioa aldatzeari ekin genion kalibrazio optimoa topatu nahian. Gamma aldatzen genuen bakoitzean *lightmd* kontroladorea berrabiarazi beharra geneukan eta honek irekiak genituen leiho guztiak ixten zizkigun. Horrela, balio honenak aurkitu nahian, bilaketa dikotomikoaren oinarriak jarraituz proba batzuk egin genituen.

Azaldu erreproiekzio errorea, eta zergatik erabili dugun proiekzio errorea.

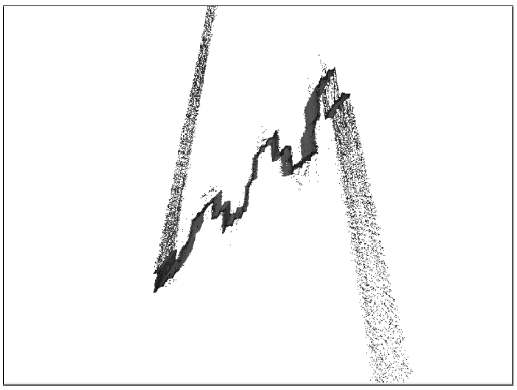
Defektuzko gammaren balioa 1 da ohiko pantailetan. Hori dela eta, bilaketa gauzatzeko lehenik eta behin 0.5, 2 eta 3 balioekin probatu genuen. Emaitza onenak 2 balioarekin lortu genituen, eta horrela pausuz pausu bilaketa eremua murriztuz joan ginen gamma balioa 1.8 balioan zehaztea erabaki genuen arte.



4.3 Irudia: Kalibrazio errore handia



4.4 Irudia: Kamera eta proiektorearen arteko erlazio okerra



4.5 Irudia: Phase Shifting gamma balio okerrarekin

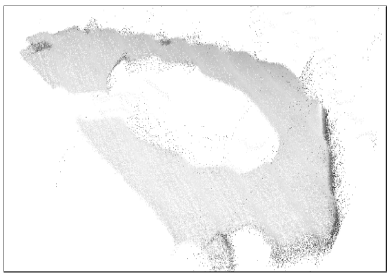
Behin hori lortuta diafragmaren irekiera eta esposizio denboraren arteko konbinazio egoki bat bilatzen aritu ginen kamerara argi kopuru egokia sar zedin.

4.5 Patroien hautapena

Esan dugun bezala argi-egituratuaren munduan kodetze ezberdin ugari aurki ditzakegu, bakoitza bere ezaugarri propioekin. SLStudio softwareek, argi egituratuan murgildurik dauden garatzaileei ateak irekiak uzten ditu, kodetze sistema bertara gehitzeko orduan.

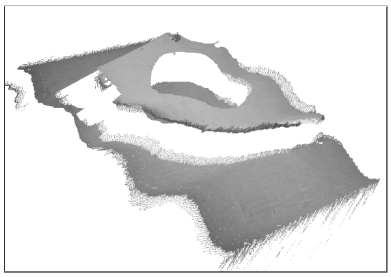
Hori dela eta, aukera ezberdinak eskaintzen dizkigu; horietako batzuk azkartasuna eskainiko digute, beste batzuk berriz zehaztasuna, eta beste batzuk oraindik garapen fasean egongo dira emaitza eskasak lortuz. Probaketa fase bat garatu genuen emaitza onenak lortzen zituzten kodetzeak aukeratuz. Aukeraketa nahikoa garbi zegoenez, zuzenean burutu genuen.

Alde batetik, kodetze batzuk zuzenean alde batera utzi genituen. Kontuan hartu behar da, teknika hauek oso patroi gutxi proiektatzen dituztela, 3 soilik (ikus 4.6, 4.7 eta 4.8 irudiak).



4.6 Irudia: 3 Phase Unwrap

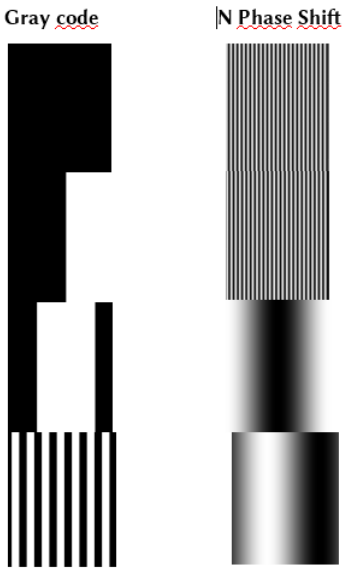
4.7 Irudia: 3 Phase



4.8 Irudia: 2 + 1 Phase

Beste patroi batzuen bitartez berriz emaitza hobeak lortu genituen baina, azkenean ohiko *gray code* eta *N phase shift* patroiak aukeratu genituen konparazioa egiteko. Konparazioa atalean kodetze hauen bitartez lorturiko puntu-hodeiak ikusteko aukera izango duzue eta antzemango da 4.6, 4.7 eta 4.8 berreraikitzeen aldean, beste zehaztasun eta kalitate maila bat dutela.

Aukeratutako bi kodetze-teknikek 11 patroi erabiltzen dituzte beraien berreraikitzeak egiteko. 4.9 irudian ikus ditzakezue kodetze sistemak erabiltzen dituen patroi batzuk. Patroi kopuru altua ez izateak zehaztasunean galera ekar dezake, baina, sistemaren berreraikitze-tasa optimizatzerako orduan abantaila bat izango da.



4.9 Irudia: SLStudiok erabiltzen dituen patroi batzuk

5. KAPITULUA

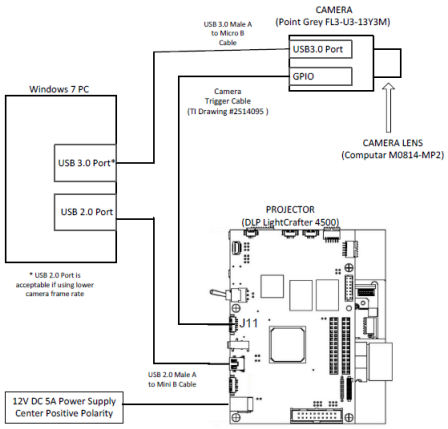
Texas Instruments 3D Application

Behin gure sisteman SLStudio konfiguratu eta proba batzuk egin ondoren, gure proiektuari zegokion enpresak eskaintzen zigun enpresak softwarea instalatu eta konfiguratzeari ekin genion. Software hau ere izaera irekikoa da eta bere iturburu kodea internetetik jaitsi eta gure sisteman konpilatzeko aukera eskaintzen digu. Itxura batean behintzat aurreko sistemaren antzeko formatuko software bat zen baina proiektorearen enpresa berdinak garatua izan zenez hardware bateragarritasun arazo gutxiago espero genituen.

SLStudio softwarearen moduan C++ lengoaian idatzirik zegoen eta horrek, kodea aldatzeko gaitasuna ematen zigun, lengoaia hau ezaguna baikenuen. Patroiak proiektatzerako orduan, SLStudio-ek erabiltzen ez zuen metodo bat erabiltzen zuen. Lehenik eta behin, proiektatu beharreko patroiak proiektorearen flash memorian gordetzen zituen USB kable bidez konektatuz (ikus 5.1 irudia). Patroiak proiektatu nahi zituenean, zuzenean bertatik egiten zuen.

Bere kodeari dagokion egitura berriz honakoa da:

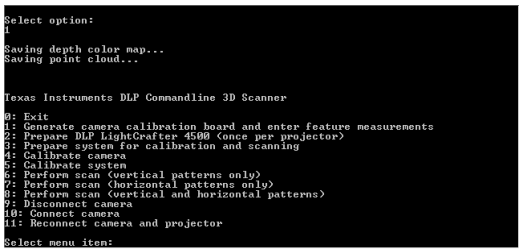
- **calibration:** Modulu honetan kamera kalibratzeko kodea, proiektorea kalibratzeko kodea eta kalibrazio datuen tratamendua egiten duen fitxategia aurki ditzakegu.
- **camera:** Kameraren klase orokorra izateaz gain, Point Grey kameraren API-aren integrazioa eta OpenCV-k kamerak tratatzeko erabiltzen duen kodea daude.
- **common:** Interfaze grafikoa eta programak erabiliko dituen datu-egiturei buruzko kodea dago modulu honetan.



5.1 Irudia: TI3D-ren konexioen eskema

- **dlp_platforms:** Konpainiaren proiektore ezberdinetarako kodea daukagu, izan ere, software hau Light Crafter 3000, 4500 eta 6000 proiektoreekin erabili daiteke.
- **geometry:** Triangulazioaren prozesua egiten duen kodea aurkitzen da bertan.
- **structured_light:** Berreraikitzeak burutzeko erabiliko dituen kodetzeak zehazten dituen kodea dago, *gray code* eta *phase shifting*.

Software honek, berreraikitako puntu-hodeiak erakusteko leiho bat irekitzen bazuen ere, terminalean exekutatzen zen, hau da, ez zuen interfaze grafikorik (ikus 5.2 irudia).



5.2 Irudia: TI3D softwarea exekuzioan

5.1 Instalazioa

Esan bezala, softwarea sisteman instalatzerako orduan bi aukera eskaintzen zizkiguten. Alde batetik, zuzenean exekutagarria instalatzeko aukera ematen ziguten eta bestetik,

iturburu-kodea konpilatu eta exekutagarria guk sortzeko. Lehenik eta behin, denbora aurrezteko ahaleginetan zuzenean exekutagarria instalatu genuen baina prozesuak aurrera jarraitu zuen heinean, hainbat arazo tarteko, iturburu-kodea konpilatu eta gure exekutagarri propioa sortzeko beharra izan genuen. Jarraian azaldurik dituzue bi prozesuak.

5.1.1 Exekutagarriaren instalazioa

Lehenik eta behin aipatu behar dugu software honek alde batetik exekutagarria eskaintzen digula eta bestetik iturburu kodea deskargatu eta instalatzeko aukera ematen digula. Horrez gain, bai exekutagarria eta baita iturburu-kodea instalatzeko ere bi gida erabilgarri eskaintzen dizkigu, hainbat dependentzia baititu software honek eta guztiak modu egokian instala ditzagun.

Horrez gain, dakizuenez software hau Windowserako soilik dago eskuragarri eta hori dela eta Windows-dun sistema bat behar genuen instalazioa burutzeko. Hemen etorri zen lehen arazoa, izan ere, Tecnalia uzten ziguten ordenagailuak ez zuen Windows sistema eragilea eta hori zela eta honen instalazioa proposatu genuen. Hala ere, enpresaren politika zela eta segurtasun arrazoienagatik ezin izan genuen Windows instalatu erabiltzen ari ginen ordenagailuan. Hori zela eta, beste ordenagailu bat eman ziguten. Ordenagailu hau 32biteko mahaigainekoa zen eta ez zituen gure betebeharrak betetzen, ez beharrezko portuen aldetik, ezta potentziaregatik ere. Hau zela eta, azkenean nire ordenagailu eramangarrian instalatu nuen Windows sistema eragilea eta bertan hasi ginen instalazioa burutzen.

Instalazioarekin ekiteko, lehenik eta behin Meshlab softwarea eta erabiliko genuen kameraren driverrak instalatu genituen gure sisteman, hau da IDS-ek eskaintzen dizkigun driverrak. Software honetan garatzaileek erabili zuten kamera Point Grey kamera da eta hori dela, kontuan hartu genuen programa ez zegoela prest gure kamerarekin lan egiteko eta zenbait aukerek ez zutela funtzionatuko.

Instalazioarekin aurrera jarraitu genuen eta ikusi genuen programaren funtzionamendua optimoa izan zedin USB 3.0 kable bat erabili beharko genuela kamerarentzat eta baita hardware *triggering*-a ere horretarako beharrezko kablea erabiliz. Gure ordenagailuak 3.0 sarrerarik ez zuenez, aukera hori alde batera utzi genuen baina etorkizuneko instalazio baterako kontuan izanda.

Programa exekutatzerako orduan C++-eko runtime liburutegi batzuk falta zirela ikusi genuen eta hauek instalatu genituen baina emaitza onik lortu gabe. Hau ikusita, Visual Stu-

dio softwarea instalatu genuen beharrezko liburutegiak instala zitzan eta azkenean funtzionatzea lortu genuen.

5.1.2 Iturburu-kodearen instalazioa

Kasu honetan ere softwareak berak eskainitako konpilazio gida batek burutu beharreko prozesua asko erraztu zigun. Gida hau jarraituz, hainbat software instalatu genituen gure sisteman softwarea konpilatu ahal izateko:

- Doxygen, programaren dokumentazioa sortzeko erabiliko den tresna
- Qt liburutua, konpilazioan OpenGL-rekin loturik dagoena. Qt insterfaze grafikoak sortzeko erabiltzen da eta OpenGL beriz 2D eta 3D aplikazio grafikoak sortzeko erabiltzen da.
- Mingw32 konpiladorea, gure iturburu-kodea konpilatzeko beharrezkoa dena.
- cMake, konpilazio prozesua kontrolatzeko erabiliko den programa.
- OpenCV, programak funtzionatzeko beharrezko liburutegiak
- FlyCapture liburutegiak, hauek PointGrey FlyCapture kamerarentzat baldin badira ere beharrezkoak zaizkigu konpilazioa burutzeko.

Hauek guztiak instalatu ondoren cMake softwarea erabili genuen lehenik eta behin SDK konpilatzeko eta behin hau egin ondoren, LightCrafter 4500 3D Scanner exekutagarria sortzeko. Hasiera batean dependentzia eta loturekin hainbat arazo izan bagenituen ere guztiak konpontzea lortu genuen. Behin instalazio eginda, debugeatzea interesgarria izango litzatekeela pentsatu genuenez berriro ere konpilazioa burutu genuen baina oraingoan konfigurazio fitxategietan debugeatzeko beharrezko sinboloak sortzeko aukera aktibatuz.

5.2 Konfigurazioa

Behin programa instalaturik bere ezarpenak aldatzeko aukera ematen duten duten hainbat fitxategi aldatu ditugu. Ezarpen orokorrak definitzeko DLP_LightCrafter_4500_3D_Scan_Application_Config.txt testua erabiltzen zuen programa honek aukera nagusiak hautatzeko eta hori horrela izanik, lehenik eta behin kameraren hautapena egin genuen. Kasu honetan bi aukera eskaintzen zizkigun:

- Point Grey Flycap kamera
- OpenCV kamera

Guk garatzaileen kamera berbera erabiltzeko asmorik ez genuenez kamera kontrolatzeko OpenCV-ren bidez gauzatu genuen kameraren konexioa. Hau horrela izanik, ondoren proposatzen zituen kameraren hainbat ezarpen aldatzeko arazoak izango genituen, eta hori zela eta, beste bideren bat aurkitu genuen ezarpen horiek aldatzeko.

Hau egin ondoren, fitxategi honetan proiektatu beharreko patroien zein motatakoak izango ziren ere aukeratu beharko genuela ikusi genuen. Bi aukera eskaintzen zizkigun programa honek:

- GrayCode
- Hybrid Three Phase Shift

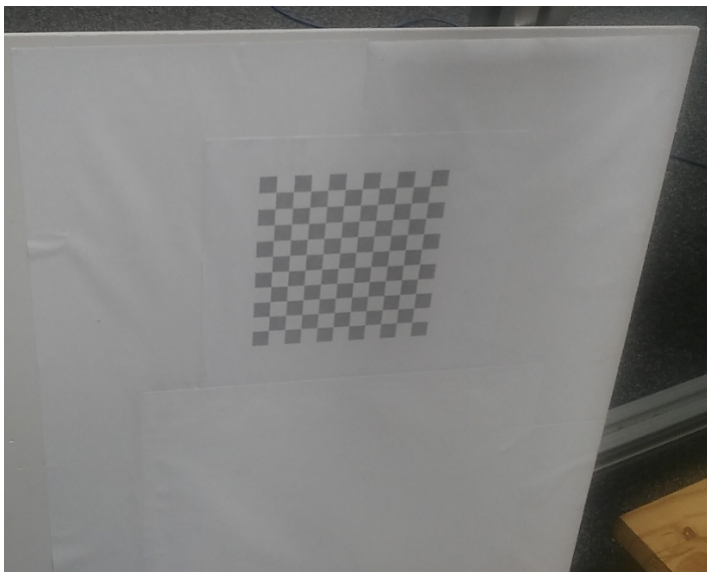
config_camera.txt eta config_projector fitxategiak aldatu genituen programa gure beharretara egokitzeko. Horrez gain, proiektoreak funtziona zezan behar zuen firmwarea instalatu eta TI3D softwarearekin lotu genuen.

Behin programak funtzionatzea lortu genuenean kameraren oinarrizko funtzionamenduarekin pare bat arazo izan genituen. Hasiera batean, kalibrazio prozesuari ekiten genionean, gure sistema ordenailu eramangarriaren web kamerara konektatzen zen, IDS kamera alde batera utziz. Hori ekiditeko web kameraren kontrolagailua gelditu behar izan genuen eta modu programa gure kamerara konektatu zen.

Behin kamerara konektatzea lortuta, jada softwarera iristen zen irudia IDS kamerakoa zen, baina, irudiaren bereizmena ez zen guk esperotakoa. Kameraren bereizmena 2048x1088koa da bere maximoan eta guri iristen zitzaiguna berriz 800x600koa zen. Momentu honetan nahiko blokeaturik gelditu ginen, izan ere, ez genekien zein ezarpen aldatu hau modu egokian jartzeko. Hori zela eta, konponbide bakarra iturburu-kodea konpilatu eta instalatzea iruditu zitzaigun, bertan zuzenean bereizmena aldatzeko asmoz. Kodea debugeatu ondoren, OpenCV-ren defektuzko balioak kargatzen zituela ikusi genuen eta hauek eskuz aldatu genituen gure bereizmenera egokituz. Horrez gain, hemen ikusi genuen balio horiek konfigurazio fitxategitik irakurtzen saiatzen zela eta beraz, hemen bi lerro gehituz, koderik ukitu gabe, bereizmena alda zitekeela.

5.3 Kalibrazioa

Programa honen kalibrazioa aurrera eramateko, kalibrazio-taula bat sortzen du softwareak. Guk hau erabili beharko dugu kalibrazioa egiteko eta honen neurriak sisteman sartu beharko ditugu eskalatzea modu egokian gauzatu dezan (ikus 5.3 irudia). Software honek, kalibrazioa bi fasetan banatzen du: lehenik, kamera kalibratzen, kameraren kanpo eta barne parametroak kalkulatuz, eta baita distortsioa ere.



5.3 Irudia: TI3D softwarearen kalibratzeko sortu genuen kalibrazio-taula

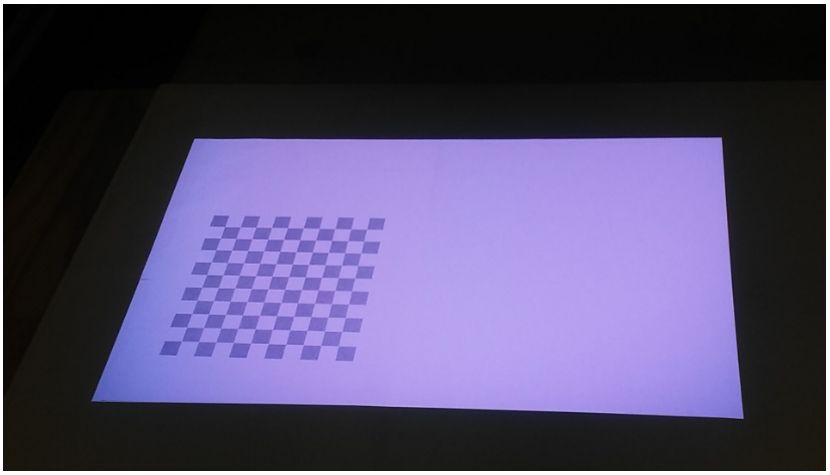
5.3.1 Kameraren kalibrazioa

Kameraren kalibrazioan hainbat pausu eta zehaztapen jarraitu behar izan genituen eta horien artean honakoak izan ziren garrantzitsuenak:

1. Kamera eta proiektorea 20-45º bitarteko angelu bat osatzen duten posizioan kokatu.
2. Interfazeko 4. aukera hautatuk dugu.
3. Kameraren obturadorearen irekiera zehaztu grisak ikusten diren punturik altuenean jarriz eta dirdirak minimizatzen saiatuz. Horrekin batera, kameraren fokua ere egokitu genuen. Aipatutako elementuetako bat aldatuz gero berriz kalibratu beharra dago.

4. 20 irudi inguru atzeman kalibrazio taula posizio, angelu eta distantzia ezberdinetan kokatuz (ikus 5.4 irudi), betiere eskaneatze eremuan.

5. Kalibrazio prozesu honek fokuaren luzera, fokuaren puntua, lentearen distortsioa eta kameraren translazio eta biraketa kalkulatuko ditu kalibrazio taularekiko. Balio hauen kalitatea kalkulatu ahal izateko birproiekzioa erabiliz errore bat kalkulatuko du. Errore hau 2 balioaren azpitik egotea gomendatzen du softwarearen egileak.



5.4 Irudia: TI3D softwarearen kamera-kalibrazioa

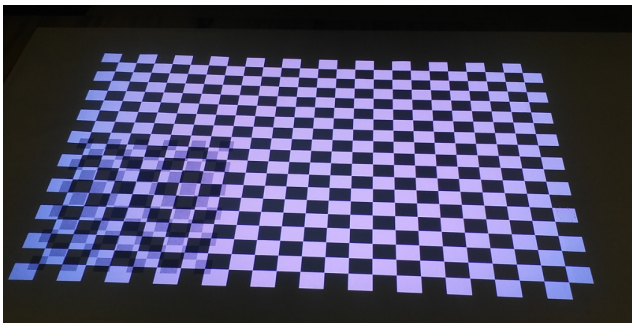
5.3.2 Sistema kalibratzen

Behin kameraren kalibrazioa burutu ondoen sistemaren kalibrazioa burutu genuen. Hau da, proiektorearen kalibrazioa burutu behar dugu, eta horrez gain, proiektore eta kameraren arteko erlazioa finkatu beharko du softwareak, bakoitzaren posizioa zein den jakin dezan. Hau da, proiektorearen barne eta kanpo parametroak kalkulatuko ditu, eta baita lentearen distortsioa ere. Kalibrazio hau burutzeko, berriro ere, aurrez erabili dugun proiekzio patroia erabili beharko dugu. Kasu honetan, proiektoreak posizio ezberdin bakoitzean 3 proiekzio burutuko ditu: zuri solidoa, xake-taula itxurako patroia eta beltz solidoa. Kamerak patroi horietako bakoitzaren irudi bana atzemango du eta horietatik abiatuz kalibrazioa burutuko du. Prozesu hau 20 posizio ezberdinetan burutuko dugu sistema kalibratzeko.

Sistema kalibratzeari ekin genionean lehenengo arazoa segituan etorri zen. Izan ere, esan dudan bezala, kalibrazio metodo honek xake-taula bat proiektatzen zuen bere proiekzio

eremuan, kamerak oso-osorik atzeman beharko zuen patroi bat. Gure sistema konfiguraturik zegoen moduan ordea ez zuen proiekzio eremu osoa atzematen.

Arazo hori konpontzeko asmotan kameraren posizioa mugitzen ahalegindu ginen baina gure egoeran, modun horretan ez genuela emaitza onik lortuko ikusi genuen. Izan ere, gure sistema momentuko muntaiarekin ez zen proiekzio eremu osoa kameran sartzeko beharrezko neurriak hartzeko. Hori zela eta, lortu genuen soluzio bakarra kameraren fokua aldatzea izan zen. Zenbait foku ezberdinekin probak egin ondoren lortu nahi genuen tamainara gehien hurbiltzen zen fokua hartu genuen (ikus 5.5 irudia).



5.5 Irudia: TI3D softwarearen sistema-kalibrazioa

5.3.3 Kameraren parametroak

Erabili dugun IDS kamera industrialak hainbat parametro ezberdin aldatzeko aukera eskaintzen digu lortuko ditugun irudiek gure beharrak ase ditzaten. Horien artean garrantzitsuena esposizio-denbora baldin bazen ere beste parametroekin hainbat proba egin genituen.

Instalatu dugun programak ez zigun kameraren parametroak aldatzeko gaitasunik. Eskaintzen zuen aukera bakarra, kameraren konfigurazio fitxategian OpenCV-ko parametroen bitartez aldatzea zen. Baina, OpenCV ez da gai kamera guztietako parametroak aldatzeko, eta gure kameraren kasuan ez zuen inongo eraginik sorrarazten kameraren parametroak aldatzen ahalegintzen zenean. Hori zela eta, soluzio bat bilatuz, lehenik eta behin pentsatu genuena gure kameraren API-a softwarearekin integratzea izan zen, baina arazoa sakonago aztertuta konponbide errazago bat bilatu genuen.

IDS kamerek parametro multzo bat bere memoria propioan gordetzeko gaitasuna dute. Hau egiteko, kamera IDS-k eskaintzen digun software baten bitartez zabaldu behar dugu

eta bertan parametroak guk nahi ditugun bezala egokitu. Behin hau buruturik parametroak bere memoria propioan gorde behar ditugu.

Hala ere, printzipioz behintzat, kamera hauek defektuzko balioekin abiatzen dira erabiltzen ditugun bakoitzean, eta hori dela eta, kameraren kontroladorea erabiliz aukera hau aldatu genion hasieratze bakoitzean bere memoriako parametroekin egin zezan. Modu honetan, programatik kanpo baldin bazen ere, kameraren parametroak kontrolatzeko gaitasuna lortu genuen, gure kamera kalibratzeko oso erabilgarri egingo zitzaiguna.

Programa exekutatzerako orduan arazo batzuk ozan genituen berriro ere Visual Studioko C++ Runtime liburutegiekin, baina Dependency Walker programaren bitartez dependentziak begiratzea lortu ondoren, 2005eko C++ liburutegiak instalatu eta berrabiarazi ondoren funtzionatzeari ekin zion.

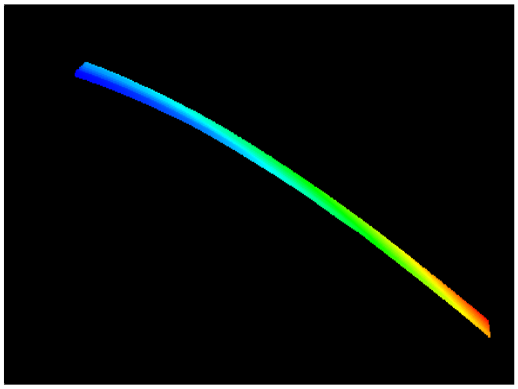
Kamera baten parametroak egokitzerako orduan bi funtsezko elementu daude.

- Kamerara iristen den argi kopurua. Hau, bi modutan kontrolatu daiteke eta konbinazio onena bilatu beharko dugu lorturiko irudia ahalik eta onena izan dadin. Alde batetik, kameraren diafragmaren irekitzea daukagu (fisikoa) eta bestetik, kameraren esposizio-denbora (software bidez ezarriko duguna).

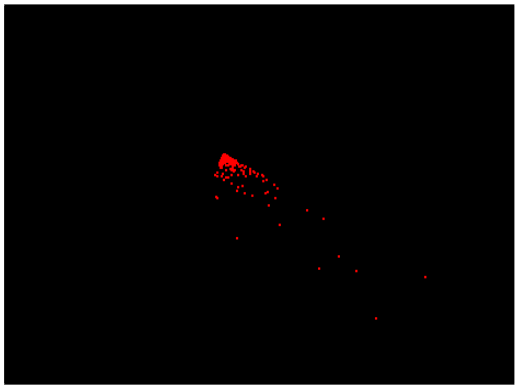
- Kamera modu egokian enfokatu beharko dugu lorturiko irudia ongi definitua egon dadin, horretarako kameraren enfokatzeko tresna erabiliz. Prozesu honetan funtsean eragina duten elementuak bi dira. Alde batetik, berreraiki behar dugun objektura dagoen distantzia izango da, izan ere, distantzia aldatzen den aldatzen den ahala gure fokua galduko dugu. Bestetik, diafragmaren irekiera daukagu, honek kamerara sartuko den argi kopuruan eragingo du (beste hainbat ezarpenek ere eragiten badute ere: filtroak, esposizio denbora etab.) eta baita irudiaren eremu sakoneran ere. Ez nahiz xehetasunetan sartuko diafragmaren irekieraren inguruan, baina hau handiegia bada gure kamerak kamerak enfokatzeko gaitasuna gal dezake eta honekin hainbat arazo ekar ditzake.

Bi parametro hauek modu egokian konbinatu beharko ditugu gure kameraren bitartez lorturiko irudia ahalik eta optimoena izan dadin. Kalibrazio proba ugari burutu genituen, 5.6 eta 5.7 irudietan ikus ditzakezue proba oker batzuk.

Azkenik, behin kalibrazioa irudi onak lortzeko gai zenean, proba batzuk egin eta puntuhodeiak ispilu moduan lortzen zituela ikusi genuen, hau da, sisteman jarrita pieza alderantziz berreraikitzen zuen, Hau da, algoritmoa aplikatzerako orduan, irudia guri interesatzen ez zitzaigun beste orientazio batean hartzen zuen. Hori konpontzeko, IDS kameraren

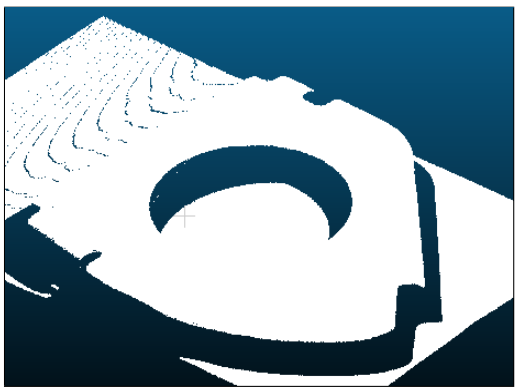


5.6 Irudia: Kamera eta proiektore erlazio okerra



5.7 Irudia: Arazoa, puntuak ez ditu ondo triangulatu

parametroak aldatu genituen, lorturiko irudia ispilu moduan jarriz, eta berriz kalibratu ondoren esperotako irudia lortu zuen (ikusi 5.8 eta 5.9 irudiak). Software honetan aldaketa hori egin genuenean SLStudio softwarean berdina gertatzen zela ikusi genuen, eta beraz, hemen ere beharrezko aldaketak egin genituen lorturiko irudia egokia izateko.



5.8 Irudia: Irudia hasieran lortzen zuen moduan

5.9 Irudia: Irudia ispilu efektua kentzea lortu ostean

5.3.4 Patroien hautapena

Atal honen hasieran deskribatu dugun moduan, software honek bi teknika ezberdin eskaintzen ditu proiektatuko dituen patroiei dagokienez. Alde aldetik Gray code daukagu eta bestetik Phase Shifting-ean oinarrituriko teknika. Hauetariko bakoitzak bere ezarpen propioak eskaintzen ditu eta gu hauek egokitzen ahalegindu gara.

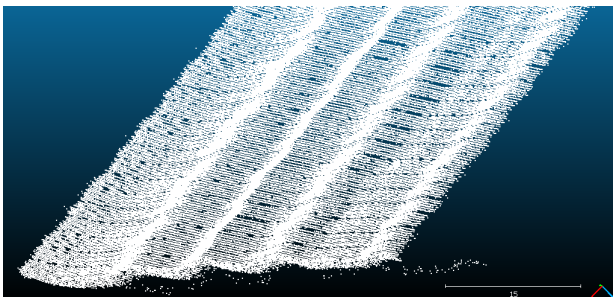
GrayCode-i dagozkionak:

- Proiektaturiko patroi kopurua

- Ea patroi bakoitzaren inbertitua proiektatu behar duen
- Azken proiekzioan lerroen artean egongo den pixel kopuua

Phase Shifting-ari dagokionean beste hainbat parametroren edizioa ahalbidetzen du, baina oraingoz hauek aldatzeko jakintza nahikorik ez daukagunez bere horretan utzi ditugu.

Hala ere, proba batzuk burutu ostean, eta SLStudio-ren moduan bezala Gammarekin proba batzuk egin ostean Phase Shifting metodoarekin ez genuen emaitza onik lortu (ikus 5.10 irudia).



5.10 Irudia: TI3D Hybrid Three Phase Shifting-ekin lorturikoa

Hori zela eta, gure konparazioa erabiltzeko *gray code* kodetzea soilik erabiltzea erabaki genuen. Goran aipatu berri dugun bezala, software honek kodetzea definitzeko hainbat parametro eskaintzen zizkigun gure patroiak zehazteko. Proiektaturiko patroi kopuruari zegokionean, maximoa aukeratu genuen, hau da, 10 patroi. Hala ere, patroi bakoitzaren inbertitua proiektatzeko aukera eskaintzen zuen eta beraz guztira 20 patroi proiektatzen zituen. Inbertituari zegokionean, patroi berdina proiektatzen zuen, baina, zuri eta beltz kolorea tokiz aldatzen zituen. Modu honetan, informazio bakoitza zuen patroiko puntu bakoitza identifikatzeko.

Horrez gain, azken patroian marren zabalera finkatzeko aukera ematen zigun, hau da, proiektatuko ziren marra estuenen zabalera. Hainbat proba eginez, eta oso marra estuek emaitza onik ematen ez zutela bagenekienez, 10 pixel-eko zabalera ezarri genien.

6. KAPITULUA

Konparazioa

Behin bi softwareak gure sisteman instalatu, konfiguratu eta kalibratu genituenean, azken pausuari ekitea falta zitzaigun. Itxura batean, software horien errendimendua ona baldin bazen ere, kalitate-analisi bat burutu nahi genuen horien inguruan software bakoitzaren zehaztasuna zein zen ikusteko. Hala ere, kontuan izan behar da horien kalibrazioa ez dela perfektua eta, beraz, gure konfigurazio eta kalibrazioekin emaitza batzuk lortu baditugu ere, horiek beste modu batean gauzatuz gero, emaitza ezberdinak lortu daitezkeela.

Konparazioan instalatu ditugun bi softwareekin lorturiko puntu-hodeiak erabiltzeaz gain, DAVID software lortutakoak ere erabiliko ditugu, merkatuan dagoen software komertzial bat erreferentziatzat hartu dezagun.

Konparazioa modu egoki batean egiteko, hainbat proba ezberdin burutuko ditugu software ezberdinen bitartez lorturiko puntu-hodeiekin. Ikusmen artifizialeko sistemen bitartez lorturiko puntu-hodeiak lan ezberdinetarako erabili daitezke eta gure azken produktuak izan beharreko ezaugarriak ez dira beti berdinak izango. Hori dela eta, ezaugarri ezberdinak neurtzen ahaleginduko gara azterketa ezberdinak burutuz. Proba gehienak burutzeko, lorturiko berreraikitzeak aurreprozesatu beharko ditugu interesatzen zaigun berreraikitze zatia isolatuz. Hori dela eta, lehenik eta behin, aurreprozesu-fasea azalduko dugu. Hala ere, jarraian azaldurik dagoen moduan, lehen analisia berreraikitze gordinen gainean egingo dugu:

- Aurreprozesatu gabeko puntu-hodeien konparazioa

  Lehenik eta behin, inongo aurreprozesamendurik egin gabe, instalatu ditugun soft-

wareen bitartez lorturiko puntu-hodeiak zuzenean aztertuko ditugu. Izan ere, guri interesatzen zaiguna berreraikitzera goazen objektua baldin bada ere, puntu-hodeiek aurreprozesatu gabe duten itxura ere interesgarria izan daiteke.

- Puntu hodei baten lerrokatzea

  Industria-munduan erabiltzen diren hainbat sistema automatizatuek ikusmen-sistema bat izan ohi dute eginiko piezen kalitatea bermatu ahal izateko (Torre, 2010). Horretarako, alde batetik, guk sortutako 3D ikusmen-sistemaren bitartez lorturiko puntu-hodei ezberdinak izango ditugu eta bestetik, berreraiki ditugun piezei dagozkien CAD modeloak. CAD modeloak gure piezak errore minimoarekin irudikatuko dituzten ordenagailu bidez eginiko modeloak dira. Gure asmoa CAD modeloa eta gure sistemaren bitartez lorturiko puntu-hodeiak lerrokatzea izango da, horien arteko distantzia ahalik eta gehien minimizatuz. Behin hori eginda, gure puntu-hodeiaren eta modeloaren arteko distantziak neurtuko ditugu eta horien analisi bat egingo dugu.

- Berreraikitze osoa

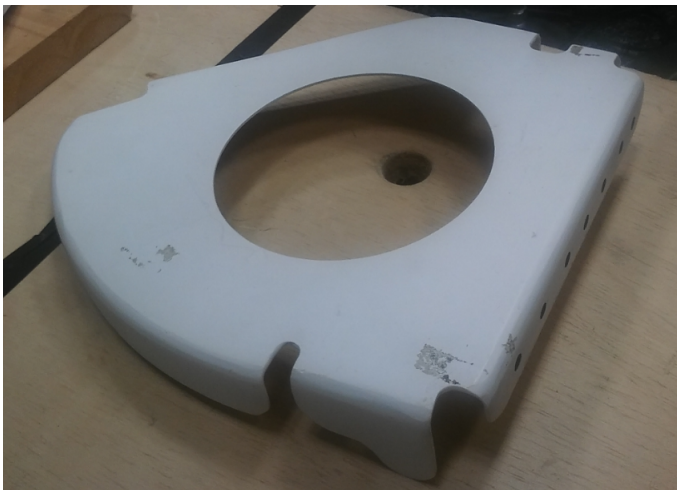
  Batzuetan, ez dugu berreraiki behar dugun objektuaren inongo CAD modelorik izango. Zenbait kasutan, gure helburua aurrean daukagun objektua zuzenean berreraikitzea izango da, ondoren hau beste eginbehar batzuetarako erabiltzeko: berreraikitzetik CAD modeloa sortu, objektua hiru dimentsiotan inprimatu... Hori dela eta, instalaturiko software bakoitzarekin objektu bat guztiz berreraiki dugu, bista ezberdinetatik irudiak lortuz eta, ondoren, horietatik abiatuz, berreraikitze oso bat lortuz. Behin hori eginda, lorturiko emaitzaren kalitatea aztertuko dugu.

- Puntu-hodeiaren analisia zuzenean

  Aurrez buruturiko probak interesgarriak baldin badira ere, funtsean gure sistemaren bitartez lorturiko puntu-hodeia nolakoa den aztertzea ere aberasgarria izan ohi da. Tamaina jakina duen puntu-hodei lau bat aztertuz, informazio interesgarri asko lortu dezakegu: dentsitatea, neurrien zehaztasuna, normalen zehaztasuna, lorturiko emaitza zenbateraino laua den aztertu...

Probak burutu ahal izateko, Tecnalian landua izan den "Quesito"izenez ezagutzen duten pieza erabiliko dugu.

6.1 irudian ikus dezakezuen pieza Airbus enpresarentzat garatua izan da eta hegazkinen hegoen egiturak egonkortzeko erabiltzen da.



6.1 Irudia: *"Quesito"pieza*

Konparazioak egiteko ere, hainbat software eskaintzen du merkatuak eta, oraingo honetan ere, software komertzialak eta software libreak aurki daitezke. Eskuragarri genuen software komertziala DAVID sistemak eskaintzen ziguna zen, izan ere, sistema horrekin lorturiko berreraikitzeak prozesatzeko hainbat tresna eskaintzen du. Hala ere, konfiguratzen ari ginen sisteman software librearen erabilera sustatu nahi genuenez, puntu-hodeien prozesamendurako ere, software libre bat erabiltzeko helburua ezarri genuen.

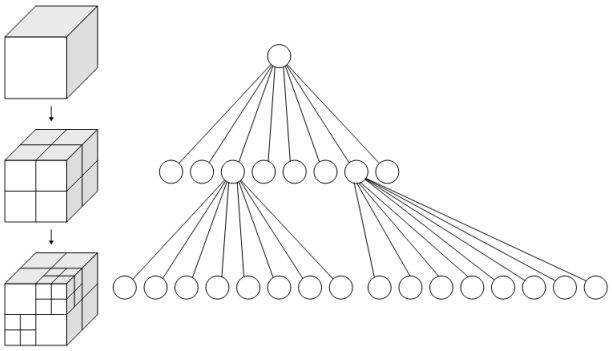
Helburu horren ildora, puntu-hodeien tratamenduan ezaguna den software libre batek atentzioa deitu zigun, Cloud Compare softwareak hain zuzen ere. Hori dela eta, gure konparazioan zehar software hori erabiltzea erabaki genuen. Horrez gain, DAVID-ek eskaintzen zigun softwarearekin alderatu genuen erabilitako softwarearen erabilgarritasun eta errendimendua aztertze aldera eta ikusi genituen emaitzak positiboak izan ziren. Jarraian aurki ditzakezue konparaziorako erabilitako softwareen ezaugarri nagusiak.

6.1 Konparazioa egiteko softwarea: Cloud Compare vs DAVID3D Shape Fussion

Esan bezala, gure konparazioa egiteko bi software proposatu genituen: alde batetik, DAVID 3D Shape Fussion [1] eta bestetik, Cloud Compare software librea [2].

[1] http://www.david-3d.com/en/support/david5/shape-fusion
[2] www.danielgm.net/cc/

Cloud Compare hiru dimentsiotako puntu-hodeiak eta triangelu-sareak editatu eta prozesatzeko softwarea da. Hasiera batean, 3D puntu-hodei trinkoen arteko konparazioak egiteko sortu zen. Hala ere, triangelu-sareen erabilera zabaltzen ari zenez, datu-egitura hori prozesatzeko aukerak integratu zituen. Konputazioak burutzeko erabiltzen duen *octree* egiturari esker, erdi mailako konputazio-ahalmena duen ordenagailu eramangarri baten bitartez puntu-hodei handiak prozesatzeko gai da. *Octree* zuhaitz-egitura duen datu-egitura bat da eta hiru dimentsiotako espazioak banatzeko erabiltzen da (Nevada, 2014). Bertan, errekurtsiboki, espazioko kubo bakoitza 8 kubotan banatzen da (ikus 6.2 irudia).



6.2 Irudia: *Octree datu-egitura*

Cloud Compare softwarea C++ lengoaian idatzirik dago, Windows, Linux eta Mac OS sistemetan konpilatu daiteke eta 32 bit eta 64 bit-etako arkitekturetan dago eskuragarri.

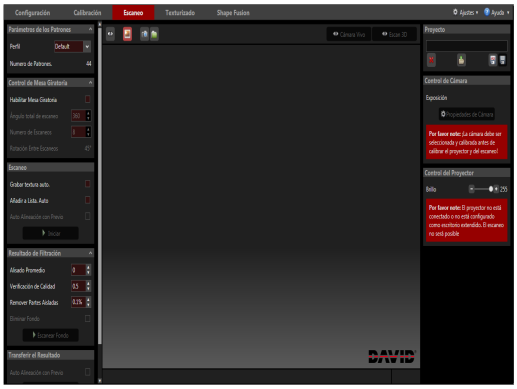
Gure helburutzat software librearen erabilera sustatzea baldin bagenuen ere, erabilgarritasunak ere paper garrantzitsu bat jokatzen zuen proiektuan. Hori dela eta, hasiera batean behintzat, egin beharreko lanak bi software horietan paraleloan burutzen hasi ginen, gure lanetarako egokiena zein zen ikusteko. Aurrera joan ahala ordea, gure beharretara gehien egokitzen zen softwarea Cloud Compare zela ikusi genuen hainbat arrazoiengatik:

- Aukera

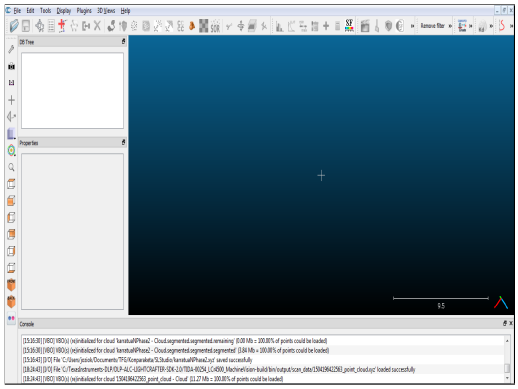
  Cloud Compare oso-osorik puntu-hodeien edizio eta prozesamendurako sorturiko softwarea da. Hori dela eta, DAVID-ek ez dizkigu erabil nahi ditugun zenbait tresna eskaintzen. Erabiltzen duen softwareak ikusmen-sistema osoa barnean hartzen duenez eskaintzen dituen tresnak nahiko oinarrizkoak dira (ikus 6.3 eta 6.4 irudiak).

- Bateragarritasuna

  Gure sistemaren bitartez lorturiko puntu-hodeiak DAVID softwarera inportatzen ha-



6.3 Irudia: DAVID softwarearen interfaze grafikoa



6.4 Irudia: Cloud Compare softwarearen interfazea

si ginenean, horiek irekitzeko gai ez zela ikusi genuen. DAVID softwareak .stl eta .obj fitxategiak soilik irakurtzen ditu, biak ere triangelu-sareak.

Izan ere, aurrez aipatu ez baldin badugu ere, bi datu-egitura nagusi bereizten dira software horien bitartez lorturiko berreraikitzeak irudikatzeko. Alde batetik, orain arte erabili datu-egitura dugun, hau da, puntu-hodeia eta, bestetik, puntu-hodeietatik eratorritako datu-egitura konplexuago bat daukagu, triangelu-sarea hain zuzen ere (ikus 6.5, 6.6 eta 6.7 irudiak). Errepresentazio modu hori, puntu-hodei baten antzekoa da baina bertako puntuak elkarren artean ertz batzuen bidez loturik daude triangelu-sare bat osatuz. Gaur egun, triangelu-sareen erabilera zabaltzen ari bada ere, puntu-hodei irregularretan triangeluak modu egokian sortzea ez da lan erraza.

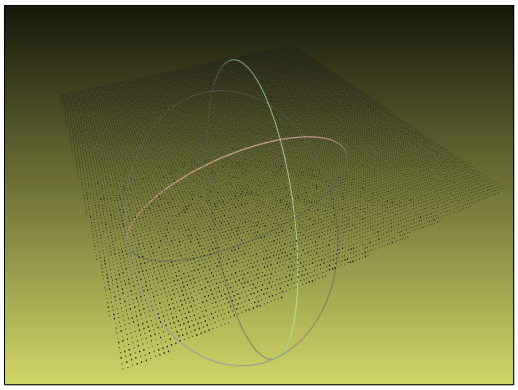
DAVID softwareak, bere berreraikitzeak triangelu-sare gisa itzultzen baditu ere, erabilitako software libreek berriz puntu-hodeiak ematen zizkiguten irteera gisa. Hori dela eta, erabili nahi genuen puntu-hodei bakoitzaren aurreprozesamenduan bi fase gehitu beharko genituzke. Puntu-hodeia triangelu-sare bihurtzeko, alde batetik, puntu bakoitzaren normala kalkulatu beharko genuke eta bestetik, triangeluak eraiki. Hori burutzeko metodo ezberdinak aurkitu genituen:

- PCL liburutegia [3]
- Meshlab software grafikoa [4]

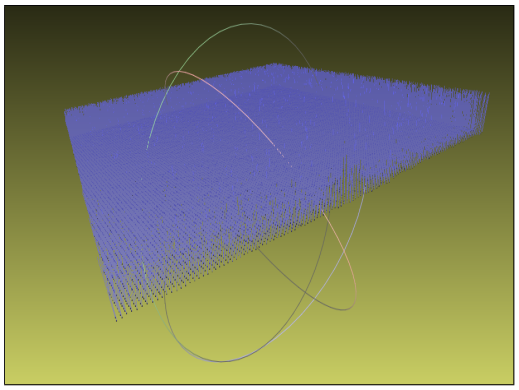
Bi metodoen probak egin bagenitu ere, azkenean DAVID Shape Fussion alde batera uztea erabaki genuen eta, beraz, ez genuen inongo automatizazio prozesurik burutu

[3] www.pointclouds.org
[4] www.meshlab.net

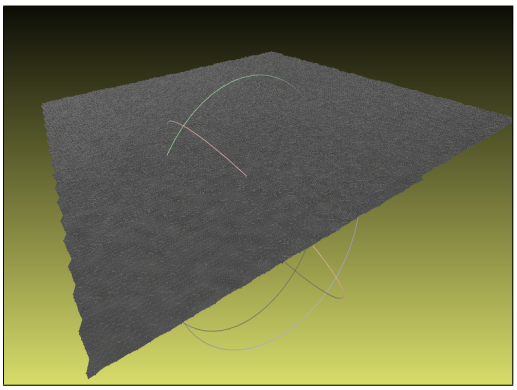
behar izan. Hala ere, etorkizunerako lanetarako, PCL-ren bitartez triangeluen berreraikitzea ahalbidetzea interesgarria litzateke.



6.5 Irudia: Puntu-hodeia



6.6 Irudia: Puntu-hodeia normalekin



6.7 Irudia: Puntu-hodeia triangeluekin

Cloud Compare softwarea berriz, inongo eskaneatze-softwarerekin loturik ez dagoenez, gai da puntu-hodei eta triangelu-sareak errepresentatzeko erabiltzen diren formatu gehienak irakurtzeko. Erabiltzera gindoazen formatuak irakurtzeko eta horiekin lana zuzenean egiteko ez zuen inongo arazorik erakusten, eta emaitzei zegokionez, zehaztasuna bermatzen jarraitzen zuen.

Beraz, aipatutako arrazoi horiek bultzatu gaituzte Cloud Compare softwarea aukeratzera konparazio-software gisa.

6.2 Puntu-hodeien aurreprozesamendua

Argi egituratuaren bitartez lorturiko irudietan, interesatzen ez zaigun informazio ugari egongo da eta horiek tratatu nahi baditugu, lehenik eta behin, interesatzen zaigun zatia

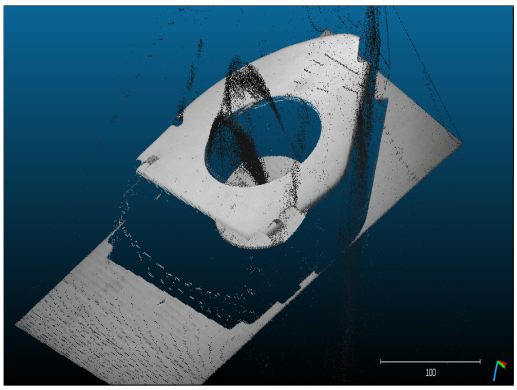
bereiztu beharko dugu. Gure puntu-hodeiek aurreprozesamendu bat jasan behar dute interesatzen zitzaigun zatia isolatzeko.

6.2.1 Segmentazioa

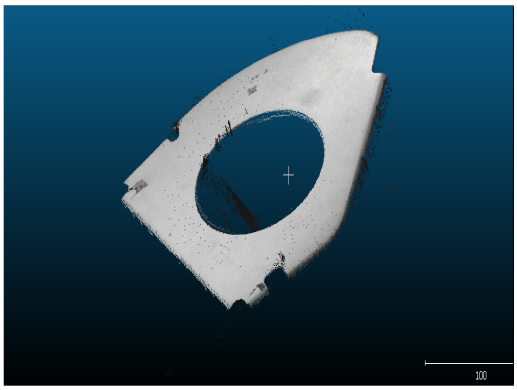
Aurrez esan dugun bezala, gure sistemak eszena ikuspuntu batetik soilik berreraikitzen du, hau da kameraren ikuspuntutik. Hori dela eta, berreraiki bahi dugun objektuaren hainbat atzemate behar ditugu berreraikitze oso bat egiteko. Berreraikitze horietan objektua osatzeko nahikoa informazio baldin badugu ere, lerrokatzea egiteko informazio ugari soberan dugu. Beraz, puntu-hodei bakoitzetik beharrezko informazioa atera behar genuen, hau da, atzemate bakoitzetik interesatzen zaigun piezaren zatia isolatzea. Hori egiteko teknika ezberdinak aurki ditzakegu programa eta liburutegi ezberdinetan.

Prozesua automatizatzeko asmotan bagabiltza *clustering* algoritmo bat erabil daiteke puntuak bere ingurukoekin multzokatu eta guri interesatzen zaigun multzoa soilik aukeratzeko. Hala ere, gure kasuan tratatu beharreko puntu-hodeien kopurua hain handia ez zenez eta, beraz, automazioa zaila izateaz gain beharrezko ez zenez, zuzenean segmentatzea erabaki genuen. Horretarako, hasiera batean Meshlab [5] softwarea erabiltzea erabaki genuen ere, prozesamenduan zituen arazoak zirela eta Cloud Compare softwarearen bitartez egitea erabaki genuen.

Cloud Compare erabiliz, puntu-hodei bakoitzean interesatzen zitzaigun piezaren zatia isolatu genuen (ikus 6.8 eta 6.9 irudiak).



6.8 Irudia: Segmentazioa burutu aurretik



6.9 Irudia: Segmentazioa burutu ondoren

Behin hori eginda, puntu-hodei bakoitzaren tamaina murriztua lortu genuen, maneiatzeko

[5] www.meshlab.net

errazagoak ziren puntu-hodei batzuk lortuz. Hala ere, zenbait puntu-hodeiek zarata ugari zuten oraindik eta hori zela eta, zarata ezabatzeari ekin genion.

6.2.2 Zarata ezabatzen

Kontuan izan behar dugu erabilitako softwareek nolabaiteko zarata sartzen dutela sortzen dituzten berreraikitzeetan. Izan ere, jasotzen duten irudia zarata izan dezakeenez eta proiektorearen islapena ere ingurune guztietan berdina ez denez, zuzena ez den ausazko informazioa agertu daiteke gure berreraikitzeetan. Hori dela eta, behin segmentazioaren pausua egin ondoren, beste pausu bat eman beharra daukagu gure berreraikitzeetatik zarata kenduz, izan ere, lerrokatzeak burutzerako orduan eragin handia izan dezake eta.

Hala ere, aipatu behar dugu alde nabarmena aurkitu genuela software ezberdinen bitartez lorturiko puntu-hodeietan. SLStudio-n lorturiko irudietan zarata handia zegoen, ikus 6.10 eta 6.11 irudiak. TI3D eta DAVID-en bitartez lorturiko puntu-hodei-etan berriz zarata gutxi zegoen.

Segmentazioarekin gertatu zaigun moduan, zarata ezabatzeko algoritmoa aplikatzerako orduan software eta liburutegi ezberdinak eskaintzen zaizkigu.Prozesu hori automatizatzeko asmoa izango bagenu PCL-ek (Point Cloud Library, pointclouds.org) eskaintzen dizkigun tresnak erabiltzea izango litzateke egokiena, baina hau ez denez gure proiektuaren funtsa, ez gara horretan sartuko. Horrez gain, segmentazioak egiteko Cloud Compare softwarea erabili genuenez, eta software honek zarata ezabatzeko tresna bat zuenez, horrekin egitea erabaki genuen.

Gure kasuan, Cloud Compare softwareak puntu-hodeietako zarata ezabatzeko erabiltzen duen algoritmoa, *Statistical outlier removal* da [6].

Algoritmo horren helburua, teknika estatistiko batzuk erabiliz, distantzia maximo bat kalkulatzea da. Hori kalkulatzeko, puntuen artean dagoen batezbesteko distantzia kalkulatzen du lehenik. Behin hori jakinda, balio horri desbiderapen estandarra gehituko dio erabiltzaileak hautaturiko *sigma* balio batengatik biderkatuz. Segidan, puntuz puntu aztertzen du puntu-hodeia, puntu bakoitzak gertuen duen bizilaguna distantzia hori baina gertuago duela ziurtatuz. Arau hori betetzen ez duten puntu guztiak puntu-hodeitik ezabatuak izango dira.

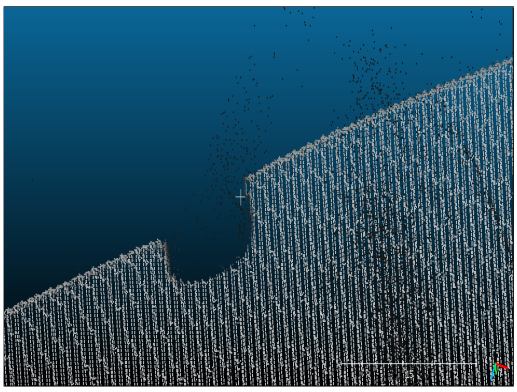
Erabiltzaileari bi parametro eskaintzen dizkio iragazkiaren portaera gure puntu-hodeira

[6] http://www.cloudcompare.org/doc/qCC/CloudCompare%20v2.6.1%20-%20User%20manual.pdf

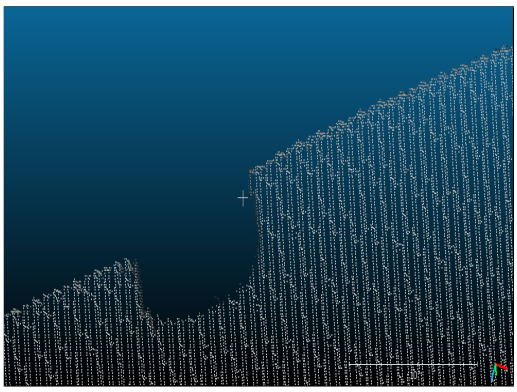
egokitu ahal izateko. Alde batetik, puntu bakoitzaren bizilagunen distantzia kalkulatzeko erabili beharreko puntu kopurua eta, bestetik, algoritmoaren portaera egokitzeko erabiliko den *sigma* balioa.

Hori horrela izanik, lehenengo parametroa aukeratzerako orduan zarataren izaerari begiratu beharra geneukan. Zarataren puntu isolatuz osaturik baldin bazegoen bizilagunen kopurua, 1 balioan ondo egongo litzateke; zarata 4-5 puntuz osaturik baldin bazegoen berriz, bizilagun kopurua hori baina altuagoa jartzea komeniko litzateke.

Sigma balioari dagokionez, puntu-hodei bakoitzarekin proba batzuk egin behar izan genituen, izan ere, puntu-hodeiaren dentsitatea altua ez bazen, zarata kendu beharrean guri interesatzen zitzaigun informazioa ezabatzeko arriskua zegoen.



6.10 Irudia: Puntu-hodeia zaratarekin



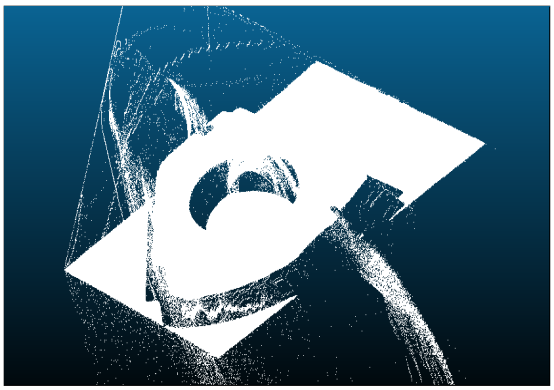
6.11 Irudia: Puntu-hodeia zaratarik gabe

6.3 Puntu-hodei konparazioa prozesatu gabe

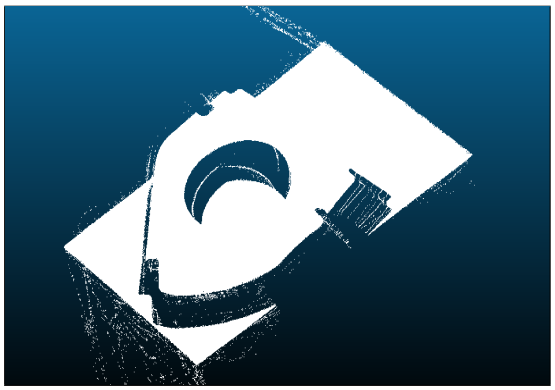
Kasu gehienetan, 3D sistemen bitartez lorturiko puntu-hodeiek aurreprozesamendu baten beharra izaten dute. Zenbaitetan, programak berak iragazki batzuk erabiltzen ditu lortutako emaitzan ager daitekeen zarata ezabatzeko edo puntu-hodeia leuntzeko. Modu horretan, sistemak lortzen duen informazio gordina erabiltzailearentzat egokitzen du baina, prozesu horretan informazio interesgarria galtzeko arriskua ere badago.

Guk dakigunez, SLStudio eta Texas Instruments (TI3D) softwareek ez dute inongo iragazkirik aplikatzen lortzen dituzten iragazkietan baina DAVID-en kasuan, ezin dugu ziurtatu hori horrela denik, ez baikara kodea ikusteko. Jarraian dauden irudiek (ikus 6.12, 6.13, 6.14 eta 6.15 irudiak) hiru sistemen bitartez lorturiko emaitzak inongo iragazki edo tratamendurik aplikatu gabe erakusten dituzte. Aurrez esan dugun bezala, SLStudioren kasuan

bi kodetze ezberdin erabiliko ditugu, probaketa-fasean biek eman baitzituzten emaitza onak.



6.12 Irudia: SLStudio gray code



6.13 Irudia: SLStudio N phase shift

6.14 Irudia: TI3D gray code

6.15 Irudia: DAVID 3D

Garbi ikus daitekeen moduan, proba honetan galtzen ateratzen den softwarea SLStudio da. Bere hodeietan aurki dezakegun zarata beste softwareekin ikus dezakeguna baina handiagoa da. SLStudioren kasuak *gray code* kodetzea edo *N phase shift* kodetzea erabilita ere sortzen duen zarata beste puntu-hodeietan sortzen dena baina handiagoa da. TI3D softwarearen kasuan ia zaratarik ez duela ikus dezakegu eta DAVID-en kasuan zaratarik sortzen ez duela esan dezakegu.

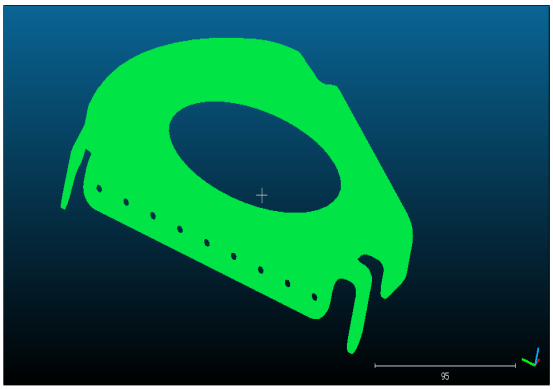
Hala ere, lehen itxura honek ez du esan nahi SLStudiorekin lorturiko berreraikitzeen kalitatea txarra denik. Nahiz eta horrek aurreprozesamendua zertxobait konplikatuko duen, zarata hori ezabatu eta interesatzen zaigun objektuaren zatia isolatzeko tresna ezberdinak existitzen dira, eta jarraian horietako batzuk aplikatuko ditugu.

6.4 Lerrokatzeen arteko konparazioa

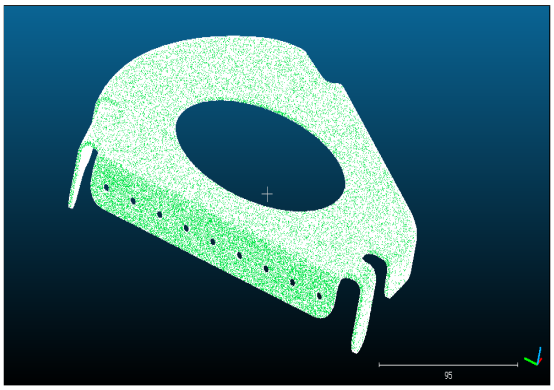
Gure sisteman erabilitako softwareen kalitatea neurtzeko burutuko dugun proba honetan, aurrez azaldu dugun objektuaren puntu-hodeiak beharko ditugu bista jakin batetik, goitikotik hain zuzen ere. Behin puntu-hodei horiek aurreprozesaturik ditugula, bere modeloarekin lerrokatuko ditugu jarraian azaldutako teknika erabiliz. Lerrokatzeak, puntu-hodeien kalitatea neurtzeko aukerak eskaintzen dizkigu, izan ere, behin hori eginda bi puntu-hodeien artean dauden distantziak neurtzeko gai izango gara, eta horien analisi bat eginez ondorioak atera ditzakegu.

Erabiliko genituen modeloei zegokienez, oinarrizko puntuez soilik osaturiko triangelu-sareak genituen. Ez genekien Cloud Compare softwareak zer nolako jokaera izango zuen triangelu-sareekin lan egiterako orduan, eta arrazoi horrengatik, hasiera batean oinarrizko triangelu-sare horri zegokion puntu-hodeia ere sortu genuen. Biak edukita, lerrokatzea paraleloan aurrera eramateko asmoa genuen, bai puntu-hodeiarekin eta baita triangelu-sarearekin ere, emaitza onenak non lortzen ziren ikusteko.

Triangelu-saretik abiatuta, aipatu berri dugun puntu-hodeia sortzeko, guri eman ziguten oinarrizko triangelu-sarean *sampling* bat aplikatu beharra daukagu. Prozesu horrek ausazko puntuak sortuko ditu triangelu-sarearen azaleran. Prozesua, triangelu-sarean sartu nahi ditugun puntu kopuruaren bitartez egin dezakegu, edo baita gure puntu-hodeian izan nahi dugun dentsitatea aukeratuz ere. Gure kasuan, puntu kopuruaren bitartez egin genuen, gure berezko puntu-hodeiak baina trinkoagoa zen puntu-hodei bat sortuz, zehaztasuna irabazten baita.



6.16 Irudia: TI3D gray code



6.17 Irudia: DAVID 3D

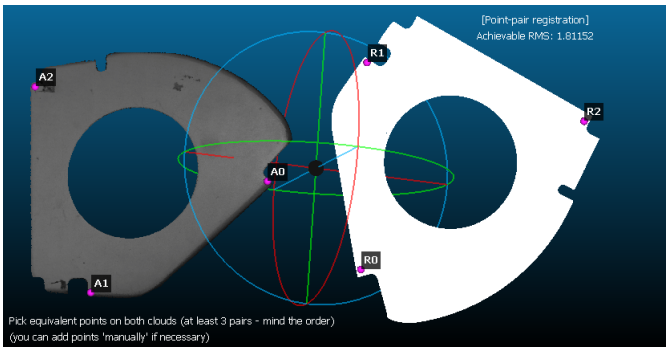
Behin 6.17 irudian ikusten den emaitza lorturik, lerrokatzeari ekin genion. Jarraian le-

rrokatzea pausuz pausu azalduta duzue eta, ondoren, lorturiko emaitzen irudi, grafiko eta ondorio batzuk.

6.4.1 Lerrokatzea puntu aukeraketaren bitartez

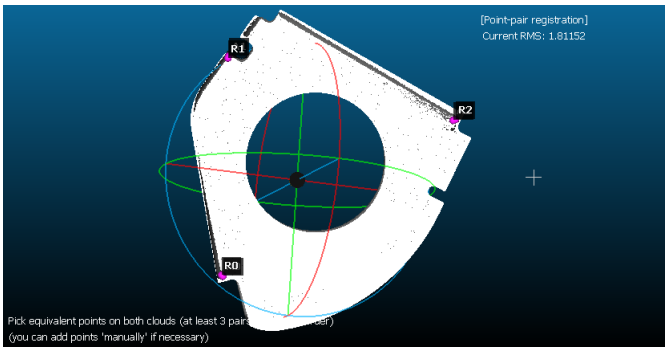
Lerrokatzeari ekiteko, lehenik eta behin, prozesu hori garatzeko beharrezkoak diren fitxategiak zabaldu behar ditugu. Alde batetik, 3D eskanerraren bitartez lortu dugun puntu-hodeia eta bestetik, berreraiki behar dugun objektuari dagokion modeloa. Esan bezala, hau sortu berri dugun puntu-hodeia edo hasieratik genuen triangelua izan daiteke, ikus 6.16 eta 6.17 irudiak.

Horiek lortuta, lerrokatzearen lehenengo fasea burutzeko puntu komunen aukeraketan oinarrituriko algoritmo bat aplikatu genuen. Horretarako, puntu-hodei bakoitzean gutxienez 3 puntu komun aukeratu beharko ditugu (ikus 6.18 irudia). Zehaztasun mailak ez du oso altua izan beharrik, izan ere modu horretan gerturapen bat soilik egingo dugu eta, ondoren, beste algoritmo bat erabiliko dugu gerturapen zehatzago bat egiteko.



6.18 Irudia: Puntu komunen aukeraketa

Hori egiteko, koordenatu-sistema ezberdinen arteko erlazioaren bilaketan oinarrituriko algoritmo bat erabiltzen du CLoud COmpare softwareak. Lerrokatzera goazen puntu-hodeiaren eraldaketa egiteko, metodo iteratiboak existitzen baldin badira ere, software honek Berthold K. P. Korn proposatutako algoritmo itxia erabiltzen du (Horn, 1987). Algoritmo horrek instalatu ditugun softwareen bitartez lorturiko puntu-hodeiek jasango duten translazio eta biraketa zuzenean kalkulatzen ditu, erreferentziatzat erabiliko dugun puntu-hodeira gerturatuz. Horretarako, aukeratutako puntuen zentroidea erabiltzen du, eta biraketak definitzeko koaternoiak erabiltzen ditu. Zentroideen arteko translazio eta biraketa erlazioekin hodeiak lerrokatzeko beharrezko matrizea kalkulatzen du, nahi izan ezkero eskuragarri izango duguna. 6.19 irudiak erakusten du prozesuaren emaitza



6.19 Irudia: Puntu hodeiak lerrokatuta

6.4.2 Lerrokatzea ICP-ren bitartez

Aipatu berri dugun bezala, lehen lerrokatzea antzeko puntuen aukeraketaren bitartez burutu genuen. Hala ere, kontuan eduki behar dugu puntuak esku hutsez aukeratzen ditugunez, hauen zehaztasuna ez dela guk eskuratu nahi duguna. Hori dela eta, lerrokatzea modu zehatzago batean aurrera eramateko, behin bi puntu-hodeiak aurreko metodoaren bitartez gerturatzen ditugunean, Cloud Compare softwareak eskaintzen digun *Iterative Closest Point* algoritmoa [7] aplikatuko dugu lerrokatzea gehiago zehazteko asmoz.

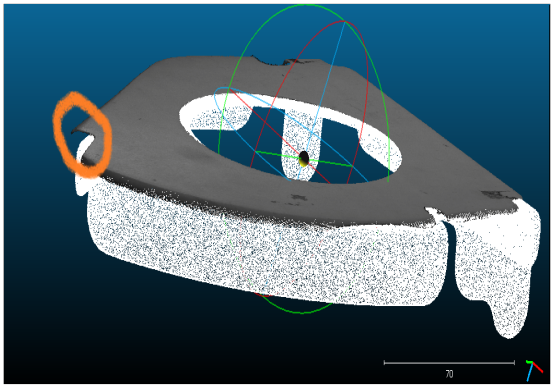
Algoritmo horrek, aurrekoaren moduan, erreferentziatzat gure objektuaren modeloa erabiliko du eta gure sistemaren bitartez lorturiko puntu-hodeiari eraldatze ezberdinak aplikatuko dizkio gehien gerturatzen dena aukeratu arte (ikus 6.20 eta 6.21 irudiak). Algoritmoak, iterazio bakoitzean, gerturatu nahi dugun puntu-hodei eta erreferentziaren arteko distantzia-errorea neurtuko du eta ondoren, translazio eta biraketa ezberdinak aplikatu emaitza hobetzen den aztertuz. Iterazio bakoitzean puntu-hodeieko puntu bakoitzarentzat ereferentzian gertuen dagoen puntua aurkitzen du. Une horretan, puntutik puntura dagoen distantziaren batezbesteko koadratikoa minimizatzen duen teknika bat erabiliko du. Batezbesteko koadratikoa (RMS) [8], batezbesteko aritmetikoaren modukoa da baina sarrera balioak bere horretan erabili beharrean, balio bakoitzaren balioa erabiltzen du. Hori horrela izanik, behin emaitza eskuratzean, horren erro karratua kalkulatuko berriro ere neurketa balio originaletara itzultzeko.

Algoritmo iteratiboetan ohikoa den moduan, gelditze-irizpide bat ezarri beharra daukagu exekuzioa momentu batean exekuzioa gelditu dadin. Horretarako, erabili dugun algoritmoak bi aukera eskaintzen dizkigu:

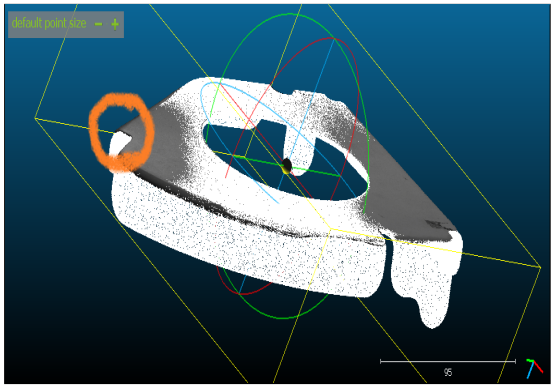
[7] www.mrpt.org/Iterative_Closest_Point_(IPC)_and_other_matching_algorithms
[8] http://mathworld.wolfram.com/Root-Mean-Square.htm

- Iterazio kopurua
- RMS balio bat zehatuz

Gure kasuan, erreferentzia eta sistemaren bitartez lorturiko puntu-hodeien arteko diferentzia zein den ez dakigunez, iterazio-kopuruan oinarrituriko gelditze-irizpide bat ezarri behar izan genuen. Gainera, software ezberdinen bitartez lorturiko puntu-hodeiak konparatu behar genituenez, guztiek izango zuten konputazio-balioa antzekoa izango zen, eta hori interesatzen zitzaigun. Horren ildora, lerrokatze bakoitzean 400 iterazio burutzea erabaki genuen.



6.20 Irudia: TI3D gray code



6.21 Irudia: DAVID 3D

Esan bezala, triangelu-sare sinple baten gainean lerrokatzea modu zehatzean burutzeko gai izango zen ez genekienez, lerrokatzea triangelu-sarearen gainean eta sortu berri genuen puntu-hodeiaren gainean aplikatu genuen. Honako hauek izan ziren algoritmoaren bukaeran lortu genituen RMS balioak, hau da, puntu-distantzien batezbesteko koadratikoak:

- Triangelu-sarearekin lerrokatuta: RMS = 0.332
- Guk sorturiko puntu-hodeiarekin lerrokatuta: RMS = 0.837

Lortutako emaitzekin ondoriozta genezakeen oinarrizko triangelu-sarearekin lerrokatzerako orduan distantziak zuzenean triangeluekin kalkulatzen zituela eta ez triangeluen erpinekin; beraz, hau zen metodorik zehatzena.

Beraz, modu horretan gauzatu genuen lerrokatzea software ezberdinekin lorturiko puntu-hodei bakoitzarekin, eta behin hori eginda, hurrengo pausuari ekin genion: distantziak kalkulatu. Aurreko fasea burutzeko softwareak distantziak kalkulatzen baldin baditu ere, horiek ez dira gordeta geratzen eta, beraz, esplizituki adierazi behar zaio hori egin dezan.

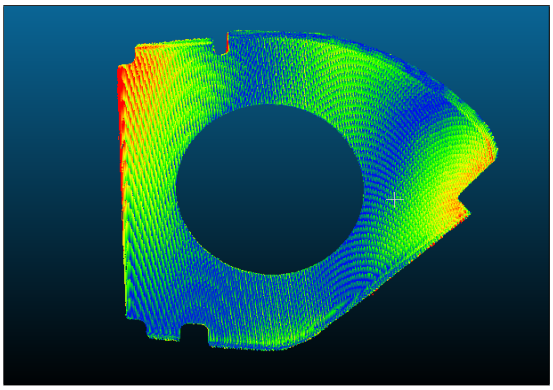
Distantziak kalkulatzerako orduan, puntu bakoitzarentzat gertuen dagoen triangelua bilatuko du, eta honekiko distantzia kalkulatuko du. Gure kasuan triangelu-sare eta puntuhodei baten arteko distantziak neurtzen ari garenez, gure neurketei zeinua jartzeko aukera dugu, triangeluaren normala erabiliz algoritmoak puntua triangeluaren zein aldetan dagoen jakin dezake. Hala ere, gure kasuan distantzia bera soilik axola zaigunez ez dugu aukera hori erabiliko.

Jarraian, erabilitako software ezberdinen bitartez lorturiko puntu-hodeien puntuek modeloarekiko dituzten distantziak aurki ditzakezue, baita horien analisi bat ere. Cloud Compare softwareak distantzien analisi bat burutzeko hainbat tresna eskaintzen dizkigu eta guk horien erabilera sustatu nahi izan dugu gure puntu hodeien kalitatea ikusi nahian. Distantziak koloreak erabiliz irudikatu ditugu ondorengo irudietan (ikus 6.22, 6.23, 6.24 eta 6.25 irudiak).

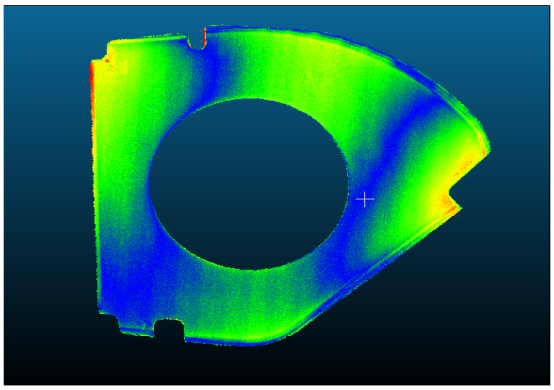
Irudietan erabili den kolore-eskala honakoa da:

Urdina < Berdea < Horia < Gorria

Urdin kolorea ikusiko dugu puntu horien erreferentziarekiko distantziatik 0tik hurbil dagoenean eta 1mm-tik gertu dagoenean berriz, kolore gorria ikusiko dugu.



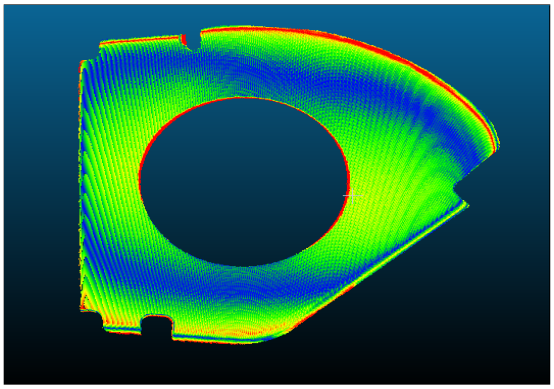
6.22 Irudia: SLStudio gray code



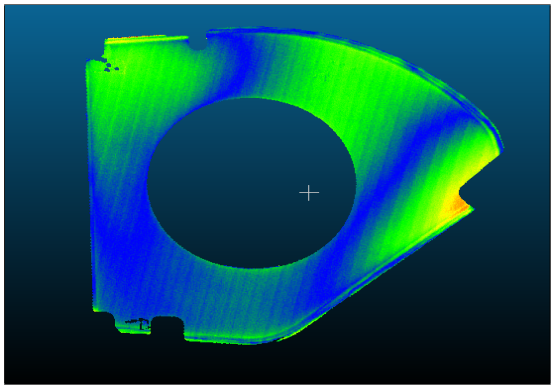
6.23 Irudia: SLStudio N Phase Shift

Horrez gain, konparazioari buruzko hainbat datu objektibo aurkitu ditzakezue 6.1 taulan.

Lerrokatzeen irudietan ikus dezakegun bezala (ikus 6.22, 6.23, 6.24 eta 6.25, software bakoitzaren bidez lorturiko berreraikitzeek emaitza ezberdinak eman dituzte. Ikus daitekeenez, zenbait softwarerekin lorturiko lerrokatzeetan puntu gorri asko ikus ditzakegu. Distantzia horien analisia burutzeko asmoz, bi grafiko sortu ditugu. Lehenengoa distan-



6.24 Irudia: TI3D



6.25 Irudia: DAVID 3D

	Puntu kop.	RMS	bataz. dist.	desb. est.
SLStudio gray code	686349	0.35	0.287	0.199
SLStudio N phase shift	407140	0.32	0.27	0.177
TI3D	386579	0.401	0.317	0.248
DAVID 3D	301657	0.249	0.205	0.143

6.1 Taula: Lerrokatzeari buruzko hainbat datu

tziak diskretizatuz egin dugu eta ondoren, ehunekoak erabili ditugu puntu kopuruak eraginik izan ez zezan.

6.26 irudiko grafikoan ikus daitekeen moduan puntu hodei bakoitzeko puntuek erreferentziarekiko lortu dituzten distantziak hainbat multzotan banatu ditugu. Informazio horren arabera, zati urdinak modelotik gertuen dauden puntuak irudikatzen du eta zati gorriak berriz urrutien dauden multzoak.

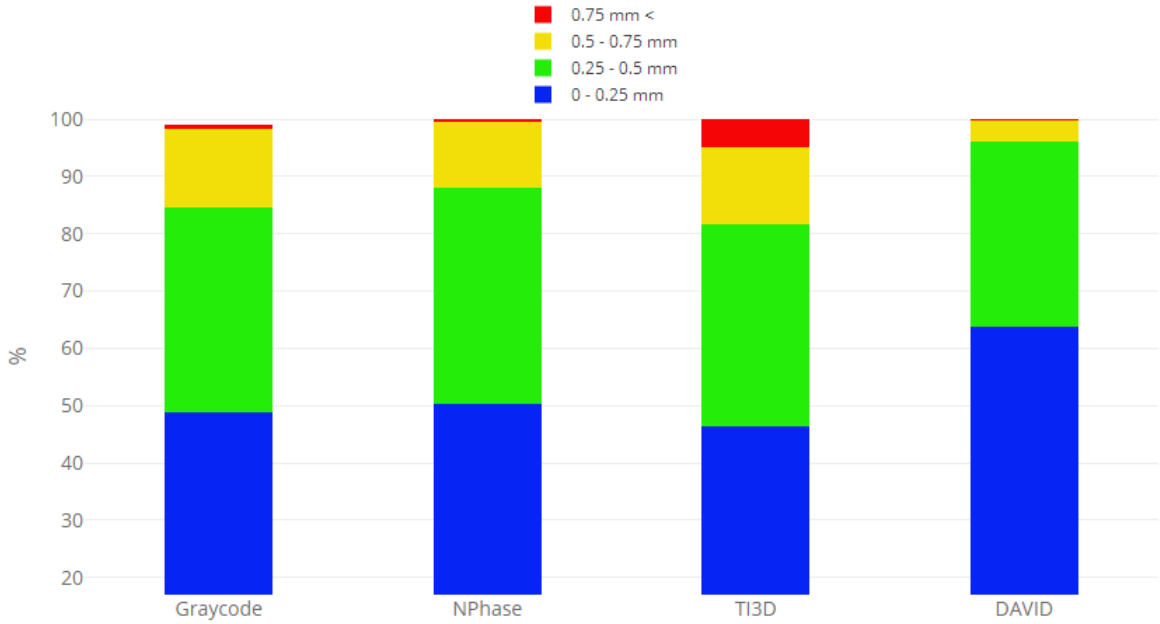
Distantzien distribuzioa modu jarrai batean ikusteko, distantzien dentsitatea neurtzen duen grafiko bat sortu dugu (ikus 6.27 irudia) R softwareak eskaintzen digun ggplot2 paketea erabiliz [9].

Dentsitate grafikoan, distantzia bakoitzak bere listan duen presentzia azaltzen digu. Hori dela eta, grafikoaren ezker aldean dentsitate altua izatea seinale ona litzateke. Lerro laranja bat ipini dugu grafikoan erreferentzia gisa, eta horrekin garbiago ikus dezakegu marraren eskuinaldean dauden distantziak gutxi direla DAVID 3D eta SLStudioren NPhase kodetzearen kasuan.

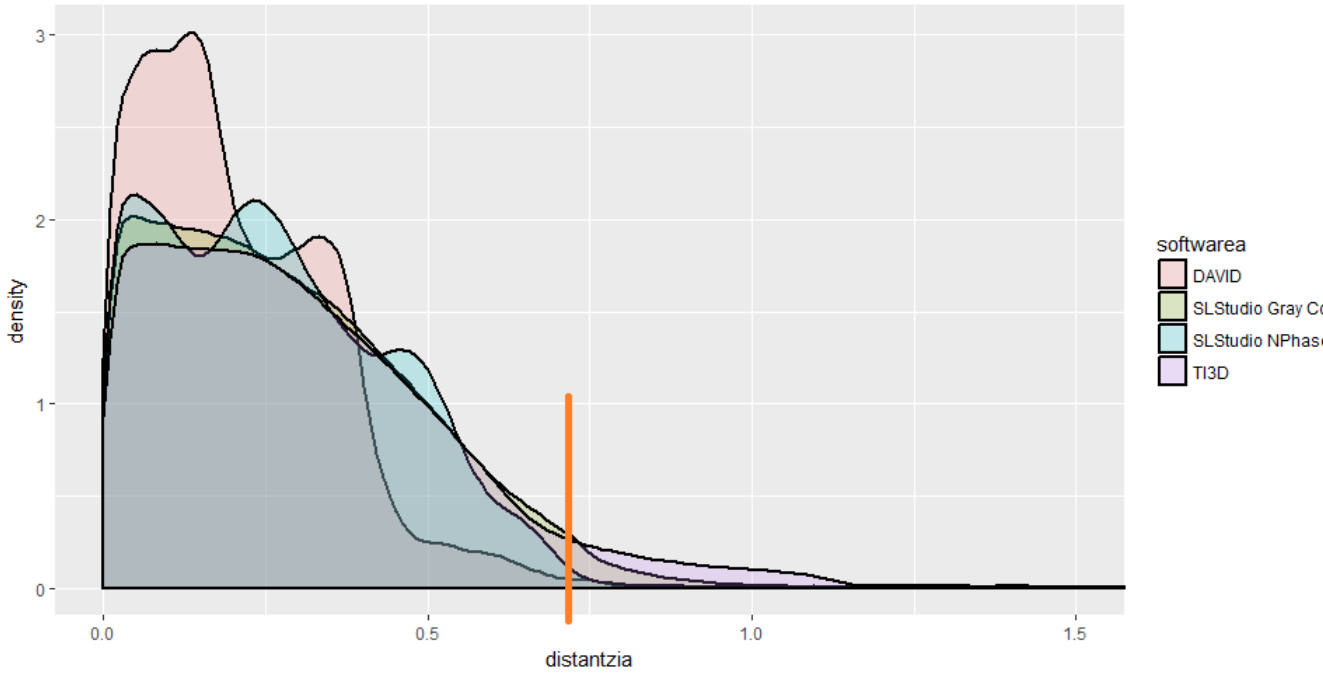
Ikusgai dituzuen grafiko eta datuek probatu ditugun sistemen bitartez lorturiko puntu-hodeiei buruzko hainbat ondorio ateratzeko aukera ematen digute.

Emaitza txarrenak eman dituzten puntu-hodeiak *gray code* metodoan oinarritzen diren

[9] http://ggplot2.org/



6.26 Irudia: Distantzia ezberdinen distribuzioa puntu-hodei bakoitzean



6.27 Irudia: Distantzien dentsitatearen distribuzioa puntu-hodei bakoitzean

softwareak izan dira; alde batetik, SLStudio-n *gray code* patroien bitartez lorturiko puntu-hodeia eta bestetik, TI3D softwarearen bitartez lorturikoa. Aipatutako lehenengoari dagokionez, batetik, berreraikitzeak leuntasun falta duela ikus dezakegu eta bestetik, bai eskuin eta baita ezker aldean puntuak modelotik urruntzen zaizkiola. Arazo hori kalibraziotik etor daitekeela pentsa dezakegu, izan ere, kamera eta proiektorearen posizioa modu zehatz batean definiturik ez badaude, puntu-hodei zilindrikoak lortzea nahiko ohikoa da.

TI3D softwarearen bitartez lorturiko puntu-hodeiaren azalera ere ez da nahi genukeen bezain leuna. Hori dela eta, teilakatze moduko bat dagoela ikus dezakegu, izan ere, altuera ezberdinetan dauden marra batzuk ikus ditzakegu. Horrez gain, berreraikitzearen goialdean begiratzen baldin badugu, gorri kolorez dagoela ikus dezakegu; eskalatze-arazo batetik dator, kalibrazio fasean aurre egiten saiatu ginena. Ikusi genuenez, kalibrazioan eskala ez zen guk nahi bezala egokitzen eta neurketak egiteko orduan zehaztasun falta zuten emaitzak lortu genituen. Hori dela eta, ahal bezain ondo egokitu bagenuen ere, modeloarekin distantziak neurtzean arazoak ematen dituela ikusi dugu, bilatzen ari garen zehaztasuna lortu gabe utziz.

SLStudio-ren NPhase kodetzea erabiliz lortu ditugu software libreen artean emaitza onenak. Azaleraren izaerari dagokionez, aurrez aztertu ditugun puntu-hodeiek ez duten leuntasun maila bat duela ikus dezakegu. Puntuek modeloarekiko duten distantziari dagokionez ere, emaitza onenak bertan lortu ditugu. Berreraikitzearen goiko aldean puntu gorri batzuk agertzen dira, baina dirudienez, posizio horietan ere puntu hodeiak nahiko itxura ona du eta beraz, errorearen arrazoietako bat lerrokatze-prozesuan egon daitekeela pentsatu dugu.

Espero zen moduan, emaitza onenak DAVID softwarearen bidez lorturikoak izan dira. DAVID-ek eskuratu ditu modeloarekiko distantzia laburrenak eta baita puntu gorri gutxien. Gainera, berreraikitako azalera leuntasun nahikoa duela ere ikus daiteke. Hala ere, proiektaturiko patroi kopurua kontuan izanik, NPhase kodeketaren bidez lorturiko emaitzak positiboak direla esango genuke.

6.5 Berreraikitze osoen arteko konparazioa

Sistemaren puntu-hodeien kalitatea neurtzerako orduan aurrerapauso handi bat eman dugu egin berri dugun prozesuarekin. Izan ere, kalitate-kontrola egiteko erabiltzen den teknika erabiliz guk lorturiko puntu-hodeiek berezko modeloarekin duten desberdintasuna

ikusteko gai izan gara. Hala ere, informazio baliagarri gehiago lortzeko, berreraikitze osoen arteko konparazioa ere egin dugu.

Zenbaitetan, ez dugu guregan izango berreraiki nahi dugun objektuaren modeloa, eta gure helburua aurrean daukagun objektua digitalizatzea izango da. Digitalizaziotik abiatuz, modeloa garatzeko ahalmena izango dugu eta baita, hiru dimentsiotan objektu hori inprimatzekoa ere. Hori dela eta, oraingo honetan, puntu-hodeiak lortzeko erabili ditugun softwareek aipatu berri dugun prozesuan zer nolako errendimendua duten aztertuko dugu.

Horretarako, aurrez erabili dugun quesito pieza erabiliko dugu, baina kasu honetan bere berreraikitze osoa burutzen ahaleginduko gara. Bista ezberdinetatik puntu-hodeiak atzemango ditugu azalera guztiari buruzko informazioa eduki arte. Zehazki, software bakoitzaren bitartez 6 puntu-hodei eskuratu ditugu.

Behin horiek lorturik, objektu osoaren berreraikitzea eraikitzeari ekingo diogu. Horretarako, lerrokatzerako erabili dugun ICP algoritmo berbera erabiliko dugu, baina oraingoan jarraian azalduta dagoen aldaerarekin:

- Kontaktuan dauden bi puntu-hodei zabalduko ditugu, biak ere modu egokian segmentaturik, eta gutxienez zati bat komuna dutelarik.
- Lerrokatzeetan egin ohi dugun lehen fasean bezalaxe, puntu aukeraketaren, edo eskuz buruturiko eraldaketa baten bitartez, bi puntu-hodeiak elkarren ondoan jarriko ditugu.
- Bi puntu-hodeiek duten zati komuna identifikatuko dugu eta bi puntu-hodeietako batean segmentazio bidez zati komuna soilik moztuko dugu.
- ICP algoritmoaren bitartez, ebaki berri dugun puntu-hodei zatia erreferentziatzat dugun puntu-hodeiarekin lerrokatuko dugu.
- Algoritmoak sortuko duen eraldatze-matrizea hartu eta mugitu gabe utzi dugun puntu-hodei zatiari aplikatuko diogu, aurrez moztu dugun zatiarekin berriro lotuz.
- Bi puntu-hodeiak bakar batean elkartuko ditugu.

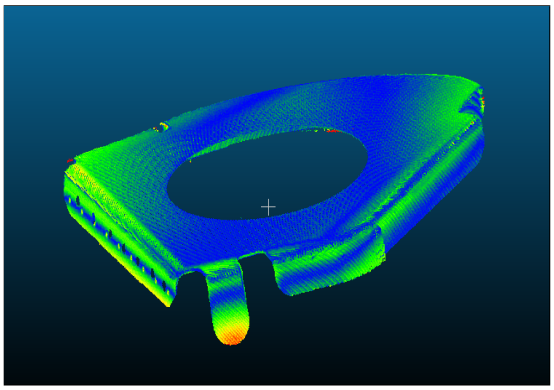
Behin software bakoitzarekin lorturiko berreraikitze osoa daukagunean, aurreko fasean egin dugunaren moduko konparazio bat gauzatuko dugu 3D modeloarekin, oraingo honetan konparatzen ari garen puntu-hodeia objektu osoa errepresentatzen duena delarik. Hala ere, fase honetan lerrokatze ugari ematen direnez, kontuan izan behar dugu gure emaitzan

egongo den errorea handiagoa izango dela eta, beraz, informazioa ez dela aurreko proban bezain zehatza izango.

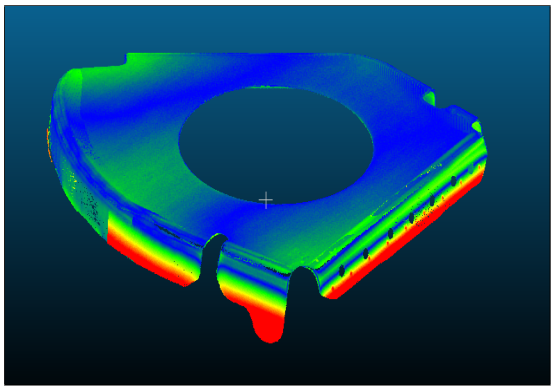
Jarraian dituzue esperimentuan lortu ditugun emaitzak. Esan bezala, lerrokatze hauek egiterako orduan erroreak egon daitezke, izan ere lehen lerrokatzeak egiteko aukeratzen ditugun zatien baitan egongo da gure emaitzaren zati handi bat. Hala ere, gehienetan lorturiko puntu-hodeiak onak baldin badira, asko errazten du lerrokatze lana, eta hori ere, aztertu nahi izan dugun ezaugarrietako bat izan da. Oraingo honetan ere erabili dugun kolore eskala aurrekoaren berbera izan da:

Urdina < **Berdea** < **Horia** < **Gorria**

Baina kasu honetan berreraikitako puntu-hodeiaren eta modeloaren arteko distantziak handiagoak izan direnez, puntu gorriek 2.4mm-tara dauden distantziak irudikatuko dituzte. Lerrokatze hauek irudi bakar batean erakusten zailagoa denez, berreraikitze bakoitzarentzat errore gehien dituen aldea erakutsiko dugu, hau da, puntu gorri gehien dituen aldea.



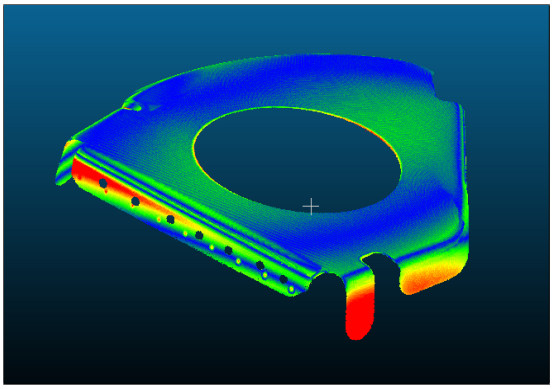
6.28 Irudia: SLStudio gray code



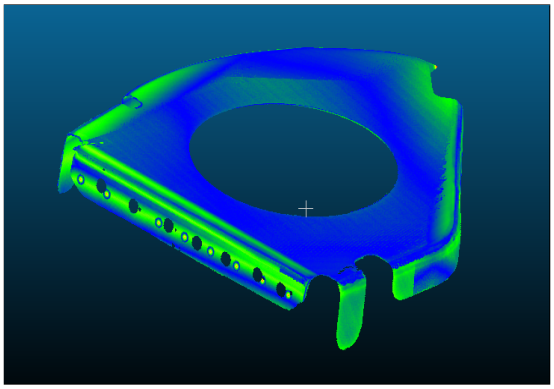
6.29 Irudia: SLStudio N Phase Shift

Irudietan (ikus 6.28, 6.29, 6.30 eta 6.31 irudiak) hainbat datu estatistiko atera ditugu, emaitzak ziurtatzen direla bermatzeko. 6.2 taulan adierazirik geratzen dira emaitza horiek. Kasu honetan, berreraikitzeen zatiak eskuz ebaki ditugunez, ez zaigu beharrezkoa iruditu puntu kopurua datu esanguratsutzat jartzea; ezta RMS balioa ere, izan ere, kasu bakoitzean 6 puntu-hodei lerrokatu behar izan ditugu eta RMS balio ezberdinak lortu ditugu:

Berriro ere, aurrez burutu dugun prozedura jarraituz, eta distantzien distribuzio hobeto



6.30 Irudia: TI3D



6.31 Irudia: DAVID 3D

	bataz. dist.	desb. est.
SLStudio gray code	0.367	0.272
SLStudio N phase shift	0.41	0.552
TI3D	0.465	0.473
DAVID 3D	0.343	0.271

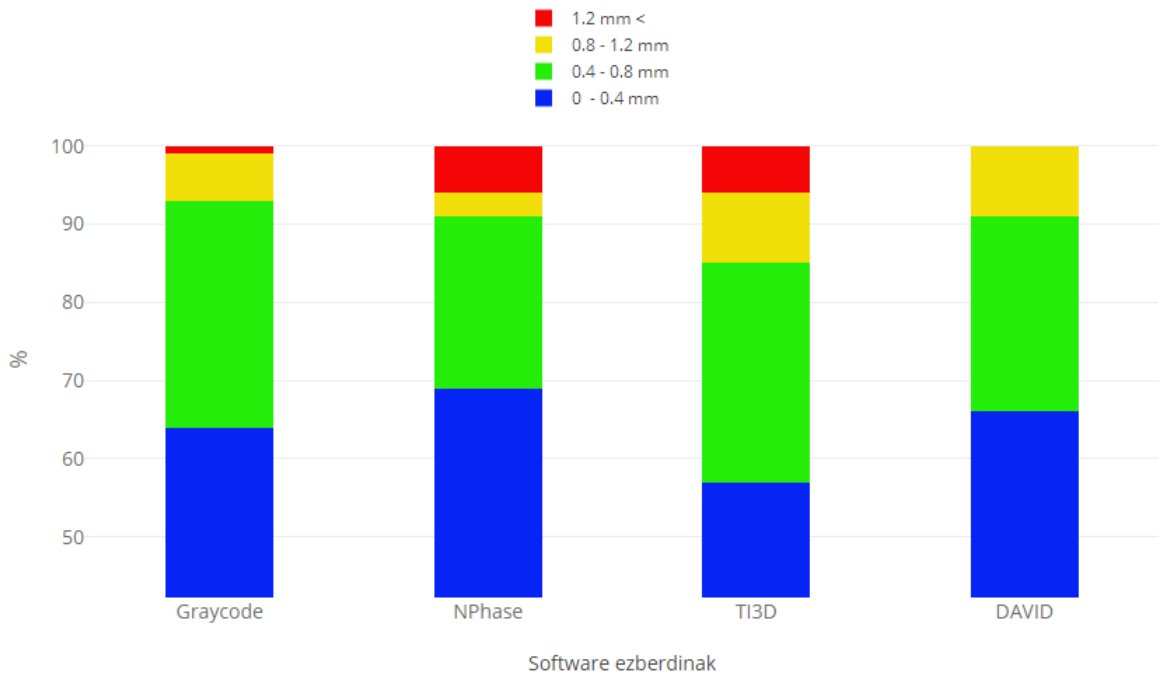
6.2 Taula: Bataz besteko distantzia eta desbiazio estandarra

ikus dezagun, taula diagrama bat sortu dugu distantziak diskretizatuz eta ehunekoak ateraz (ikus 6.32 irudia):

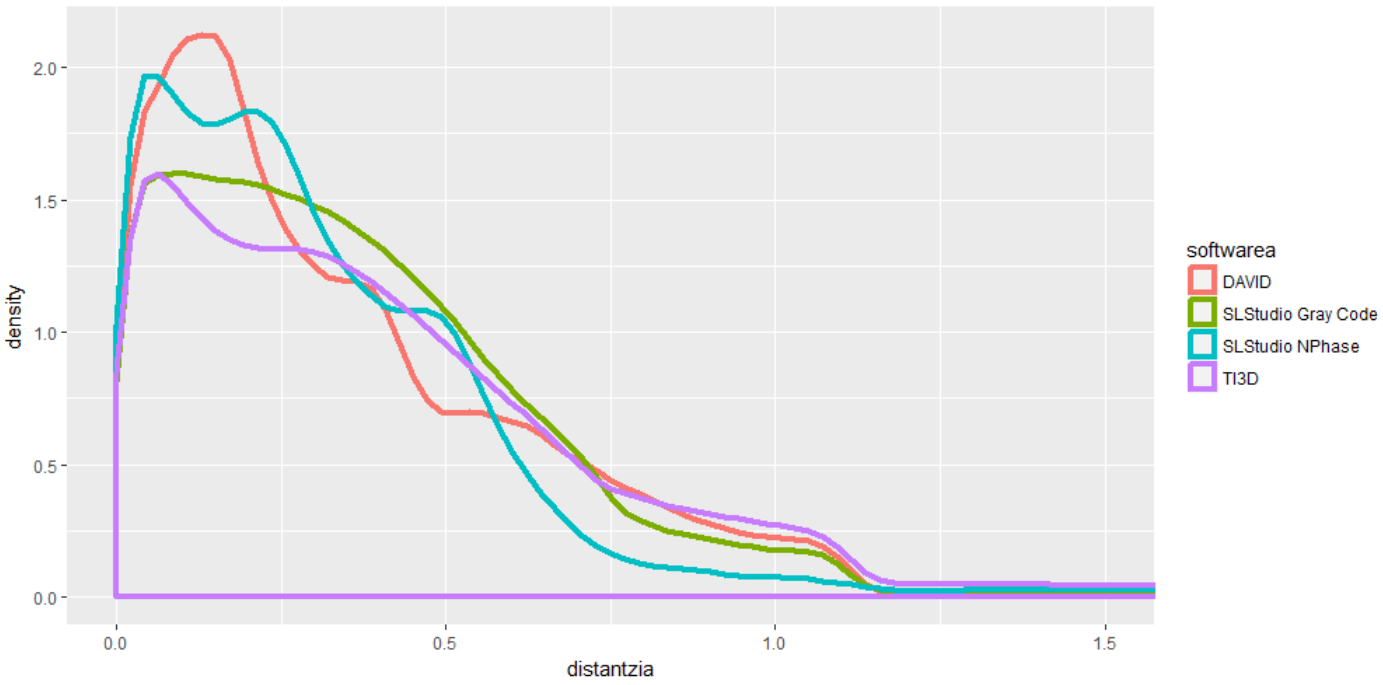
Grafikoan ikus daitekeenez, 0.4 mm baina gutxiagora proportzioan distantzia gehien dituena SLStudio da N Phase Shift kodeketarekin lortutakoak dira, kalitate handia lortzeko gai dela erakutsiz. Hala ere, aurrera joan ahala, lehen aipatutakoa bermatzen da: zenbait guneetan lorturiko lerrokatzea txarra izan dela, eta hori aztertu beharreko zerbait izango da. Oraingoan, SLStudio-ren gray code kodetzeak egonkortasuna erakutsi du, 1.2 mm baina distantzia handiagora oso puntu gutxi baititu. Informazio gehiago izateko, berriro ere distantzien dentsitateak azaltzen dituen grafiko bat sortu dugu (ikus 6.33 irudia). Distantzia balio batzuk handiak eta murritzak zirenez, 0-1.5 tartea soilik erakutsi dugu, hasiera batean behintzat.

Bestalde, esan beharra daukagu, grafiko hau ikusiz aurrez lortu ditugun emaitza batzuk kolokan jarri direla, izan ere, hasiera batean behintzat, emaitza onenak ematen dituen softwarea NPhase dela dirudi. Guk, hala ere, aurreko irudietan (ikus 6.28, 6.29, 6.30 eta 6.31 irudiak) ikusi berri genuen hori ez zela posible. Hori argitze aldera, gerturapen bat eta x koordenatuaren berreskalatze bat burutu genuen 6.34 irudia lortuz:

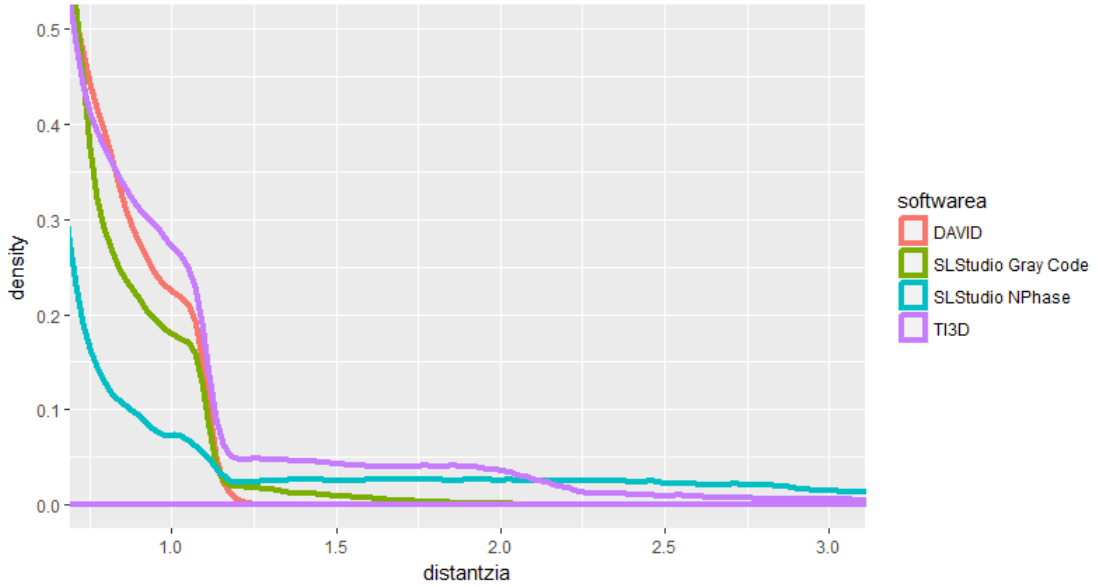
Bertan ikus daitekeenez, SLStudio-ren *N Phase Shift* aukerarekin eta TI3D softwrearekin lorturiko datuak ez dute beste softwareek lortutakoak bezala konbergitzen eta aurrera



6.32 Irudia: Puntu-hodei bakoitzean distantzia tarte ezberdinen distribuzioa



6.33 Irudia: Distantzia tarte ezberdinen distribuzioa puntu-hodei bakoitzean



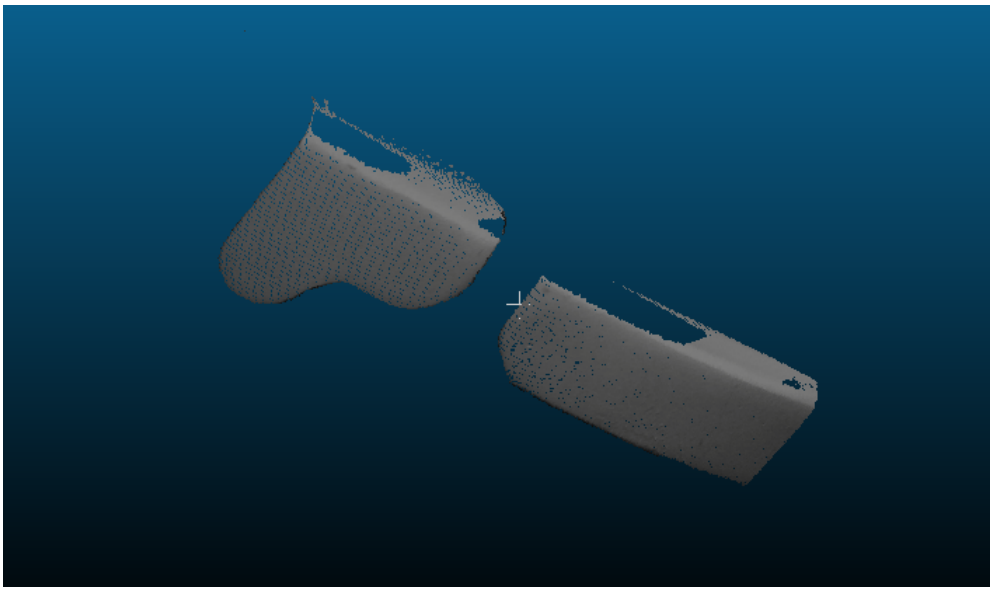
6.34 Irudia: Distantzia tarte ezberdinen distribuzioa puntu-hodei bakoitzean

jarraitzen dute, bertan kontuan hartzeko moduko distantziak daudela erakutsiz. Bestetik, nabarmendu beharra daukagu SLStudioren *gray code* aukerak eta DAVID softwareak lortu dituzten antzeko balioak.

Konparazioak software zehatzenaren lekuan, berriro ere, DAVID softwarea ezarri baldin badu ere, emaitza ezberdinak eman dizkigu SLStudiok erabiltzen dituen patroien artean. Berriro ere, emaitza txarrenak TI3D softwareak eman dizkigula esan dezakegu. Eskalatze-prozesua etortzen zaigu burura, beste behin ere, lehen aipatu dugun bezala, arazoak izan genituen softwarea eskalatzerako orduan. Orain ikusten da objektu osoaren berreraikitzea konparatzean arazo hori handitu egiten dela.

Aurrez lortu genituen emaitzetatik gehien aldatu dena SLStudio softwareko NPhase kodetzea izan da. Kasu horretan, hodeiaren leuntasun eta neurriak nahiko onak izaten jarraitzen dutela ematen badu ere, irudian (ikus 6.29 irudia) garbi ikus genezake atal batzuetan duen zehaztasun maila oso eskasa dela. Lortu dugun desbiderapen estandarrak ere hori adierazten digu, izan ere, batezbesteko distantzia TI3D softwareak baina hobea baldin badu ere, emaitza guztien arten artean desbiderapen estandar altuena duena dela ikus dezakegu. Hori bere puntuen sakabanatzetik dator, garbi ikus daitekeelako puntu-hodeiaren atal jakin bat dela emaitza txarrak ematen dituena. Hori aztertzeko asmoz, berreraikitzea zatiz zati aztertu eta zenbait puntu-hodeietan informazio falta dagoela ikusi dugu.

6.35 Irudian ikus dezakezuen puntu-hodeia lortzeko prozesuan, eskaneatu beharreko pie-



6.35 Irudia: Distantzia tarte ezberdinen distribuzioa puntu-hodei bakoitzean

zak kamerarekiko zuen angelua ez zen goitiko bistan eskaneatu genuenean izan genuen angelua bezain ona, eta dirudienez NPhase teknikak proiektatzen dituen patroiek ez dute modu egokian lan egiten egoera honen aurrean. Informazio falta horrek lerrokatzea burutzerako orduan hainbat arazo eman zizkigun eta hortik, konparazioan azaldu zaigun errorea etorri da. Hala ere, arazo hori ekiditeko lortuko bagenu, lorturiko emaitzak SLStudio-ren *gray code* patroiak erabiltzean baina hobeak izatea esperoko litzateke, hau da DAVID softwarearekin lorturiko emaitzetatik gertu egongo litzateke.

SLStudio-k erabilitako *gray code* kodetzea izan da software libreen artean emaitza onenak lortu dituena. Puntu-hodei egokiak lortu dituela dirudi objektuaren alde guztietan eta distantzietan lorturiko erroreak txikiak izan dira kode irekiko beste bi softwareekiko. 6.28 irudian ikus dezakegunez, eta desbiderapen estandarrak erakusten digunez, erroreak nahiko homogeneoak izan dira, zati txiki batean soilik ikus ditzakegu puntu gorriak.

Proba honetan ere, emaitza onenak lortu dituen softwarea DAVID izan da. Oraingoz lortutako emaitzak ikusirik fidagarritasun handia eskaintzen duela ikus genezake eta berreraikitze onenak egiteko gai dela. Hala ere, SLStudio-ren bitartez lorturiko puntu-hodeiak horien kalitatera hurbil daitezkeela ikusi dugu, nahiz eta software libre honek erabiltzen duen patroi kopurua txikiagoa izan.

	Puntu kopurua
SLStudio gray code	342.038
SLStudio N phase shift	335.472
TI3D	311.174
DAVID 3D	297.247

6.3 Taula: Puntu-hodei bakoitzaren puntu kopurua

6.6 Puntu-hodeien analisi zuzena

Aurrez egin ditugun probek informazio nahikoa ematen baldin badigute ere, puntu-hodeien konparazioari amaiera emateko, azken analisi bat egingo dugu. Oraingo honetan, software bakoitzaren bitartez neurri jakinak dituen pieza leun eta lau bat berreraiki genuen. Behin hori eginda, aurrez aipatu dugun aurreprozesamenduaren bitartez puntu-hodeiari zarata kendu eta piezaren goiko atala soilik isolatu genuen. Orduan, software guztietan pieza berberaren goiko aldeari analisi ezberdinak apliatu genizkion: dentsitatea, kurbadura eta neurketa besteak beste. Lehenik eta behin, piezaren goiko aldea irudikatzen duen puntu-hodei bakoitzaren puntu kopurua adierazita dago (ikus 6.3 taula).

6.6.1 Dentsitatea

Puntu-hodeiaren xehetasun gehiago aztertzeko asmotan dentsitatea neurtuko dugu. Baina, kasu honetan dentsitateari buruz hitz egiten dugunean ez gara hodei osoaren dentsitateari buruz ari, izan ere tamaina berdineko hodeiak berreraikitzen ari garenez, puntu kopuruak nahikoa informazio ematen digu horri buruz.

Oraingo honetan, puntu bakoitzaren dentsitatea zein den neurtuko dugu hodeitan berezitasunik dagoen ikusteko. Hori egiteko ere Cloud Compare softwareak eskainitako tresna bat erabiliko. Berreraikitzeko puntu bakoitzaren inguruan guk ezarritako erradioa duen esfera bat irudikatuko duena eta bertan aurkitzen diren puntuak zenbatuko dituena. Kontaketaren bitartez bitartez puntu bakoitzaren dentsitatea kalkulatuko du.

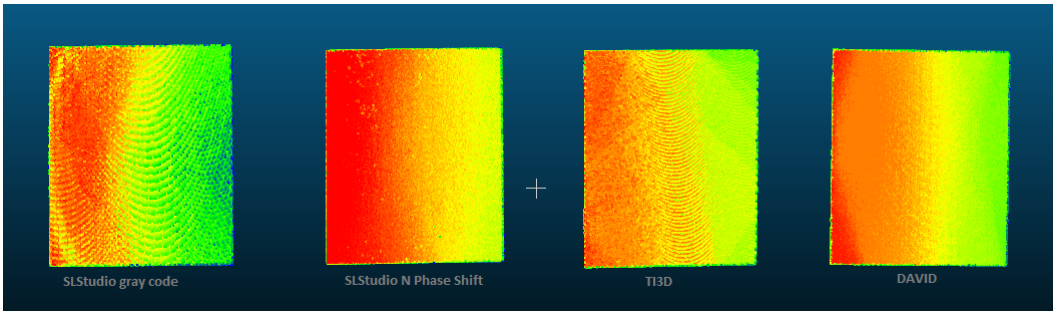
Dentsitatea kalkulatzerako orduan bi modu ezberdin proposatzen dizkigu erabilitako algoritmoak:

- Bolumen dentsitatea: Esferaren barnean dauden bizilagunak kontatzean (N), dentsitatea kalkulatzeko balio hori esferaren bolumenarekin zatituko du = $N/(4/3.\pi.R^3)$

- Azalera dentsitatea: Esferaren barnean dauden bizilagunak kontatzean (N), dentsitatea kalkulatzeko balio hori esferaren azalerarekin zatituko du = $N/(\pi.R^2)$

Azken finean, gure asmoa dentsitateak kalkulatzea denez, gure kasuan aukeraketak ez du garrantzirik, konparaketa burutzerako orduan dentsitatearen balioa ezberdina izan da, gure puntu-hodeien dentsitatean egongo den aldea berdina izango baita.

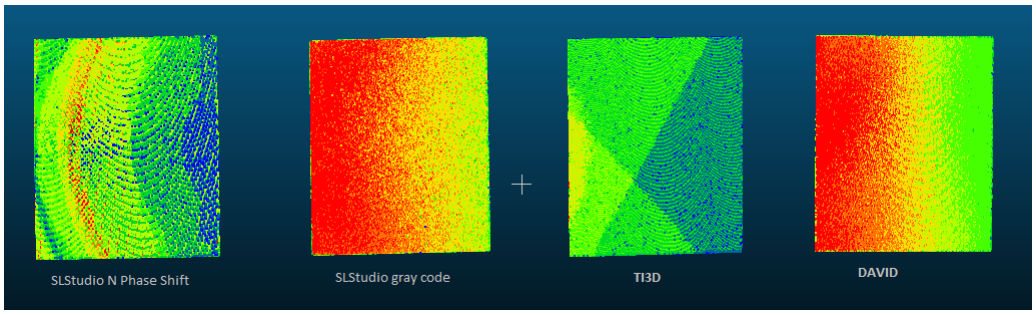
Esan bezala, dentsitatea kalkulatzeko moduak ez du eragin handirik lorturiko emaitzan, baina dentsitatea kalkulatzeko erabiliko dugun esferaren tamainak ordea eragin zuzena da. Izan ere, jakin badakigu puntu-hodei guztiek puntu-kopuru antzekoa dutela eta beraz, hodei osoaren dentsitatea ere antzekoa izango dela. Aipaturiko esferaren bitartez, puntuen kontzentrazioak aztertzeko aukera izan dezakegu, eta modu horretan puntu-hodeiaren homogeneotasuna aztertu. Proba hauetan ere erabili dugun kolore-eskala aurreko probetan erabilitakoaren berbera izan da.



6.36 Irudia: Distantzia tarte ezberdinen distribuzioa puntu-hodei bakoitzean

6.36 irudian 0.9 mm-ko esferak egin ditugu puntu bakoitzaren inguruan dentsitatea kalkulatzeko. Esfera horien tamaina erlatiboki nahiko handia denez, puntu bakoitzaren inguruko puntu kopuru handia sartzen da bertan eta modu horretan ez da ondo bereizten dentsitate aldaketa. KOntuan izan behar irudi hauetan kameratik gertuen dagoen aldea ezkerraldea dela. Hori dela eta, ezkerraldeko puntuek dentsitate handiagoa dutela ikus dezakegu, baina, hori esperotako zerbait da. Hala ere, SLStudioren gray code kodetzean ikus dezakegu piezaren eskuinaldean duen dentsitatea beste piezena baño baxuagoa dela.

6.37 irudian, dentsitatea sakonago aztertzeko aztertzeko, hau kalkulatzeko erabilitako esferen erradioa 0.4 mm-ra jaitsi dugu. Modu horretan, azalera homogeneorik lortzen ez duten teknikek arazo handiagoa izango dute. Eta hori ziurtatzen dute lortutako datuek, izan ere, bai DAVID eta baita SLStudioren N Phase Shift kodetzeak ere esperotako dentsitatea lortzen dute, ezkerraldetik eskuinaldera progresiboki txikituz. TI3D eta SLStudioren gray code tekniketan berriz, lorturiko azalera ez dela hain homogeneoa ikus dezakegu,



6.37 Irudia: Distantzia tarte ezberdinen distribuzioa puntu-hodei bakoitzean

6.4 Taula: Kurbaduraren azterketan lorturiko balioak

	Altuera (mm)	Zabalera (mm)
SLStudio gray code	170'5 mm	141'5 mm
SLStudio N phase shift	166'4 mm	138'2 mm
TI3D	169'5 mm	136'2 mm
DAVID 3D	167'3 mm	138'2 mm

hainbat puntutan dentsitate baxuak lortzen baitituzte, batez ere kameratik urruti dauden puntuetan.

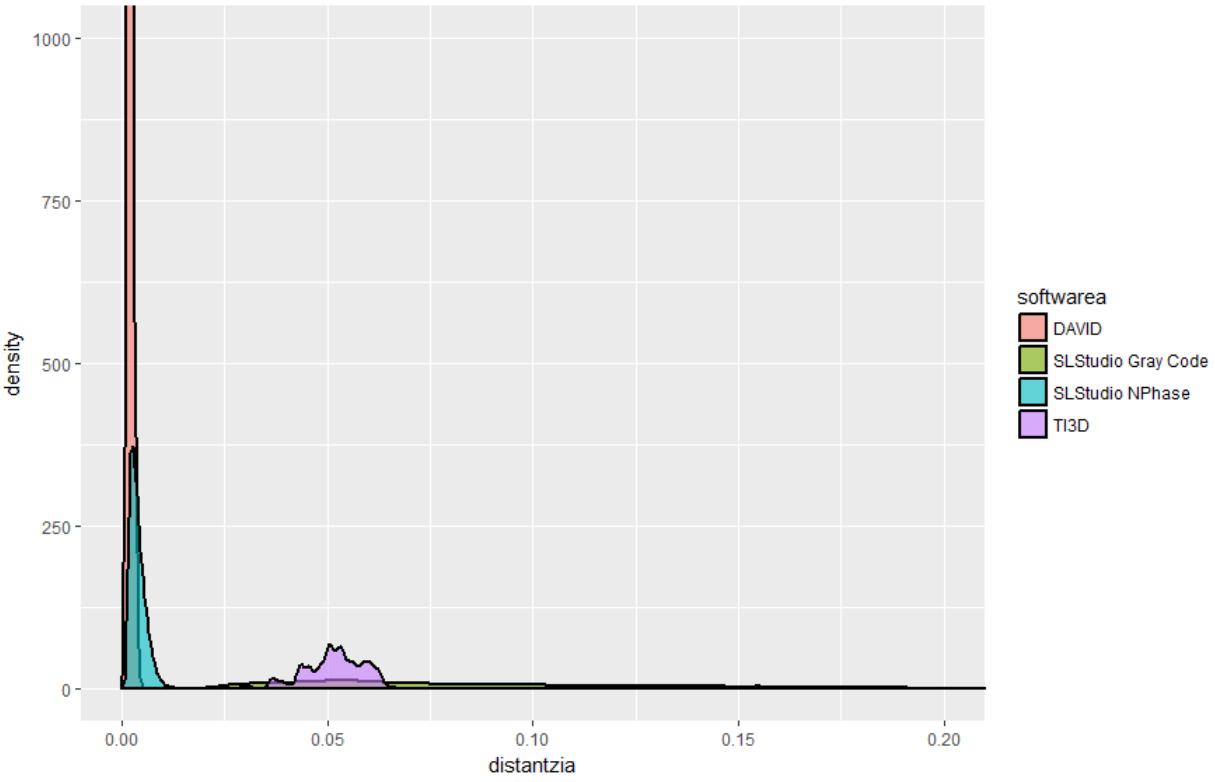
6.6.2 Kurbadura

Analisi honen bitartez puntu-hodeian kurbadurak atzematen ahaleginduko gara, gure puntu-hodeien leuntasuna aztertzeko. Erabilitako algoritmoak, puntu bakoitzak ingurukoekin osatzen koadrika leunena aztertzen du kurbadura estimatu ahal izateko. Hortik abiatuz, aukera ezberdinak proposatzen bazaizkigu ere, normalak erabiltzen ditu kurbadura kalkulatzeko.

Emaitzen artean ezberdintasun nabarmenak daudenez, eskala berdinarekin irudikatuz gero ez dira lorturiko ezberdintasunak argi ikusten. Hori dela eta, emaitzak taula batean soilik erakutsiko ditugu oraingoan (ikus 6.4) . Maximoa kalkulatzeko unean *outlier*-ak kendu ditut, hau da, datuen izaera errespetatzen ez duten datu solteak.

Hala ere, emaitzen nondik norakoa modu bisualago batean ikusteko, dentsitate grafiko bat burutu dugu, eta 6.38 irudian ikus dezakezue.

Ikus dezakegunez bi multzo berei ditzakegu. Alde batetik, DAVID eta SLStudioren N Phase Shift kodetzearekin lorturiko emaitzak ditugu. DAVID-enak hobeak baldin badira ere, software libreak gertutik jarraitzen duela ikusi dugu.

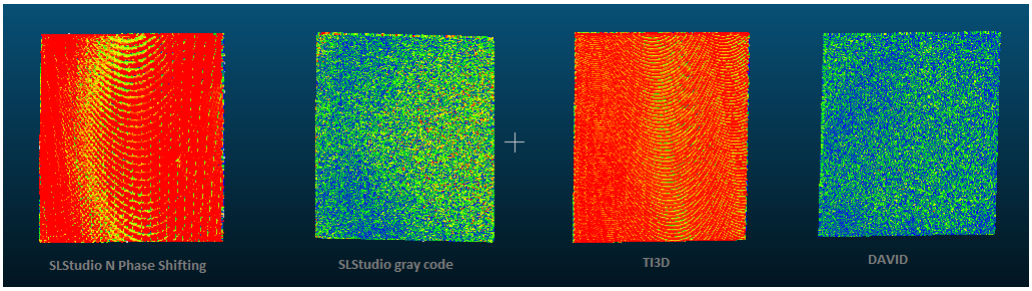


6.38 Irudia: Kurbaduraren azterketan lorturiko balioen dentsitatea

Bestetik, TI3D eta SLStudioren gray code kodeketaren bitartez lorturiko datuak ditugu. Bien emaitzak eskasak baldin badira ere, atentzioa deitzen du SLStudiok kasu honetan duen kurbadurak. Ikusten ari garenez, osotasunean bere berreraikitzeak itxura ona badute ere, hodeiak funtsean dituen zehaztasun-galerak erakusten ari da.

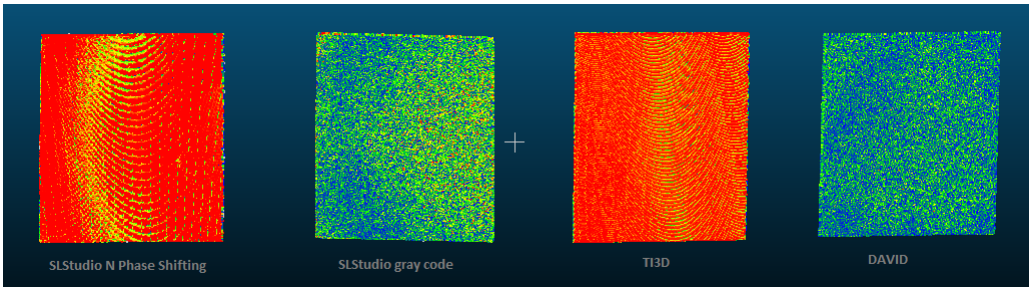
6.6.3 Zimurdura

Proba hau aurrekoaren oso antzekoa da eta hori dela eta lorturiko emaitzak ere berdintsuak izan dira modu moderatuagoan baldin bada ere. Kasu honetan, algoritmoak, inguruko puntuekin koadrikoak bat sortu beharrean, planoak sortuko ditu. Ondoren, puntuen normalak planoarenarekin konparatu, eta desberdintasun txikiena duena aukeratuko du, zimurdura balioa kalkulatuz. Hemen dituzue aurreko proba berresten duten irudiak (ikus 6.40 irudia).



6.39 Irudia: Zimurduraren azterketan lorturiko irudiak

Bertan, nabarmendurik geratzen dira aurreko grafikoetan lorturiko emaitzak. Hala ere, kurbaduraren azterketan egon diren aldeak hauen aldean handiak izan direla garbi gera dadin 6.38 irudiko grafikoarekin konparatzeko, 6.40 irudian ikus dezakezue zimurduraren dentsitate grafikoa



6.40 Irudia: Zimurduraren azterketan lorturiko balioen dentsitatea

	Altuera (mm)	Zabalera (mm)
SLStudio gray code	170’5 mm	141’5 mm
SLStudio N phase shift	166’4 mm	138’2 mm
TI3D	169’5 mm	136’2 mm
DAVID 3D	167’3 mm	138’2 mm

6.5 Taula: Neurketan lorturiko datuak

6.6.4 Neurketa zehaztasuna

Azken analisi honetan berreraiki dugun piezaren neurriak neurtuko ditugu. Piezaren neurriak aurrez ezagunak, eta neurketa hori bera egingo dugu konparatzen ari garen softwareen bidez lorturiko puntu-hodeietan.

Neurketa egiteko, Cloud Compare-ek bi puntu hautatzeko aukera ematen digu eta bi puntu horien arteko distantzia automatikoki kalkulatzen du. Gure piezaren neurriak honakoak dira: 168 mm x 140 mm. Softwareen bitartez eginiko neurketeta 6.5 taulan ikusi ditzakezue.

Lorturiko balioak ikusiz, berriro ere, balio egonkorrenak DAVID softwareak lortzen dituela ikus dezakegu, baina SLStudio softwarearen bi patroien bitartez lorturiko balioak kontuan hartzekoak direla esan genezake.

TI3D-ren kasuan, beste behin, agerian geratzen dira jada aipatu ditugun eskalatze-arazoak, batez ere objektuaren zabalera neurtzerakoan. Ezagutzen ez dugun arrazoi batengatik, neurketa bertikalki ongi burutzen baldin badu ere, horizontalki, kontuan hartzeko errore bat dagoela esan dezakegu.

Bestetik, altuera neurtzerakoan hiru software libreak balio antzekoak lortzen dituztela ikusi dugu, hiru kasuetan 1’5 mm inguruko errorea dagoela ikusi dugu.

6.7 Konparazioaren ondorioak

Proba bakoitzaren azterketaren ondoren, emaitzak nahiko garbi gelditu direla ikus badezakegu ere, hauen laburpen bat egingo dugu ondorioak garbi gera daitezen. Ikusi dugun bezala, orokorrean emaitza txarrenak lortu dituen softwarea TI3D softwarea izan da. Zarata gutxi duten berreraikitzean lortu baditu ere, emaitzak modu egokian eskalatzea lortu ez dugunez, lerrokatzeen probetan ez ditu emaitza onak eman. Gainera, puntu-hodeiaren leuntasuna aztertu dugunean ere kalitate falta duela erakutsi du. Horrez gain, kontuan

izan behar dugu 20 patroi proiektatzen dituela eta beraz, berreraikitze denbora nahiko altua izango dela.

SLStudioren kasuan *gray code* kodetzea erabiltzen duenean, bere hodeian leuntasun falta handia erakutsi du. Espero genuen bezala, teknika honetan oinarritzen diren berreraikitzeek, leuntasun mailan, beste teknikek erakutsi ez dituzten galera batzuk erakutsi dituzte. Hala ere, esan beharra dago, berreraikitze osoaren konparazioan uste baina emaitza hobeak eman dituela, pieza posizio ezberdinetan jarririk ere, kamerak patroien informazioa ondo jasotzen duela erakutsiz.

SLStudioren *N Phase Shift* metodoak berriz espero baina emaitza hobeak eman ditu. DAVID sistemaren emaitzak hobetzeko gai izan ez baldin bada ere, proba gehienetan gertuen ibili den teknika izan da. Hala ere, informazio galerak erakutsi dite berreraikitze osoaren proban, angeluaren arabera lorturiko emaitzak okertzen direla erakutsiz. Arazo hau konpondu beharra baldin badago ere, emaitzak orokorrean onak izan dira, eta 11 patroi soilik erabiltzen dituela jakinda, etorkizunean kontuan edukiko dugun softwarea izango da.

Espero zen moduan, DAVID izan da emaitza onenak lortu dituen sistema. Proiektaturiko patroiak gehiago baldin badira ere, proba guztietan beste sistemen gainetik agertu zaigu. Hau ikusirik, etorkizunean gure bilaketan aurrera jarraitu beharko dugu, kodetze berri eta konplexuagoak bilatuz edo guk geuk sortzen saiatuz. Horretarako, konbinazio batzuk sortzen saiatu gaitezke, *N Phase Shift* eta *gray code* tekniken artean adibidez.

Lorturiko datuak horiek izanda ere, esan beharra dugu SLStudio modu egokian konfiguratuko bagenu, bere berreraikitze-abiadura DAVID sistemarena baina altuagoa izango litzatekeela eta beraz, kalitate-kontrolerako egokia ez baldin bada ere, beste erabilpen batzuetarako erabil dezakegula. Beste zenbait lanetan erabiltzeko egokia izan daiteke software hori: piezen identifikazio eta lokalizaziorako, robotaren segurtasun mekanismoak gauzatzeko edo roboten nabigazioan erabiltzeko.

7. KAPITULUA

Ondorioak eta etorkizuneko lanak

Atal honetan, egindako lanari buruzko laburpena, proiektuan zehar ateratako ondorioak eta etorkizunean aurreikusten diren aukerei buruz hitz egingo da.

7.1 Laburpena

Mundu errealeko eszenak hiru dimentsiotan eta modu digitalean berreraikitzeko gaitasuna duten software eta teknikak aztertu eta horien arteko konparazio bat egitea izan da proiektu honen helburu nagusia. Ikusmen artifizialaren mundu zabalen, gaur egun, gero eta erabiliagoa den argi egituratuaren alorra aztertu da.

Laburbilduz, honako hauek izan dira proiektuan landutako ataza nagusiak:

- 3D berreraikitzeak lortzeko softwareen bilaketa eta analisia
- SLStudio eta TI3D softwareen instalazioa, konfigurazioa eta kalibrazioa
- Softwareak konparatzeko Cloud Compare softwarearen instalazio eta analisia
- SLStudio eta TI3D software libreen eta DAVID software komertzialaren arteko konparazioa
  - Software bakoitzarekin lortutako puntu-hodeien artean lau alderaketa nagusi egin dira: prozesatu gabeko puntu-hodeien konparazioa, lerrokatzeen arteko

konparazioa, berreraikitze osoen arteko konparazioa eta puntu-hodeien analisi zuzena

- Ondorioak ateratzea

7.2 Ondorioak

Lehenik eta behin, proiektuaren helburuei dagokienez, lortu direla esan beharra dago, izan ere, gure sistema software libre baten bitartez erabiltzeko prest gelditu da, eta espero zen konparazioa egin da. Hala ere, ez dugu lortu software librearen bitartez DAVID 3D software komertzialaren zehaztasuna lortzerik. Hori horrela baldin bada ere, lorturiko emaitzak ez dira txarrak izan. Izan ere, konparazioaren atalean ikus daitekeen moduan, software libre bidez lorturiko emaitza onenak ez daude DAVID 3D softwareak eskaintzen duen zehaztasunetik hain urruti SLStudioren kasuan, eta kontuan izan behar dugu, probatutako software komertzialak baina patroi gutxiago erabiltzen dituela. Hori dela eta, sistema azkarragoetan erabilgarritasuna izan dezake.

Argi egituratua, gaur egun, gero eta gehiago erabiltzen ari den teknologia baldin bada ere, oraindik ere software librearen munduan egiteko ugari dagoela ikusi dut, oraindik ere software eta algortitmo gehienak komertzialak baitira. Atentzioa deitu zidan, TI-ko foroetan jasoriko laguntzak, eta batez ere, SLStudio-ren sortzaileak gure eskaerei zegokienean erakutsi zuen jarrerak. Aberasgarria iruditu zitzaidan hartu-eman hori mantentzea.

Horrez gain, puntu-hodeien prozesamendurako baliabide ahaltsuak erakutsi dituen software libre bat aztertu da, Cloud Compare. Azterketa hori hasiera batean finkaturik zeuden helburuen barruan ez bazegoen ere, oso aberasgarria izan dela esan daiteke eta etorkizunean honen erabilera Tecnaliako ikerketa-taldean gehiago integratzeko aukera dagoela.

Ikusmen artifizialaren ikerketa-talde batean egotean, gaur egun robotikan erabiltzen diren aurrerapenak ezagutu eta horiekin harremana izateko aukera aberasgarria izan dut. Berreraikitzeak burutzeko erabiltzen diren teknika ezberdinak aztertu eta ezagutu ditut. Ikerketa horretan, sistema hauen alde konputazionala nolakoa den ikasteaz gain, kameraren bidezko irudi atzipenean dagoen mundua ere ezagutu dut, gainetik bada ere. Behin berreraikitzeak lortzean, horien prozesamenduan erabiltzen diren teknika ezberdinak ikasi ditut, hala nola, kalitate-kontrolean hainbeste erabiltzen den puntu-hodei eta modeloaren arteko lerrokatzea. Ikuspegi ezberdinetatik lorturiko berreraikitzeetatik abiatuz, eskaneatu nahi dugun pieza osoa nola lortzen den ere ikasi dut. Puntu-hodeien ezaugarriak aztertzeko

metodo ezberdinak erabili ditut, analisia puntu bakoitzaren ezaugarri ezberdinak aztertuz aurrera eramanez, dentsitatea edo kurbadura aztertuz.

Zenbait software libreren instalazio eta konfigurazioak zer nolako konplexutasuna izan dezakeen ikasi dut, batez ere, erabilitako hardware edo sistema eragilea berdina ez denean. Horrez gain, argi egituratuan oinarritutako sistemen kalibrazio optimo bat lortzearen zailtasunak ere ikusi ditut, bertan faktore askok esku-hartzen dutela ikusi baitut. Horrez gain, espero ez nituen hainbat arazo izan ditut proiektuan zehar, adibidez proiektua garatzeko erabilitako ordenagailua lortzeko arazo ezberdinak izan ditut konpainiako egitura dela eta.

Epeei dagokienez, ez zen lortu aurreikusten zen bezala proiektua ekainean entregatzea eta aipatutako arazo eta garapen lan-karga zirela medio, proiektuaren amaiera hilabete bat atzeratu da. Gainera, aipatu behar dut kalibrazio egoki bat bilatu nahian gehiegi tematu naizela eta nahi baina denbora gehiego inbertitu dudala ataza horretan. Gainera, inoiz dimentsio honetako proiektu bat garatu ez izanak denborak kontrolatzeko zailtasunak ekarri zizkidan.

7.3 Etorkizuneko lanak

Ondorengoak izan dira gure proiektuan etorkizunean lantzeke geratu diren lanak:

- Hardware *trigger*-a eta USB 3.0-a konfiguratu SLStudio softwarean, modu honetan berreraikitze-abiadura altuak lortuko genituzkeelako. Ezaugarri horrekin, berreraikitze jarraia egiteko gai izango ginateke, hau da, eszenaren hiru dimentsiotako atzemate jarrai bat.

- uEyeren API-aren ezaugarri gehiago software libreekin integratu hasieratze-denbora azkartzea lortuz eta horrez gain, software osoago bat osatuz.

- Probatu gabe utzi genuen 3D Underworld softwarea martxan jarri bere errendimendua aztertzeko.

- Kameran filtro polarizatuak probatu pieza dirdiratsuen berreraikitzeak hobetzen diren ikusteko. Softwareen kalibrazio eta probaketa faseetan, argi egituratuan oinarriturik dauden sistemek eszenako dirdirekin arazoak zituztela ikusi genuen.

- DAVID-ek eskaintzen duen triangelu-sareak zuzenean sortzeko aukera integratu instalaturiko software libreetan. PCL liburutegien bitartez egin daitekeela ikusi genuen.
- Zarata lortzeko iragazkiak zuzenean software libreekin integratu, batez ere SLStudioren kasuan interesgarria litzateke. Horretarako ere, PCL liburutegiak erabil ditzakegu.
- Patroi gehiagotan oinarritzen diren kodetzeak bilatu edo guk geuk sortu, adibidez SLStudiok eskaintzen dituen *gray code* eta *N Phase Shift* konbinatzen saiatuz.

Bibliografia

Horn, B.K.P. (1987). Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society of America, Vol. 4(4), pp. 629-642.

Munro, P. and Gerdelan, A. (2006). Stereo Vision - Computer Depth Perception, Tech report, 159.731 Machine Vision, Massey University.

Nevala, E. (2014). Introduction to Octrees. https://www.gamedev.net/articles/programming/general-and-gameplay-programming/introduction-to-octrees-r3529/, January 20, 2014.

Salvi, J., Pages, J., Battle, J. (2004). Pattern codification strategies in structured light systems. Pattern Recognition, Vol. 37(4), pp. 827-849.

Torre, C. (2010). Contribuciones al alineamiento de nubes de puntos 3D para su uso en aplicaciones de captura robotizada de objetos, Tesis Doctoral, Universidad de Cantabria.

Wilm, J., Olesen, O.V. and Larsen, R. (2014). SLStudio: Open-Source Framework for Real-Time Structured Light, 4th International Conference on Image Processing Theory, Tools and Applications, DOI: 10.1109/IPTA.2014.7002001, IEEE Publishers.

Wilm, J., Oline Vinter, O., Rasmus, L. (2014a). Accurate and Simple Calibration of DLP Projector Systems, Proceedings of SPIE, the International Society for Optical Engineering, DOI:10.1117/12.2038687