SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Mirta Čolić, Iva Hrastnik, Luka Karniš, Andro Katanec, Lovro Pliskovac, Dominik Sekula

SimplyDigitize - sustav za digitalizaciju zasnovan na fotogrametriji

Zagreb, 2022.

Ovaj rad izrađen je u Laboratoriju za robotiku i inteligentne sustave upravljanja na Fakultetu elektrotehnike i računarstva pod vodstvom izv. prof. dr. sc. Matka Orsaga te asistentice mag. ing. Jelene Vuletić i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2021./2022.

SADRŽAJ

1. Uvod

U 21. stoljeću rastom popularnosti 3D printera raste i interes za 3D skenere, strojeve koji kamerom snimaju predmet iz svih kutova te koristeći fotogrametriju samostalno izrađuju 3D model snimljenog predmeta. Na tržištu se 3D skeneri mogu pronaći u raznim oblicima i veličinama, s raznim načinima rada te se koriste u različite svrhe [1][2]. Primjeri takvih skenera prikazani su u sljedećem poglavlju na slici 2.1.

Tim koji je izradio ovaj rad sastoji se od šest studenata Fakulteta elektrotehnike i računarstva koji su se odlučili okušati u izradi cijelog sustava za 3D skeniranje naziva "SimplyDigitize". Početak je bio izrada samog skenera, ali je prošireno na usavršavanje autonomije skeniranja i izradu web stranice s prikazima 3D modela. Programsko sučelje sastoji se od ROS paketa koji objedinjuje tri čvora, za kameru, motore i Meshroom, koji su nužni za uspješnu izradu 3D modela objekta iz stvarnog svijeta.

U posljednje vrijeme, posebno od pojave pandemije 2020. godine, sve su popularnije virtualne izložbe. Virtualna izložba je odlična prilika da sadržaj bude dostupan većem broju ljudi kao i onima koji su u nemogućnosti doći na izložbu uživo. Skener je predviđen za skeniranje različitih kolekcija predmeta, a posebno zanimljiva primjena bila bi skeniranje umjetnina i izrada virtualne izložbe. Upravo iz tog razloga, u suradnji s Gliptotekom HAZU-a dogovorena je buduća suradnja u kojoj bi koristeći "SimplyDigitize" bile skenirane odabrane skulpture i izrađeni pripadajući 3D modeli. Modele se može prikazati online na web stranici u obliku online galerije gdje je klikom na fotografiju odabrane skulpture moguće vidjeti njegov 3D model, odnosno samu umjetninu iz više kutova.

U sklopu ovog rada prikazana je funkcionalnost sustava na problemu skeniranja

kolekcije ručno rađenih plišanih igračaka. Za navedenu kolekciju izrađena je online galerija kojom se demonstrira funkcionalnost sustava. Primjer dva plišana medvjedića korištena za testiranje sustava moguće je vidjeti na slici 1.1, a slike njihovih gotovih 3D modela na slici 1.2.

Slika 1.1: Fotografija medvjedića korištenih za skeniranje

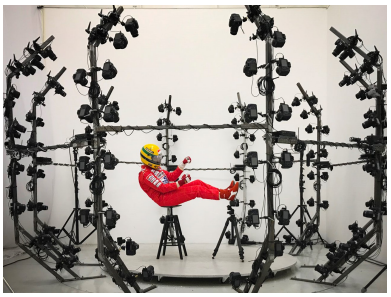
Slika 1.2: 3D modeli medvjedića

2. Tehničke značajke 3D skenera

Prvi korak pri dizajniranju skenera je istraživanje postojećih rješenja i određivanje zahtjeva dizajna. Među postojećim industrijskim rješenjima skeniranje predmeta se može obavljati slikajući predmet na rotirajućoj podlozi koristeći jednu kameru ili slikajući stacionaran objekta s mnogo kamera, kao što je prikazano na slici 2.1. Skener s više kamera nema pomičnih dijelova, zbog čega je jednostavniji i brži. Međutim ovakvi skeneri su višestruko skuplji zbog većeg broja kamera koje su potrebne. Puno ekonomičniji pristup je koristiti jednu kameru koja slika predmet iz više kutova. Mana ovog rješenja je da je potrebno dovesti predmet u različite poze ili pomicati kameru oko objekta bez mijenjanja scene.

U ovom poglavlju je dan pregled hardvera i softvera korištenog u projektu. Prvo je opisano projektiranje i izrada konstrukcije skenera, zatim je dan pregled korištene opreme, odnosno kamere i motora. Na kraju je opisan korišteni softver i kako se njime upravlja hardverom.

(a) EinScan-SE skener [1]

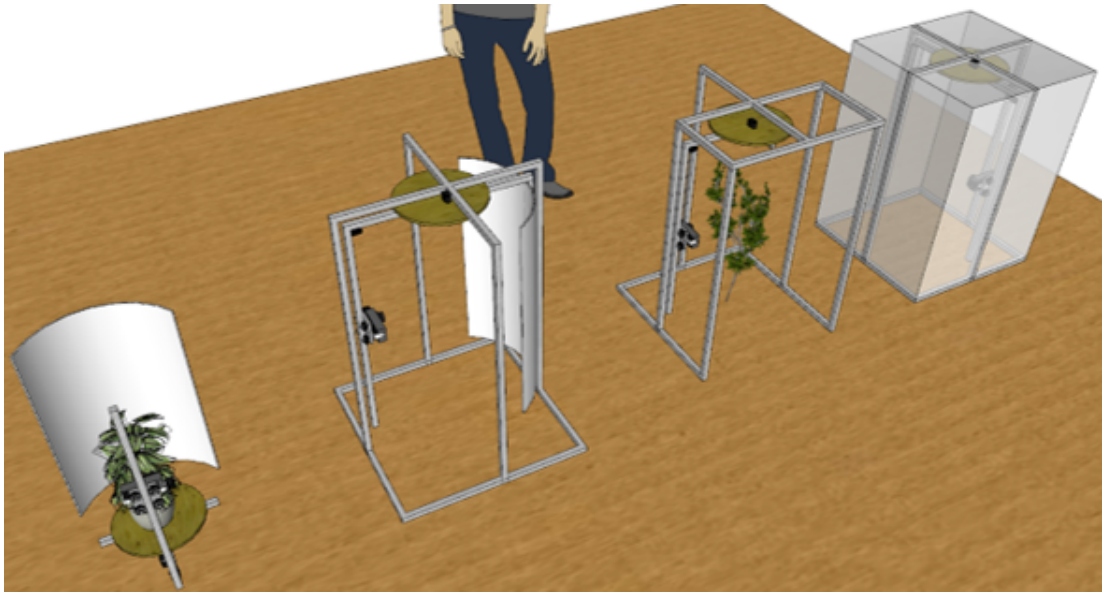


(b) Posebno izrađen skener [2]

Slika 2.1: Komercijalni fotogrametrijski skeneri

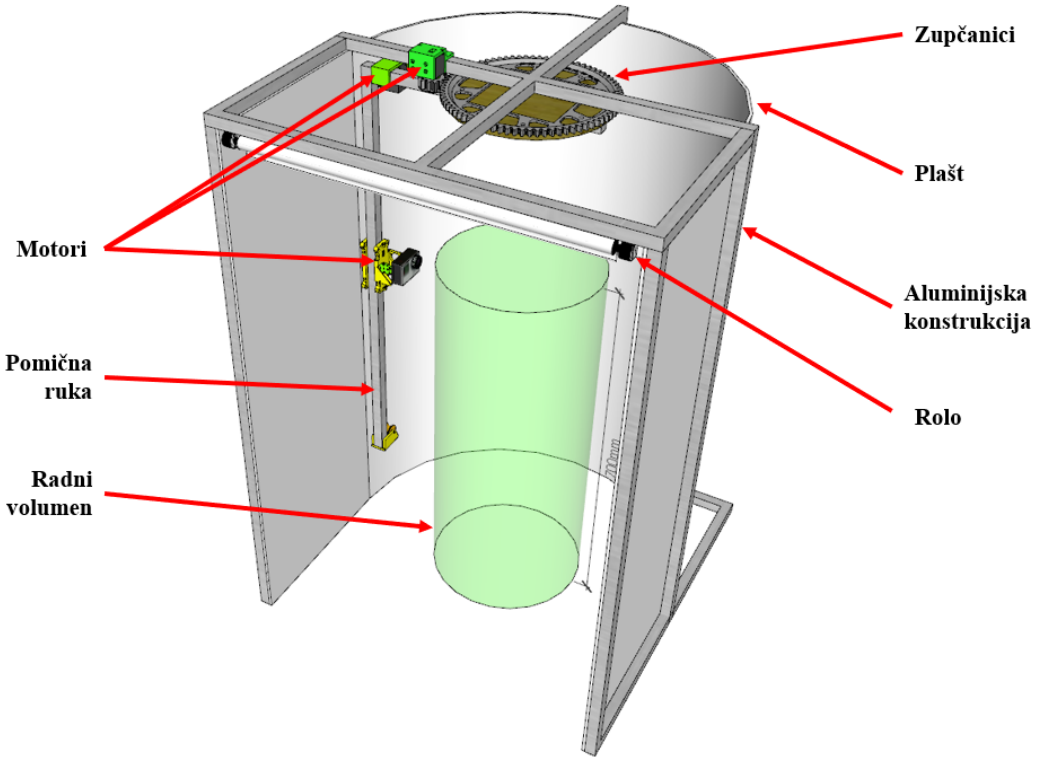
2.1. Konstrukcija

Cilj izrade konstrukcije skenera je omogućit skeniranje što većeg unutrašnjeg volumena, a da su pri tom vanjske dimenzije što manje. Potrebno je omogućiti da se skener lako može prenositi zbog čega ne smije biti težak. Bitno je, tijekom skeniranja koje može trajati duže vrijeme, da se predmet ne pomiče te da se osvjetljenje ne mijenja. Stoga je potrebno da skener ima svoje osvjetljenje i da je objekt zaštićen od vanjskih utjecaja. Razna idejna rješenja, prikazana na slici 2.2, napravljena su u softveru za 3D modeliranje *Google SketchUp-u*, koji je zbog svoje jednostavnosti idealan za izradu prototipova.



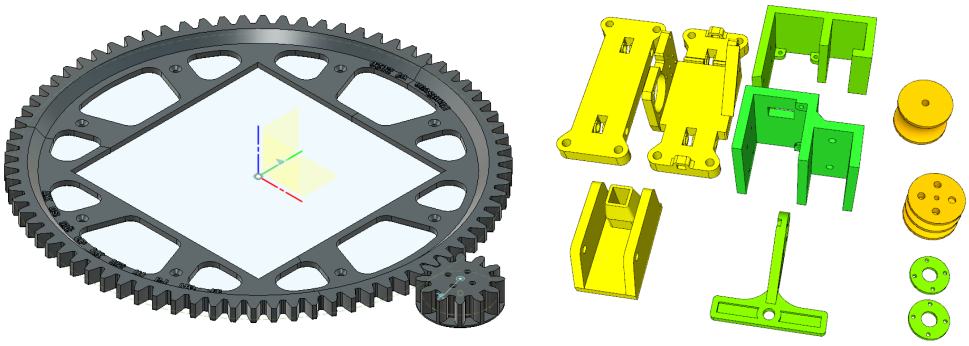
Slika 2.2: Idejna rješenja

Odabran je dizajn sa slike 2.3 jer je kod takvog dizajna moguće dovesti skener nad nepomičan objekt i plašt ga štiti od vanjskih utjecaja čime je osigurano konstantno osvjetljenje. Također takva izvedba skenera ne dodiruje objekt, što je prednost u odnosu na druge prototipe jer svojim pomicanjem skener ne trese objekt koji slika, što ubrzava proces slikanja.



Slika 2.3: Odabrani dizajn

Konstrukcija skenera, kao i jednostavniji dijelovi za pokretanje, projektirani su u programu *Google SketchUp-u*. Zupčanici prikazani na slici 2.4, koji rotiraju ruku skenera, projektirani su u programu za projektiranje *Autodesk Fusion 360*. Za njihovu izradu korišten je *SpurGear* dodatak. Okvir skenera izrađen je od 11 metara kvadratnih aluminijskih cijevi i 15 plastičnih spojnica. Aluminijske cijevi se mogu spajati spojnicama u različite konfiguracije te za to nisu potrebni posebni alati. Skener je dimenzioniran tako da ima maksimalan volumen skeniranja te je pritom dovoljno malen da se može prenositi kroz standardna vrata. Konačne dimenzije skenera su 720x720x1050 mm. Kamera se rotira oko centra skenera zbog čega je najveći volumen koji je moguće skenirati cilindar promjera 30 cm i visine 70 cm.



(a) Zupčanici u programu Fusion 360 **(b)** Ostali dijelovi za pokretanje kamere

Slika 2.4: 3D isprintani dijelovi

Kako bi unutar skenera bili idealni uvjeti za skeniranje potrebno je ispuniti nekoliko preduvjeta. Predmet mora biti jako i uniformno osvijetljen sa svjetlom bijele boje. Jaka svjetlost je potrebna kako bi kamera mogla snimiti dobre fotografije s puno detalja. Direktno vanjsko svjetlo, poput sunčeve svjetlosti, može stvoriti sjene koje su nepoželjne na konačnom modelu. Bitno je i da se tijekom skeniranja scena ne mijenja.

Osvjetljenje u skeneru sačinjavaju dvije trake pričvršćene za pomičnu ruku te dvije žarulje pri vrhu unutrašnjosti skenera, što je prikazano na slici 2.5a. LED trake su stavljene sa svake strane kamere i protežu se od vrha do dna unutrašnjosti skenera, što osigurava da na modelu nema sjena bez obzira na oblik predmeta koji se skenira ili poziciju kamere. Dvije žarulje pri vrhu dodatno osvjetljavaju prostor te su postavljene kako bi difuzno osvjetljavale predmet. Za osvjetljenje, koje se kontrolira prekidačem, potrebno je imati mrežno napajanje.

Plašt, odnosno pozadina skenera napravljena je od bijelo obojenog MDF-a koji je zakovicama pričvršćen za okvir, kao što je prikazano na slici 2.5b Zbog svoje mat obrade ne stvaraju se odsjaji od svjetla, a svjetlo se dobro raspršuje, poboljšavajući uniformnost osvjetljenja. Plašt također štiti predmet skeniranja od vanjskih utjecaja, kao što su različita vanjska osvjetljenja ili vjetar (npr. skeniranje biljke).

Jedna strana skenera nije prekrivena bijelim MDF-om kako bi se moglo pristupiti unutrašnjosti skenera. Na toj strani je postavljena rolo zavjesa koja ispunjava istu zadaću kao i MDF ploče, ali ju je moguće otvarati i zatvarati po potrebi.

Na vrhu skenera postavljena je ploča kako bi se na nju mogao postaviti laptop koji se spaja s motorima i kamerom.

(a) Osvjetljavanje objekta snimanja

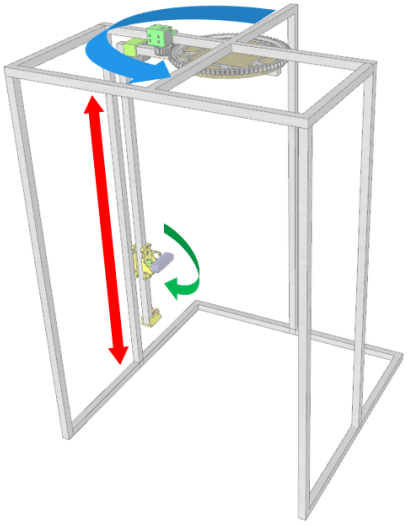
(b) MDF pozadina

Slika 2.5: Prikaz skenera

Skener ima 3 stupnja slobode, prikazana na slici 2.7, te se pokreće Dynamixel motorima. Konkretno, koriste se tri Dynamixel XL430-W250-T motora koji su detaljnije obrađeni u poglavlju 2.3.

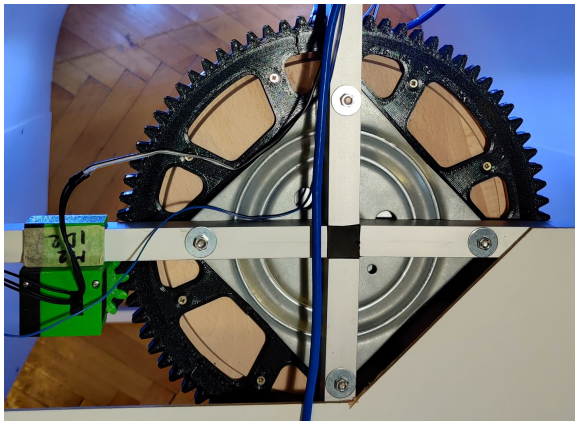
Na vrhu skenera preko okretnog nosača pričvršćena je aluminijska ruka po kojoj se kamera pomiče gore-dolje. Ruka je složena od dvije kvadratne aluminijske cijevi spojene na okruglu dasku, te je prikazana na slici 2.6.

Slika 2.6: Ruka skenera

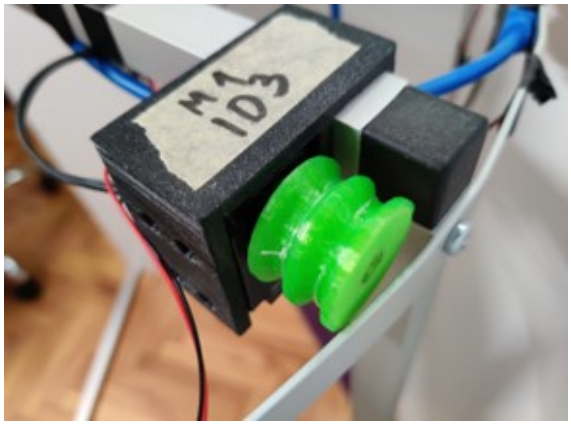


Slika 2.7: stupnjevi slobode skenera

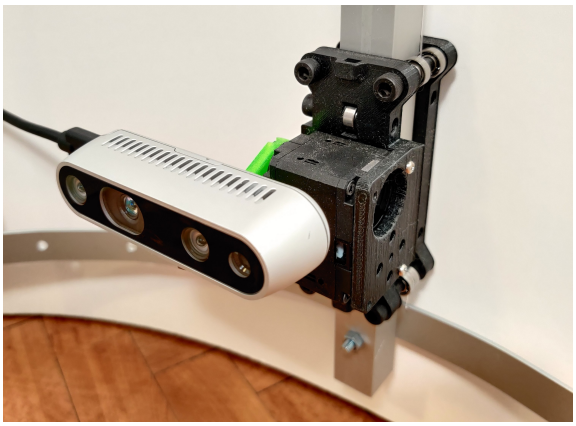
Ruka se okreće preko zupčanika i motora koji je pričvršćen za okvir. Nosač kamere napravljen je od dvije 3D isprintane ploče koje su povezane vijcima i na kugličnim ležajevima klize duž ruke. Nosač kamere najlonskim je koncem povezan na kolotur pri vrhu ruke koji je spojen na motor. Mehanizmi za pokretanje prikazani su na slici 2.8.



(a) Rotacija ruke



(b) Podizanje i spuštanje nosača kamere



(c) Namještanje nagiba kamere

Slika 2.8: Motori za pokretanje kamere

2.2. Kamera

Zadaća kamere je prikupljanje slika predmeta temeljem kojih će se kasnije izraditi i sam model skeniranog predmeta. Slike moraju biti što bolje rezolucije kako bi se uhvatili svi detalji predmeta kojeg se skenira, a budući da je unutarnji prostor skenera relativno malen, bitno je da kamera ima široko vidno polje. Uz to, prostor za kameru je ograničenih dimenzija, stoga je bitno da je kamera što manja.

Slika 2.9: Prikaz korištene kamere [3]

Uzevši sve gore navedene zahtjeve u obzir, odabrana je *GoPro* kamera. Konkretno, korišten je model *GoPro Hero 4*, prikazan na slici 2.9.

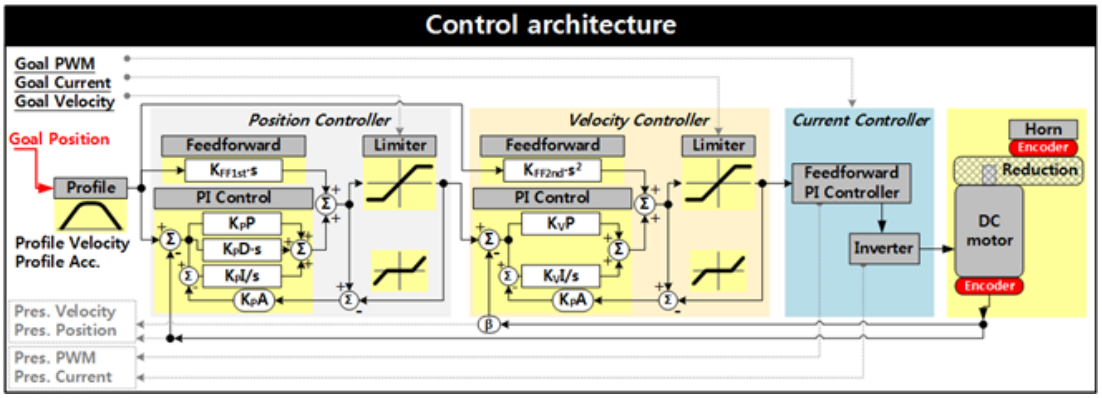
Još jedna od prednosti ove kamere je što postoji prilagođena python biblioteka *goprocam* [4] koja služi za povezivanje *GoPro* kamere s programskim jezikom Python. Pomoću ove biblioteke ostvaruje se veza s *GoPro* kamerom preko WiFi-ja te se u bilo kojem trenutku može uslikati slika pomoću naredbe *gopro.take_photo()*.

2.3. Motori

Kako bi kamera prikupila potrebne slike mora proći određenu putanju oko objekta koji se skenira, a to joj omogućuju 3 motora koja rade zajedno da bi kameru postavili u potrebne pozicije. U implementaciji SimplyDigitize koriste se Dynamixel motori. Dynamixel je proizvođač pametnih aktuatora koje je razvio ROBOTIS za korištenje u robotici. Dynamixel nudi veliku raznovrsnost u odabiru motora koji se primjenjuju za izradu višezglobnih robotskih konfiguracija poput humanoidnih robota i manipulatora.

Slika 2.10: Modeli Dynamixel motora

Modeli Dynamixel motora se međusobno razlikuju po brzini, zakretnom momentu i naredbenim signalima te su prigodni za osobnu, ali i za profesionalnu upotrebu. Dynamixel motori su integrirani što znači da se svaki aktuator sastoji od istosmjernog motora, senzora, kontrolera, reduktora, pogona i mreže integrirane u jedan modul. Napajani su istosmjernim naponom iznosa 12V, što rezultira manjom strujom kako bi se postigla veća energetska učinkovitost. Dynamixel motori digitalno komuniciraju preko manjih podatkovnih paketa koji koriste polu-duplex UART osam bitni protokol. Prednost ovih motora je način upravljanja, korisnik može birati između različitih načina rada: upravljanje strujom (zakretnim momentom), upravljanje položajem, upravljanje kontrolom brzine i upravljanje PWM-om. Također je omogućena kompletna kontrola pozicije i brzine PID regulatorom koja rezultira preciznim pokretima motora 2.11.



Slika 2.11: Upravljački model dynamixel motora

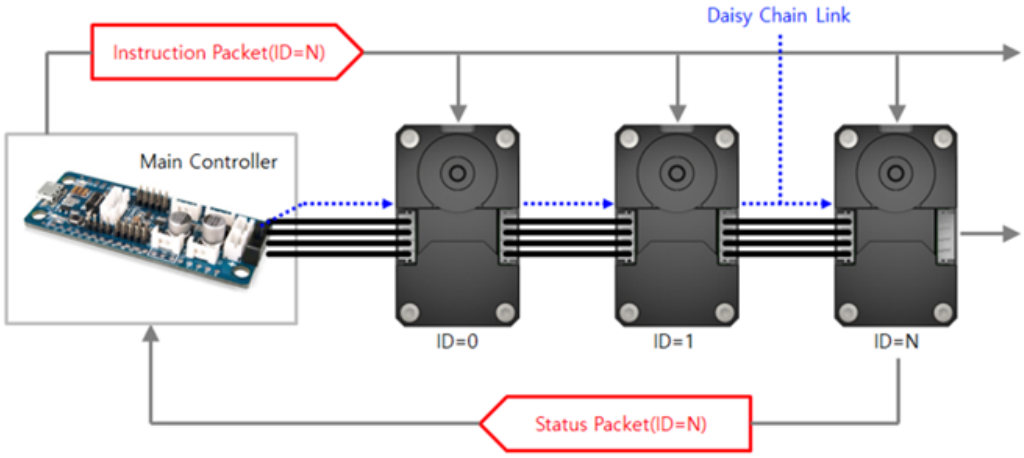
Rad s Dynamixel aktuatorima podržan je u mnogim razvojnim okruženjima i programskim jezicima poput JAVE, ROS-a, MATLAB-a, PYTHON-a i C/C++ na većini

operacijskih sustava kao što su Linux, Windows, Mac OS i Arduino.

Modeli Dynamixel motora koji postoje su Dynamixel, Dynamixel-X i Dynamixel-P, prikazani na slici 2.10. Za izradu skenera korišten je Dynamixel-X (prikazan na slici 2.12) koji je najnoviji model visokih performansi najčešće korišten za izradu višezglobnih robota.

Slika 2.12: Dynamixel-X model

Dynamixel motori se međusobno povezuju daisy-chain topologijom preko koje se šalju podatkovni paketi preko jedne zajedničke komunikacijske sabirnice. Daisy-chain mreža je način spajanja motora gdje je više motora spojeno u strukturu koja omogućava spajanje motora s kontrolerom i izvorom napajanja preko samo jednog kabla. Budući da ova struktura nalikuje vijencu tratinčica engl. *daisy*, dobila je naziv daisy-chain. Daisy-chain ustaljena je topologija mreža koja se koristi za povezivanje čvorova mreže linerano ili u prsten. Grafički prikaz daisy-chain mreže prikazan je na slici 2.13



Slika 2.13

2.4. Softver

Kako bi se moglo upravljati kamerom i motorima skenera, te cijeli proces skeniranja povezati u jednu cjelinu potrebno je koristiti programsko okruženje pomoću kojeg se to može relativno jednostavno izvesti.

2.4.1. Robotski operacijski sustav

Korišten je Robotski Operacijski Sustav (eng. *Robot Operating System*), skraćeno ROS. [5] ROS je open-source programsko okruženje koji pomaže istraživačima i programerima u razvoju aplikacija za robote.

Izrada softvera za robote je vrlo zahtjevan i kompleksan zadatak jer različiti roboti imaju različite hardvere. Kako bi sve funkcioniralo, potrebno je napisati softver počevši od najniže upravljačke razine pa sve do viših razina poput percepcije, apstraktnog razmišljanja, planiranja misija itd.[6] ROS pojednostavljuje ove probleme omogućujući brži i jednostavniji razvoj robotskih aplikacija.

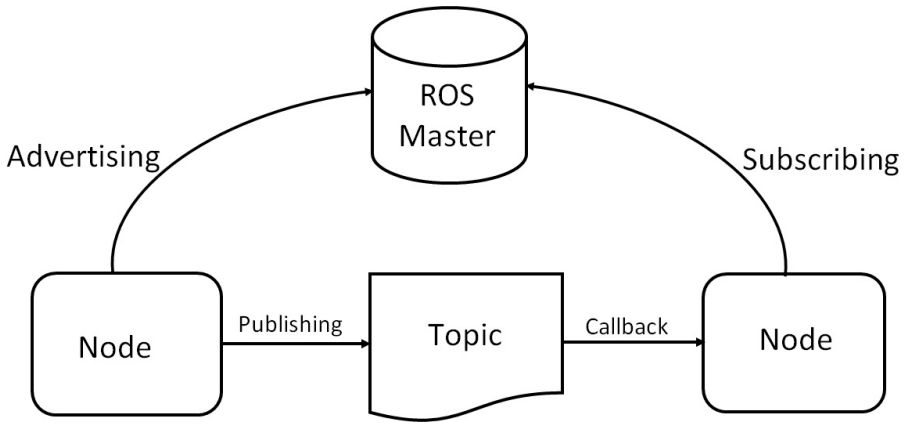
ROS je jezično neutralan (eng. *language-neutral*) zato što podržava 4 programska jezika C++, Python, LISP i Octave. To su vrlo različiti programski jezici, što omogućuje programerima veliku fleksibilnost u odabiru onog jezika koji njima i projektu na kojem rade najviše odgovara.

Temeljni implementacijski koncepti u ROSu su čvorovi (eng. *nodes*), poruke (eng. *messages*), teme (eng. *topics*) i servisi (eng. *services*).

Čvorovi su procesi koji provode izvršavanje manjih zadataka, a robotski sustav obično se sastoji od puno takvih čvorova. Čvorovi međusobno komuniciraju prosljeđivanjem poruka. Poruka je strogo tipizirana struktura podataka. Podržava uobičajene primitivne tipove podataka (integer, floating point, boolean, itd.) kao i nizove primitivnih podataka i konstanti. Poruke također mogu biti sastavljene od drugih poruka i nizova drugih poruka, ugniježđene proizvoljno duboko.

Čvorovi šalju poruke tako što ih objave (eng. *publish*) na određenu temu, a čvor koji je zainteresiran za određenu temu može se pretplatiti (eng. *subscribe*) na nju. Više

čvorova može objavljivati na istu temu, isto kao što se više čvorova može pretplatiti na istu temu. Skica arhitekture jednostavnog ROS sustava prikazana je na slici 2.14.



Slika 2.14: Prikaz jednostavne ROS topologije [7]

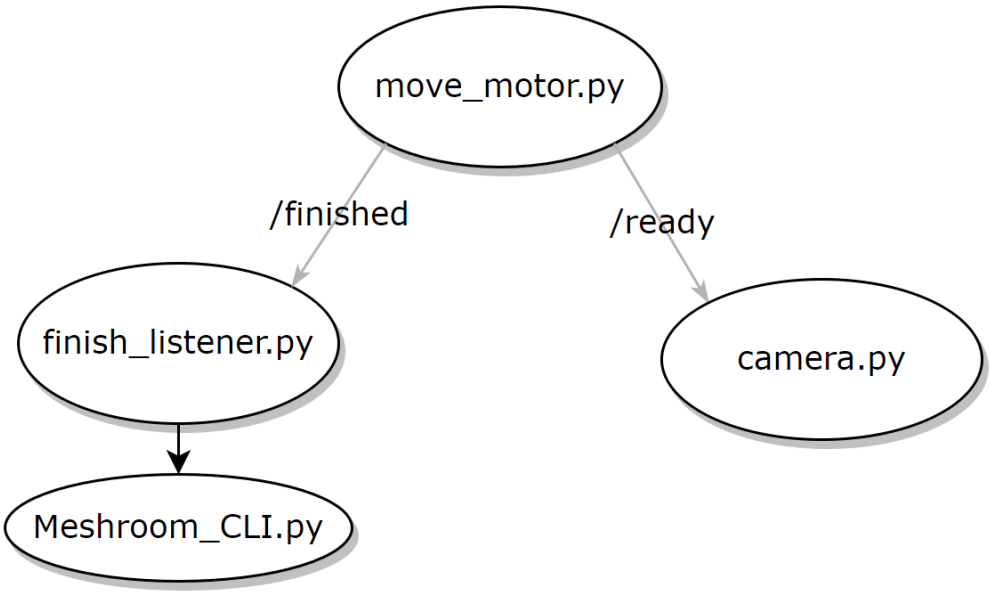
Iako model objavljivanja i pretplate na određenu temu omogućuje fleksibilnu komunikaciju između čvorova, ona nije prikladna za sinkronizirane transakcije. Ovaj problem rješavaju servisi, koji funkcioniraju po principu zahtjev-odgovor. Definirani su imenom i parom strogo napisanih poruka: jedna za zahtjev, druga za odgovor. Za razliku od tema, samo jedan čvor može objavljivati servis određenog imena.

Svaki zaseban čvor može se pokrenuti iz naredbenog retka, a budući da ih obično ima puno, uzastopno upisivanje naredbi može postati zamorno. Kako bi se omogućilo zajedničko pokretanje više čvorova odjednom, ROS pruža alat pod nazivom *roslaunch*. *Roslaunch* čita XML datoteku te pokreće zadane čvorove jedan za drugim, što omogućuje lakše manipuliranje kompleksnih ROS paketa.

Konkretno, ovaj programski paket sastoji se od tri čvora napisana u [8] programskom jeziku Python: `camera.py`, `move_motor.py` i `finish_listener.py`, te `Meshroom_CLI.py` Python skripte.

Čvor `move_motor.py` služi za upravljanje motorima na skeneru te radi upareno s čvorom `camera.py` koji služi za upravljanje kamerom i snimanje slika. Kad motori dođu u poziciju na kojoj se treba snimiti slika, `move_motor.py` čvor šalje poruku preko teme `/ready` kao znak kameri da može snimiti sliku. Kada je skeniranje predmeta gotovo čvor `move_motor.py` šalje poruku čvoru `finish_listener.py`

preko teme `/finished` kao znak da on može pokrenuti skriptu `Meshroom_CLI.py` koja pokreće izradu 3D modela iz prikupljenih slika. Diagram korištenih čvorova i tema preko kojih komuniciraju prikazan je na slici 2.15



Slika 2.15: Prikaz toplogije razvijenog ROS paketa

Kako bi se svi čvorovi povezali i pokrenuli istovremeno napravljena je *roslaunch* skripta pod nazivom *start_scanning.launch* preko koje se proces skeniranja može pokrenti pomoću naredbe:

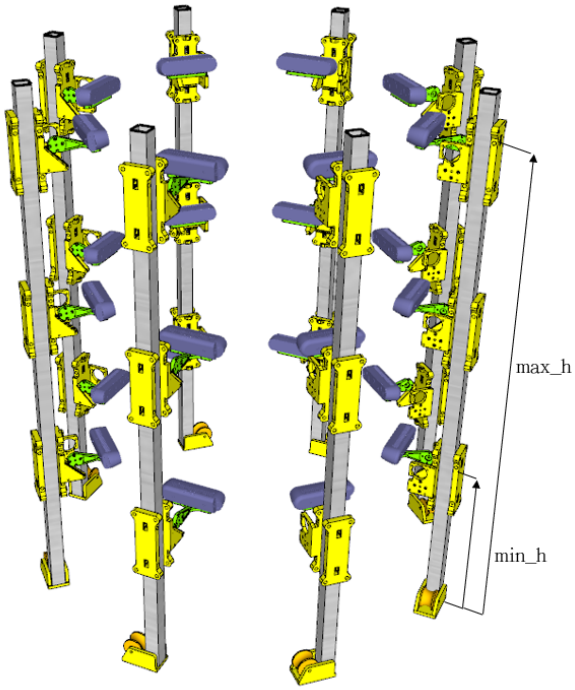
```
roslaunch simply_digitize start_scanning.launch
```

2.4.2. Upravljanje motorima i kamerom

Motori su opremljeni 12-bitnim enkoderom za određivanje pozicije i brzine vrtnje. Korišteno je upravljanje pozicijom motora. Kad se pokrene skeniranje očitaju se trenutačne pozicije motora i postave se kao početne pozicije od kojih će započeti skeniranje.

Prije skeniranja potrebno je postaviti parametre u skripti `move_motors.py` koji određuju kako će se kamera pomicati oko predmeta. Prvo se postavljaju najniža i najviša točka kamere s parametrima `min_h` i `max_h`. Oni određuju visine od najniže po-

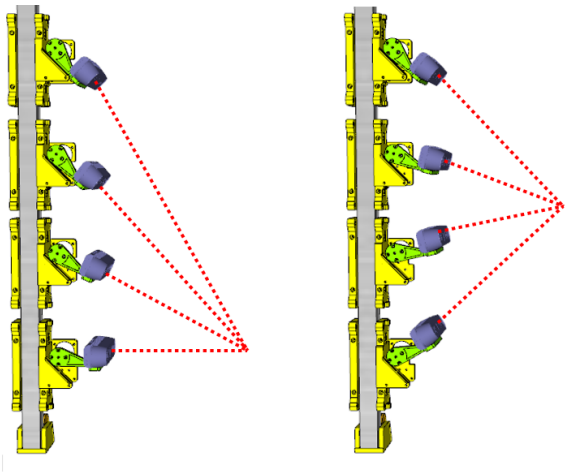
zicije nosača kamere u centimetrima. Zatim se mogu odrediti parametri `num_of_photos` i `num_of_h`, koji određuju iz koliko različitih kutova će se predmet slikati. Broj pozicija slikanja u jednoj rotaciji je određen parametrom `num_of_photos`. Broj visina na kojoj će se slikanje odviti određuje se parametrom `num_of_h`. Umnožak tih dvaju parametara nam daje ukupan broj slika koje će se dobiti pri jednom skeniranju. Na slici 2.16 mogu se vidjeti pozicije kamera koje će se dobiti kada bi se postavili parametri `num_of_photos` u 8 i `num_of_h` u 3.



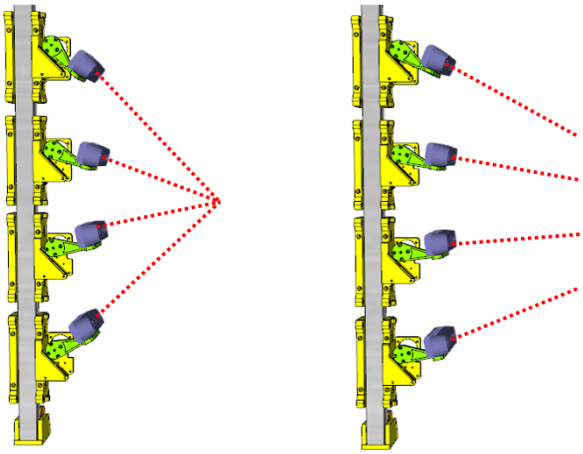
Slika 2.16: Primjer pozicija kamere za `num_of_h` = 3 i `num_of_photos` = 8

Nagib kamere prilikom kretanja određujemo parametrima `low` i `ang_amount`. Parametar `low` ograničava opseg nagiba kamere tako da se kamera naginje od horizontalnog položaja samo prema dolje i služi za skeniranje nižih predmeta. Na slici 2.17 (lijevo) se vidjeti kako će se kamera usmjeravati samo niže od horizontalnog položaja kada je `low` postavljen u jedinicu. Parametar `ang_amount` ograničuje raspon naginjanja kamere bez obzira na parametar `low`. Ako je `ang_amount` vrijednosti 1 onda će se kamera naginjati u punom rasponu. Kako ga korisnik smanjuje tako će se i raspon naginjanja kamere smanjivati. Postavljanjem `ang_amount` na vrijednost

0 onemogućuje se nagib kamere i ona ostaje horizontalna. Parametar `ang_amount` ovisi o veličini predmeta koji se skenira. Za `ang_amount` preporučuje se krenuti s iznosom `(max_h - min_h)/70`. Utjecaj parametra `ang_amount` može se vidjeti na slici 2.18. Opisi svih varijabli se mogu naći i u datoteci `move_motors.py`. Za izradu skupa od 720 slika (predmet slikan s 20 visina i 36 različitih kutova) potrebno je otprilike 40 minuta.



Slika 2.17: `low`=1 (lijevo) `low`=0 (desno)



Slika 2.18: `ang_amount1` > `ang_amount2`

3. Fotogrametrija

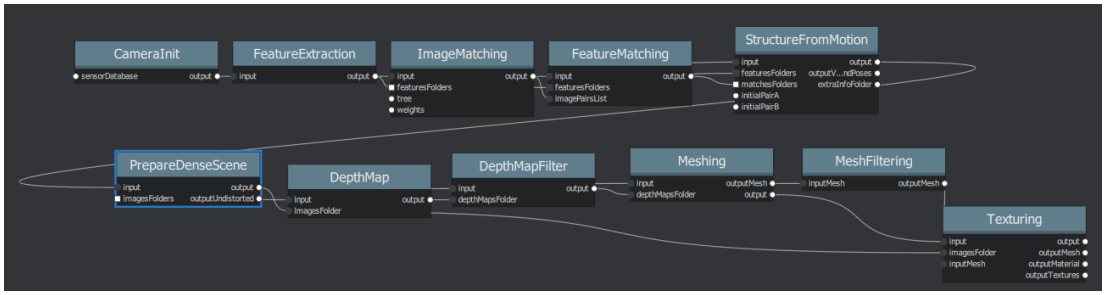
Povijesni razvoj tehnologija za stvaranje virtualnih okruženja doživio je eksponencijalni porast tijekom posljednjih desetljeća. Brojni su razlozi takvog naglog razvoja, počevši od stvaranja iznimno realističnih karata geografskih područja, mapiranja tekstura [9], stvaranja modela za korištenje pri snimanju serija ili stvaranja video igrica [10] [11]. Ovakve tehnologije su raznim timovima omogućile stvaranje nevjerojatno kvalitetnih prikaza lokacija i objekata s izuzetno malo potrebe za ljudskim intervencijama pri modeliranju.

Dok su takve tehnologije poznate po svojoj iskoristivosti na velikim projektima, gdje su financijske mogućnosti izuzetno velike, za potrebe skeniranja relativno malih objekata, sasvim je dovoljna tehnologija pod nazivom fotogrametrija. To je tehnologija za dobivanje informacija o fizičkim objektima kroz procese snimanja, mjerenja i interpretiranja fotografija [12]. Fotogrametrija je metoda generiranja 3D modela korištenjem 2D fotografija. Postoji više različitih tehnika izvedbi, od kojih najčešća podjela je na zračnu fotogrametriju (kamera pozicionirana u zraku) i vezana za zemlju (kamera na tlu ili na držaču blizu tla). Vrlo često se koristi u svrhe stvaranja modela statičkih elemenata prilikom razvoja video igara [13]. Također se koristi za dobivanje vrlo detaljnih modela muzejskih primjeraka kada su u pitanju digitalizacija i virtualne izložbe [14]. Neki od primjera fotogrametrijskih alata su AliceVision Meshroom, All3DP i Agisoft Metashape.

3.1. Meshroom

Meshroom, punim nazivom AliceVision Meshroom, je open-source alat za 3D rekonstrukciju i praćenje kamere [15]. On omogućava stvaranje detaljnih modela s teksturom kroz analizu skupova slika statične scene dobivenih raznovrsnim modelima kamera. Temelji se na structure-from-motion i multi-view-stereo tehnikama fotogrametrije koje zajedno analiziraju dobivene skupove slika i sakupljaju nepromjenjive točke s obzirom na poziciju i orijentaciju kamere u tzv. oblak točaka (engl. *point cloud*). Pomoću tih dobivenih točaka pozadinskim izračunima dobivaju se svi ostali potrebni parametri za stvaranje grube površine na koju se postavlja tekstura.

Računalno okruženje Meshroom-a omogućuje obavljanje različitih zadataka računalnog vida na vrlo jednostavan i intuitivan način. Svaki zadatak algoritma je reprezentiran kao čvor u usmjerenom acikličkom grafu (engl. pipeline), gdje su izlazi jednog čvora ulazi drugog. Parametri svakog čvora mogu se podešavati do iznimnih detalja, što stvara visoku razinu fleksibilnost pri radu s različitim kamerama, objektima ili lokacijama. Takvom implementacijom omogućuje se ponovno pokretanje potrebnog dijela pipeline-a pri promjeni bilo kakvih parametara (npr. promjena skaliranja rezolucije utjecat će na ponovno pokretanje samo zadnjeg čvora) zbog pričuvne memorije svakog čvora. Svi čvorovi grafa izvode se na desetcima ili stotinama različitih slika te je pri implementaciji zbog smanjenja računalnog tereta ovaj dio implementiran s elementima paralelizma.



Slika 3.1: Defaultni Meshroom pipeline

Iako je na 3.1 prikazana defaultna struktura pipeline-a za upravljanje Meshroom-om, moguće je dodavati nove, uklanjati i uređivati sve individualne čvorove, no potrebno je očuvati standardnu strukturu algoritma da bi on funkcionirao. Tim uređivanjem moguće je utjecati na vrijeme izvođenja cijelog grafa jer pri defaultnim postavkama stotinjak slika dobre kvalitete iziskuje nekoliko sati izvođenja. Opis tih čvorova dostupan je u potpoglavlju o implementaciji Meshroom-a preko Python skripte.

Osim grafičkog sučelja (GUI) Meshroom također nudi i izvođenje iz naredbenog retka (Meshroom CLI), što omogućuje automatizaciju cijelog procesa izrade 3D modela. Samo izvođenje se razlikuje jedino u tome što pri korištenju implementacije preko naredbenog retka, međurezultati i konačni rezultat čvorova se pasivno spremaju na disk, dok su pri korištenju GUI-a odmah prikazani u 3D Viewer-u. Izvođenje iz naredbenog retka je implementirano putem skripte pisane u Python jeziku.

3.2. Procesni koraci 3D rekonstrukcije

Rekonstrukcija je implementirana kao python skripta pomoću koje se pozivaju pojedini koraci iz Meshroom pipeline-a.

Python je programski jezik visoke razine poznat po svojoj lakoći čitanja i jednostavnosti pisanja. Python se najčešće koristi kao skriptni jezik, gdje nisu potrebne cjelokupne implementacije aplikacija već samo pokretačka skripta. Za potrebe ovog rada napisana je skripta pod nazivom Meshroom_CLI.py koja služi za pokretanja Meshroom-a iz naredbenog retka.

Implementacija skripte bazirana je na načinu izvedbe algoritma Meshrooma za stvaranje modela koji funkcionira u obliku pipeline-a i čvorova, gdje u skripti individualne funkcije predstavljaju čvorove grafa, a sama skripta predstavlja pipeline. Skripta se sastoji od 13 funkcija od kojih su dvije opcionalne (meshDecimate, meshResampling), a služe za uglađivanje i pojednostavljenje mesh-a.

Funkcije, odnosno čvorovi se pokreću sa svojim parametrima koje Meshroom pri normalnom izvođenju sam postavlja. Način korištenja tih funkcija možemo pronaći

u konzoli koju Meshroom nudi pri pokretanju unutar grafičkog sučelja ili u logovima spremljenima na računalu nakon završetka obrade. Nazivi funkcija su isti kao i čvorovi koje Meshroom implementira zbog jednostavnijeg i intuitivnijeg koda.
Skripta se pokreće naredbom:

```
python3 Meshroom_CLI.py "Path_1" "Path_2" "Path_3"
```

Pri čemu je **Path_1** put do Meshroom bin datoteke, **Path_2** put do proizvoljne datoteke za spremanje izlaza, a **Path_3** put do slika.
Primjer jednog takvog izvođenja:

```
python3 Meshroom_CLI.py
    "/home/user/Meshroom-2021.1.0/aliceVision/bin/"
    "/home/user/Desktop/output"
    "/home/user/Desktop/slike"
```

U nastavku su navedeni procesni koraci 3D rekonstrukcije. Svaka od navedenih funkcija koristi `binPath` i `baseDir` parametre za određivanje putanje do bitnih datoteka.

3.2.1. *cameraInit*

Funkcija *cameraInit* služi za inicijalizaciju kamere odnosno učitavanje metapodataka slika, informacija sa senzora s mogućnošću miješanja više različitih kamera i žarišnih duljina. Ulazni parametar je `imgDir` koji predstavlja apsolutnu putanju do slika. Vrijednost odabrana za vidno polje kamere je 170 zato što je na GoPro kameri korištena postavka Wide. Njezinim izvršavanjem stvara se datoteka cameraInit.sfm u kojoj su zapisane grupe svojstvenih vrijednosti na temelju metapodataka slika.

3.2.2. *featureExtraction*

Ulazni parametri funkcije *featureExtraction* su `numberOfImages` i `imagesPerGroup` koji će zajedno određivati broj slika tj. grupa slika koje će se istodobno slati na obradu. Ovaj korak izdvaja značajke iz slika, kao i opisnike (engl. *descriptor*) za te značajke tako da pronalazi različite skupine piksela koji su nepromjenjivi kada se mijenja gledište kamere.

Za potrebe ekstrakcije značajki, Meshroom nudi višestruke izbore za izvlačenje značajki pod parametrom `describerTypes`. U ovom radu korišten je algoritam SIFT (engl. *Scale-invariant feature transform*), što je najpoznatija metoda za detekciju značajki [16]. Odabrana kvaliteta ekstrakcije tj. `describerPreset` je normal, jer ne postoji neka izražena razlika u kvaliteti, dok je trajanje znatno veće pri korištenju opcije high. Također, s obzirom na razvijenost grafičkih kartica, parametar `forceCpuExtraction` postavljen je u 0 što označava da će ovaj korak iskorištavati samo GPU za izvlačenje značajki.

Izvršavanje ove funkcije koristi cameraInit.sfm te stvara feature i descriptor (*.feat, *.desc) datoteke te uz to i listu povezanih slika (imageMatches.txt) potrebnih za sljedeći korak.

3.2.3. *imageMatching*

Cilj ovog koraka je pronaći slike koje gledaju na ista područja scene, odnosno pronalazi povezane slike. Za poboljšanje algoritma povezivanja slika funkcija koristi parametar `tree`, koji predstavlja putanju do *vocabulary tree* datoteke u korist rada sa velikim skupovima podataka.

U ovom koraku izvršavanja funkcije stvara se imageMatches.txt, datoteka u kojoj su liste povezanih slika.

3.2.4. *featureMatching*

Za potrebe ove funkcije odabran je algoritam za pronalazak podudaranja pod nazivom ANN_L2, te geometrijski estimator acransac koji se prilagođava na utjecaj šuma u podacima. Te opcije vidljive su u parametrima `photometricMatchingMethod` i `geometricEstimator`.

Koristi sve do sada dobivene datoteke da uskladi značajke koristeći iznad navedene algoritme između parova slika kandidata dobivenih u prijašnjim koracima. Kao i kod *featureExtraction* funkcije, cilj je slati slike u grupama od maksimalno 40 članova (za naš slučaj), jer je to, prema dokumentaciji Meshroom-a, najefektivniji način njihove obrade. Iz tog razloga funkcija ova funkcija također koristi ulazne parametre `numberOfImages` i `imagesPerGroup`.

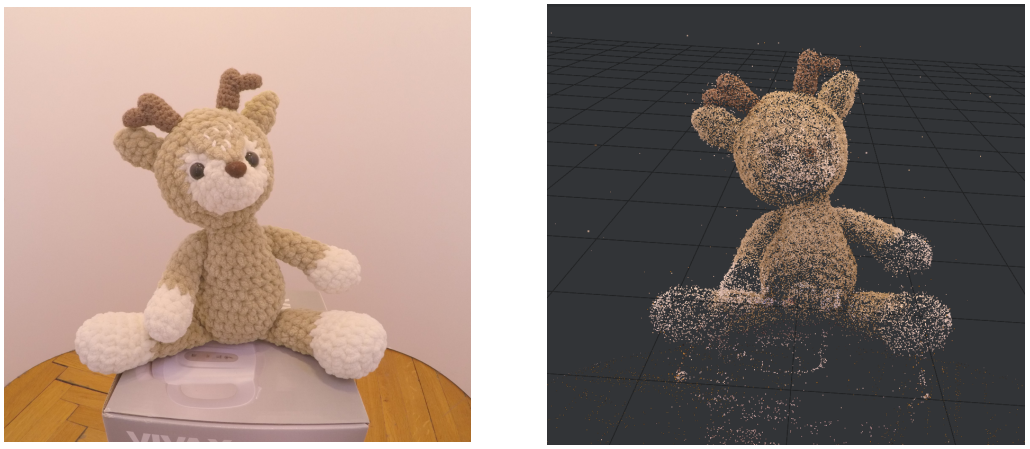
Funkcija nakon izvršenja stvara tekstualne datoteke koje predstavljaju korespondenciju između slika, koristeći opisnike značajki.

3.2.5. *structureFromMotion*

Na osnovu značajki dobivenih sa *featureExtraction-a*, povezanih u *featureMatching* funkciji, stvara strukturu scene. Ta struktura se sastoji od 3D točaka dobivenih putem preklapajućih dvodimenzionalnih fotografija snimljenih iz različitih poza. Skup tih točaka predstavlja oblak točaka našeg skeniranog objekta te će se koristiti za dobivanje cjelokupnog modela. Kao i u *featureMatching* čvoru, korišten je acransac

`localizerEstimator`.

Funkcija nakon završetka stvara datoteke s informacijama o oblaku točaka. Taj oblak točaka moguće je prikazati kroz grafičko sučelje ili putem aplikacija koje imaju mogućnost prikaza tih izlaznih datoteka. Jedan takav prikaz dostupan je na 3.2.



(a) Originalna slika

(b) Oblak točaka

Slika 3.2: Prikaz izlaza *structureFromMotion*

3.2.6. *prepareDenseScene*

Ova funkcija uklanja izobličenosti slika i od njih stvara nove EXR slike. EXR predstavlja rasterski format slika koji se vrlo često koristi u granama računalne grafike. Taj format dozvoljava pohranu više proizvoljnih kanala (RGB, alfa, difuzne...). [17]. Funkcija nakon izvršenja stvara EXR datoteke i konfiguracijsku datoteku.

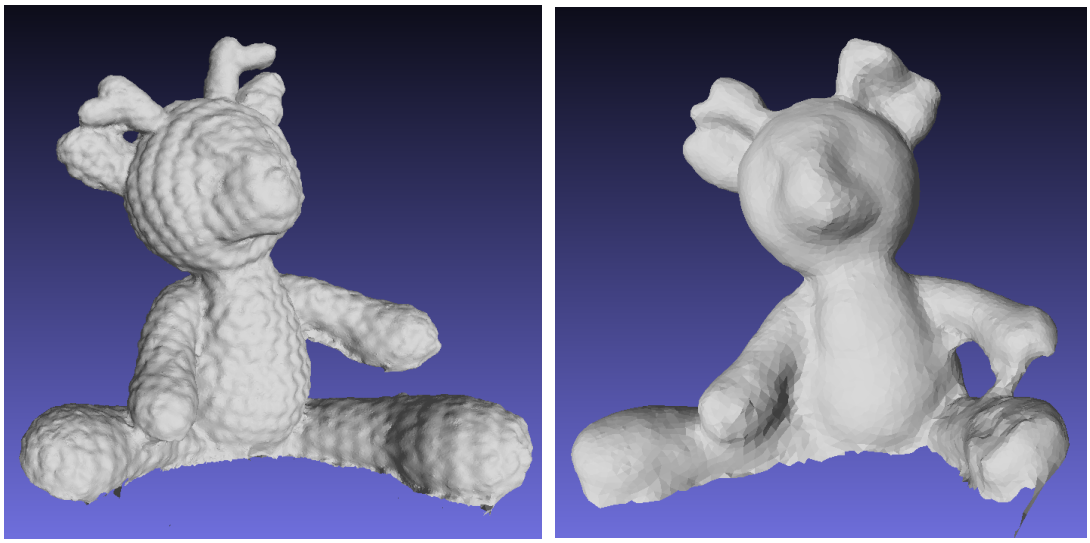
3.2.7. *depthMap*

DepthMap korak za sve kamere koje su razriješene sa *structureFromMotion* korakom pokušava dohvatiti dubinsku vrijednost za svaki pojedini piksel. Metoda provođenja je opisana u dokumentaciji za Meshroom [15].

Kao *featureExtraction* i *featureMatching* čvorovi šalje podatke u grupama. Parametar `downscale` predstavlja faktor skaliranja, te je postavljen u 2 kako bi se uz

malo pojednostavljenje smanjio teret na GPU, dok odabir vrijednosti veće od 2 rezultira nedovoljno preciznim modelom. Također, postavljeno je korištenje svih dostupnih GPU-ova postavljanjem parametra `nbGPUs` u 0.

Funkcija pri završetku stvara folder s generiranim dubinskim mapama.



(a) Faktor skaliranja = 2 **(b)** Faktor skaliranja = 8

Slika 3.3: Usporedba modela generiranih uz različite faktore skaliranja

3.2.8. *depthMapFilter*

Funkcija obrađuje dubinske mape dobivene *depthMap* korakom koje vrlo često nisu potpuno konzistentne, odnosno određene mape će tvrditi da vide područja koja su zaklonjena sa strane drugih karata. Ovaj korak će izolirati te dijelove i forsirati konzistenciju dubina. Broj lokacija kamere korištenih za filtraciju postavljen je na 10 sa parametrom `nNearestCams` jer je to srednja vrijednost intervala koji Meshroom nudi (0-20). Ovaj korak samo poboljšava već dostupne mape i sprema ih.

3.2.9. *meshing*

Funkcija stvara mesh od danih dubinskih mapa ili izlaza od *structureFromMotion*. Ako se iskorištava SfM izlaz za potrebe stvaranja mesh-a, proces će biti znatno brži, ali će model biti lošije kvalitete te je zbog toga odabir na korištenju dubinskih mapa. Taj odabir moguće je odrediti putem parametra `estimateSpaceFromSfM`.

Ulazni parametri `maxInputPoints` i `maxPoints` određuju broj točaka korištenih iz dubinskih mapa. Oni ovise o hardveru te je potrebno eksperimentirati s njima. Za ovaj projekt korištene su vrijednosti 100000 i 500000 zbog nedovoljne računalne snage.

Izlazi funkcije su densePointCloud.abc koji predstavlja datoteku s parametrima oblaka točaka i mesh.obj datoteka.

3.2.10. *meshFiltering*

Funkcija obrađuje izlaz iz *meshing* čvora s ciljem filtracije i spajanje trokuta koji su nastali greškom u prošlom koraku.

Sve osim najvećeg mesha se uklanja kako bi se filtrirala pozadina. Promjena te opcije moguća je postavljanjem parametra `keepLargestMeshOnly` u True. Rezultat ovog koraka je filtrirani mesh.

3.2.11. *texturing*

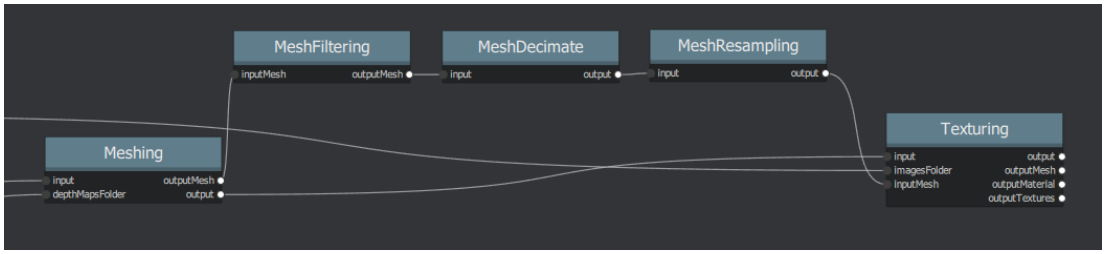
Funkcija stvara UV mape tekstura i projicira te teksture na stvoreni model. S obzirom na kompleksnost modela izlazna tekstura može biti spremljena u jednu ili više datoteka.

Postoji više metoda za generiranje UV mapa. Ako se ne koriste *meshDecimate* i *meshResampling* funkcije, najbolji odabir metode za razmotavanje mesha je Basic, a inače LSCM ili ABF. Postavljanje te metode moguće je putem parametra `unwrapMethod`. Funkcija također koristi faktor skaliranja, koji je isti kao u prošlim čvorovima, tj. 2.

Na izlazu funkcija stvara texturedMesh.obj, texturedMesh.mtl i datoteke s teksturama. Za potrebe ovog projekta odabran format datoteke s teksturama, `outputTextureFileType`, je PNG, a veličina izlazne teksture, `textureSide`, je 4096.

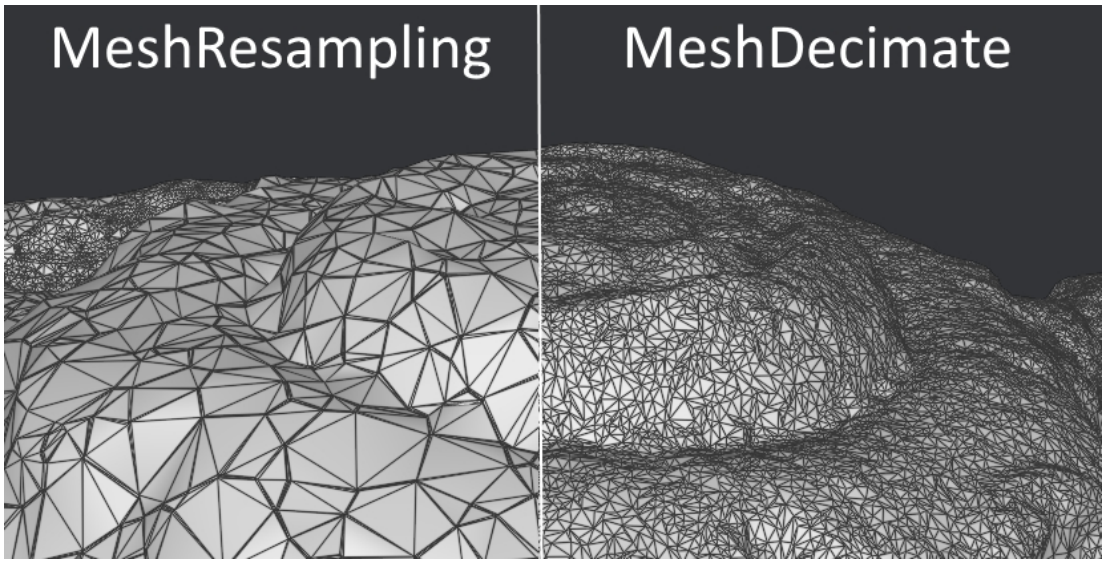
3.2.12. *meshDecimate* i *meshResampling*

Funkcije *meshDecimate* i *meshResampling* služe za uglađivanje i pojednostavljenje stvaranja 3D modela, ali ponekad to loše utječe na kvalitetu detalja. Slikoviti prikaz njihove funkcionalnosti dostupan je na 3.5 Te funkcije se izvode nakon *meshFiltering*, ali prije *texturing* čvora, kao što je vidljivo na 3.4



Slika 3.4: Prikaz dijela pipeline-a s funkcijama pojednostavljenja

Defaultno ove funkcije se ne koriste, a za njihovo korištenje potrebno je u skripti postaviti konstantu SIMPLIFY u True.



Slika 3.5: *MeshResampling* radi smanjenje broja lica uz održavanje cjelokupnog oblika, prikazanog na lijevoj slici, dok *meshDecimate* pojednostavljuje mesh bez promjene vizualnog izgleda modela, prikazano na desnoj slici

4. Primjena SimplyDigitize sustava za pripremu virtualne izložbe

U ovom poglavlju opisan je postupak primjene razvijenog sustava na primjeru kolekcije plišanih igračaka. Opisane su upute za korištenje, opisan je postupak skeniranja i izrade web stranice.

4.1. Upute za korištenje

U ovom poglavlju opisat će se postupak pokretanja skenera. Prvo će se opisati potrebni programi kako bi se skener mogao pokrenuti, a zatim i samo pokretanje.

Za početak, potrebno je računalo s Linux operacijskim sustavom te instaliranim ROS-om. Za Linux inačicu preporučuje se koristiti *Ubuntu 20.04.4 LTS*, a za ROS verziju *Noetic*.

Sav programski kod napisan u sklopu ovog rada nalazi se na *GitHub* repozitoriju te mu se može pristupiti preko sljedeće poveznice [18]. Tamo se nalaze i detaljne upute za instalaciju i samo pokretanja, što će ovdje biti samo ukratko opisano.

4.1.1. Instalacija potrebnog softvera

S poveznice [19] potrebno je u ROS radni prostor klonirati sljedeće pakete za upravljanje motorima:

- *dynamixel-workbench*
- *dynamixel-workbench-msgs*

– *DynamixelSDK*

Kloniranje paketa izvršava se pomoću naredbe *git clone*.

Zatim je, na isti način, potrebno klonirati *SimplyDigitize* paket koji se nalazi na poveznici [18]. Ovaj paket sadrži četiri Python skripte i *launch* datoteku, čije su funkcionalnosti detaljnije opisane u poglavlju 2.4.1.

Potrebno je i instalirati program *Meshroom* preko sljedeće poveznice [20]. On služi za izradu 3D modela iz prikupljenih slika skenera. Nakon što je instaliran, potrebno je putanju do *Meshrooma* dodati u datoteku *.bashrc* naredbom:

```
echo 'export LD_LIBRARY_PATH=/path_to_meshroom/Meshroom
/aliceVision/lib:$LD_LIBRARY_PATH' >> ~/.bashrc
```

gdje *path_to_meshroom* predstavlja put do Meshroom programa.

4.1.2. Pokretanje skeniranja

Ako je sve iz prethodnog poglavlja instalirano, skener je spreman za pokretanje. Objekt koji se želi skenirati treba postaviti u sredinu radnog prostora skenera te ga je potrebno dovoljno udaljiti od unutarnjeg ruba skenera tako da ne bude preblizu kamere kako bi slike bile kvalitetne. Sada se može spustiti zastor i prilagoditi parametre unutar *camera.py* čvora, što je detaljno opisano u poglavlju 2.4.2.

Samo pokretanje može se izvršiti ručno ili pomoću *launchfilea*. Kod ručnog pokretanja pokreće se svaki čvor u zasebnom prozoru terminala pomoću sljedećeg niza naredbi:

– `rosrun simply_digitize camera.py`

– `roslaunch dynamixel_workbench_controllers dynamixel_controllers.launch`

– `rosrun simply_digitize move_motor.py`

Nakon toga se pokreće postupak skeniranja objekta, a kada je skeniranje gotovo 3D izrada modela pokreće se sljedećom naredbom:

```
python3 Meshroom_CLI.py "Path_1" "Path_2" "Path_3"
```

- **Path_1** - Put do meshroom bin datoteke
- **Path_2** - Put do proizvoljne datoteke za spremanje izlaza
- **Path_3** - Put do slika

Drugi, jednostavniji način pokretanja je preko *launchfile-a*. Kako je prethodno već objašnjeno, *launchfile* služi za pokretanje više ROS čvorova zajedno pa se (umjesto gore navedenih naredbi) skeniranje može pokrenuti pomoću jedne naredbe:

```
roslaunch simply_digitize start_scanning.launch bin_path:="Path_1"
base_dir:="Path_2" img_dir:="Path_3"
```

- **Path_1** - Put do meshroom bin datoteke
- **Path_2** - Put do proizvoljne datoteke za spremanje izlaza
- **Path_3** - Put do slika

4.2. Skeniranje kolekcije ručno rađenih plišanih igračaka

Razvijeni sustav za digitalizaciju umjetničkih kolekcija demonstriran je na primjeru privatne kolekcije ručno rađenih plišanih igračaka, te se na isti način može napraviti digitalna kolekcija bilo kojih drugih predmeta.

Konkretno, kolekcija se sastoji od više plišanih igračaka prikazanih na slici 4.1. Ovakvi modeli pogodni su za skeniranje zbog svojih dimenzija koje su unutar granica koje skener može snimiti. Maksimalni promjer predmeta koje skener može snimiti iznosi 30 centimetara te maksimalna visina 60 centimetara.

Slika 4.1: Kolekcija plišanih igračaka

Prije početka skeniranja potrebno je model postaviti na povišeno postolje u središtu skenera, kao što je prikazano na slici 4.2. Također potrebno je spustiti rolo zastor, upaliti svjetla te spojiti računalo s motorima i pokrenuti skeniranje prema postupku opisanom u poglavlju 4.2.

Slika 4.2: Model postavljen za skeniranje

Parametri skeniranja postavljeni su tako da se snimi ukupno 84 slika snimljenih s kamerom postavljenom na tri različite visine. Trajanje skeniranja jednog modela s ovim brojem slika iznosi nešto manje od 10 minuta.

Nakon što je skeniranje modela završeno, automatski započinje 3D rekonstrukcija modela implementirana kao python skripta `Meshroom_CLI.py`, što je detaljnije opisano u poglavlju 3.

4.2.1. Prikaz rezultata skeniranja

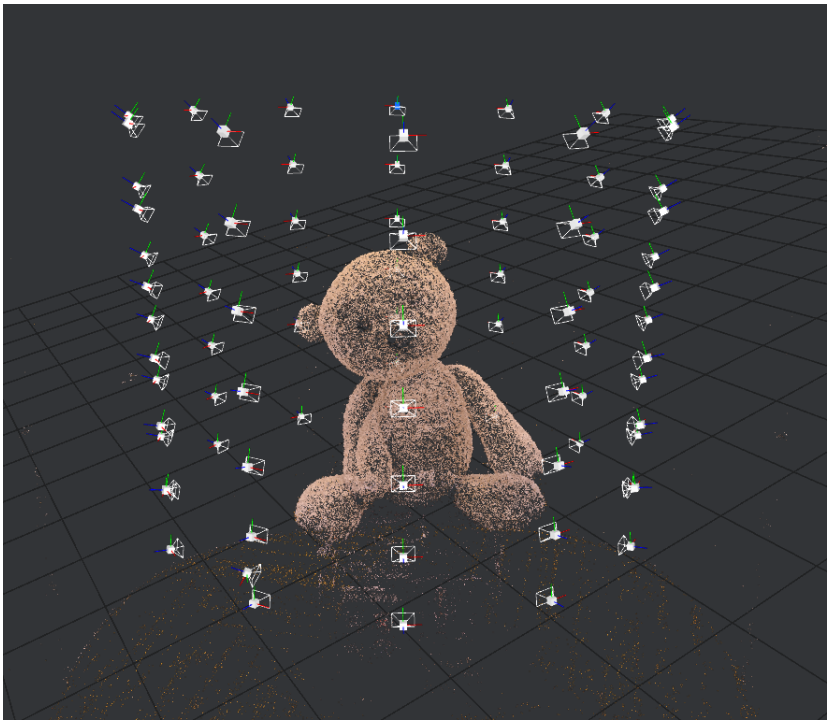
Slika 4.3 prikazuje jednu od 84 slike koje je skener prikupio tijekom procesa skeniranja. Oblak točaka generiran tijekom procesa 3D rekonstrukcija zajedno sa svim

pozicijama kamere iz kojih je model uslikan prikazan je na slici 4.4, a slika 4.5 prikazuje konačan rezultat procesa fotogrametrije.

Slika 4.3: Slika iz skenera

Konačni 3D modeli cijele kolekcije prikazani su na slici 4.6. Generirani 3D modeli plišanih igračaka imaju sve površine zatvorene, nijanse boja su slične stvarnima te modeli izgledaju realistično. Može se zaključiti da skener dobro funkcionira te da su digitalni modeli kolekcije plišanih igračaka uspješno izrađeni. Na ovaj bi se način mogla napraviti digitalizacija raznih predmeta poput umjetnina ili nekih drugih kolekcija predmeta.

Virtualna izložba generiranih 3D modela plišanih igračaka dostupna je na web stranici opisanoj u sljedećem poglavlju.

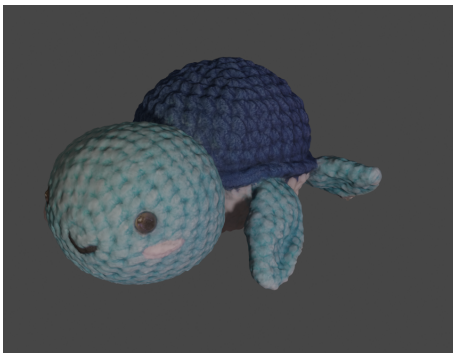


Slika 4.4: Oblak točaka generiran tijekom procesa fotogrametrije

Slika 4.5: Gotov 3D model medvjedića

(a) Jelen Jelena

(b) Medvjedić Lovro



(c) Morska kornjača Dominik

(d) Šeširić

(e) Medvjedić Matko

(f) Meduza Mirta

Slika 4.6: Prikaz 3D modela plišanih igračaka rekonstruiranih korištenjem razvijenog sustava za digitalizaciju umjetničkih kolekcija

4.3. Web stranica kao virtualna 3D izložba

Virtualne izložbe predstavljaju izrazito zanimljiv i koristan koncept. One pružaju jednostavan i ekonomski prihvatljiv način prikaza jedinstvenih primjeraka umjetnina bez potrebe za fizičkom prisutnošću. Također nude funkcionalno atraktivan i interaktivan model prikaza željenih objekata s ciljem promoviranja kulturnih mjesta širom svijeta putem pristupačnog kanala kao što je internet.

Sama ideja za korištenje web stranice kao platforme za izložbe nije u potpunosti nova. Mnoge izložbe različitih koncepata postoje relativno dugo, kao što je primjerice Louvre, jedan od najpoznatijih muzeja na svijetu, koji na svojoj web stranici nudi virtualni "hod" za razgledavanje muzeja, nalik na *Google-ov street view* [21]. Vrlo zanimljiv primjer je i *Museum of the world*. To je web stranica Britanskog muzeja koja pruža interaktivno sučelje s vremenskom crtom po kojoj je moguće kretati se i gledati primjerke u muzeju, kao i slušati zvučne zapise o objektima interesa [22].

Osim opisanih modela virtualnih izložbi, postoje i 3D interaktivne izložbe. One su relativno modernije zbog razvoja računalnih komponenti i tehnologija za 3D modeliranje. Autori [23] prikazuju različite metode i ostvarenja ovakvih 3D izložbi te analiziraju potrebu za stvaranjem izložbi istog tipa budući da su neki objekti uništeni ili se raspadaju od starosti. Ovakve izložbe pružaju najveću razinu interaktivnosti, ne samo zbog svojih svojstva pregleda u tri dimenzije, već i zbog pružanja kreativne slobode izlagačima za stvaranja prilagođenih virtualnih okolina, a time potencijalno i organizacije za buduće fizičke izložbe.

Za potrebe ovog rada napravljena je jednostavna web stranica s ciljem prikaza slika i modela predstavljenih u poglavlju 5. Glavna ideja stranice je prikaz svih dostupnih modela s njihovim slikama i nazivima. Klikom na svaku sliku otvara se prikaz 3D modela tog objekta.

Stranica je pisana koristeći open-source programska okruženja kao što su Vue.js i TroisJS koja omogućavaju jednostavnu implementaciju prikaza 3D modela. Vue.js je Javascript-ovo *front-end* programsko okruženje razvijeno za potrebe stvaranja korisničkih sučelja i aplikacija. TroisJS je također javno dostupno programsko okruženje

koje kombinira ThreeJS, Vue.JS i ViteJS, a korišteno je za prikaz 3D modela. [24].

Prije samog korištenja modela na stranici, potrebno je zadovoljiti neke preduvjete, odnosno modeli moraju biti u formatu **FBX** ili **GLTF** i tekstura mora biti ugrađena u datoteke modela.

Izgled početne stranice, prikazan na 4.7, dobiven je koristeći samo *front-end* radi jednostavnijeg stvaranja i smanjenja tereta na stranici. Iako je korišten samo *front-end*, stranica je skalabilna te dozvoljava dodavanje novih modela po potrebi.

Slika 4.7: Izgled stranice s galerijom modela

Zahvaljujući Vue.JS programskom okruženju i jednostavnoj sintaksi, korištenje TroisJS puno je jednostavnije od korištenja ostalih, sličnih programskih okruženja. Cijeli kod tog programskog okruženja predstavlja jedan stvoreni *renderer* unutar kojeg se dodaju elementi za prikaz. Osnovne komponente *renderer-a* su kamera i scena, koja može sadržavati model, osvjetljenje, pozadinu i slično. Osim jednostavnih modela i osvjetljenja, dozvoljava i akcije, primjerice rotaciju modela, kamere ili osvjetljenja. Za potrebe ovog rada korištena je vrlo jednostavna scena, koja se sastoji od sive pozadine, dva svjetla (ambijentalno i *point-light*) te našeg 3D modela. Implementirani prikaz 3D modela na stranici vidljiv je na slici 4.8. Važna napomena vezana uz prikaz

modela jest da vrijeme prikaza modela može varirati, a u prosjeku iznosi 8s.

Slika 4.8: Prikaz 3D modela

Heroku, javno dostupna platforma za razvoj aplikacija u oblaku, korištena je zajedno s GIT-om za postavljanje javno dostupne stranice. S obzirom na to da Heroku podržava Javascript programski jezik, realizacija je bila izrazito jednostavna. Pristup stranici dostupan je preko sljedeće poveznice korištenjem Google Chrome web browsera: https://simply-digitize.herokuapp.com/. [25]

5. Zaključak

U zadnjih nekoliko godina, sve popularniji postaje trend digitalizacije predmeta za različite primjene, od kojih su posebno istaknute virtualne galerije.

U ovom radu izrađen je sustav za 3D skeniranje predmeta, programska podrška i web stranica za prikaz galerije 3D modela. Projektirani su i 3D isprintani potrebni zupčanici, nosači motora i kamere koji osiguravaju kretanje kamere oko objekta. Ugrađeno je potrebno osvjetljenje i pozadina kako bi objekt bio bolje izdvojen od okoline i sjena uklonjena. Korištena su tri motora koja omogućuju fotografiranje objekata iz svih kuteva kako bi rekonstrukcija bila što uspješnija. Ručnim podešavanjem parametara skeniranja korisnik sustava samostalno bira broj fotografija i osigurava fotografiranje predmeta iz optimalnog broja kutova. Prikupljeni skup fotografija koristi se za 3D rekonstrukciju modela. Automatizirana 3D rekonstrukcija modela implementirana je kao python skripta iz koje se pozivaju pojedini koraci Meshroomovog pipelinea. Kvaliteta 3D modela, osim karakteristikama fotografija, određena je fotogrametrijskim parametrima.

Detaljno su opisani procesni koraci 3D rekonstrukcije i parametri potrebni za njeno ostvarenje. Ti procesni koraci služe za uspješno stvaranje kvalitetnih 3D modela prikazanih u radu.

Funkcionalnost sustava pokazana je na kolekciji plišanih igračaka. Kolekcija je uspješno digitalizirana i modeli vjerno prikazuju objekte iz stvarnog svijeta. Takvi modeli mogu se koristiti u različite svrhe, te je tako ostvaren željeni rezultat. Za prikaz rezultata izrađena je web galerija 3D modela ručno rađenih plišanih igračaka koja se nalazi na sljedećoj poveznici. [25]

LITERATURA

[1] *Skener 1K*. https://www.shining3d.com. Pristupano 20.6.2022.

[2] *Skener nK*. https://pangolin-editions.com/2019/01/01/senna-paul-oz/. Pristupano 20.6.2022.

[3] *GoPro*. https://gopro.com/en/us/. Pristupano 2.5.2022.

[4] *gopro-py-api*. https://github.com/konradit/gopro-py-api. Pristupano 29.3.2022.

[5] *ros.org*. https://www.ros.org/. Pristupano 12.6.2022.

[6] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system, 2009.

[7] *ROS 101: An Intro to the Robot Operating System*. https://www.designnews.com/gadget-freak/ros-101-intro-robot-operating-system. Pristupano 13.6.2022.

[8] *python.org*. https://www.python.org/. Pristupano 13.6.2022.

[9] Frederick Weinhaus and Venkat Devarajan. Texture mapping 3d models of real-world scenes. *ACM Comput. Surv.*, 29:325–365, 12 1997.

[10] *Artec scanners help create CGI models for TV series Fear the Walking Dead*. https://www.artec3d.com/cases/3d-models-for-cgi. Pristupano 14.6.2022.

[11] Nataska Statham, João Jacob, and Mikael Fridenfalk. Photogrammetry for game environments 2014-2019: What happened since the vanishing of ethan carter. 11 2020.

[12] *Photogrammetry - an overview*. https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/photogrammetry. Pristupano 10.6.2022.

[13] Nataska Statham. Use of photogrammetry in video games: A historical overview. *Games and Culture*, 15:155541201878641, 07 2018.

[14] *New work: Photogrammetry for the new Medicine Galleries at the Science Museum!* https://museuminabox.org/category/photogrammetry/. Pristupano 15.6.2022.

[15] Carsten Griwodz, Simone Gasparini, Lilian Calvet, Pierre Gurdjos, Fabien Castan, Benoit Maujean, Gregoire De Lillo, and Yann Lanthony. Alicevision Meshroom: An open-source 3D reconstruction pipeline. In *Proceedings of the 12th ACM Multimedia Systems Conference - MMSys '21*. ACM Press, 2021.

[16] T. Lindeberg. Scale Invariant Feature Transform. *Scholarpedia*, 7(5):10491, 2012. revision #153939.

[17] *OpenEXR*. https://www.openexr.com/. Pristupano 3.5.2022.

[18] *SimplyDigitize GitHub*. https://github.com/andro1999/SimplyDigitize.

[19] *ROBOTIS-GIT*. https://github.com/ROBOTIS-GIT. Pristupano 26.3.2022.

[20] *AliceVision*. https://alicevision.org/#meshroom. Pristupano 10.4.2022.

[21] *Visite Virtuelle | Petite Galerie - musée du Louvre*. https://petitegalerie.louvre.fr/article/visite-virtuelle. Pristupano 13.6.2022.

[22] *Museum of the World*. https://britishmuseum.withgoogle.com/. Pristupano 13.6.2022.

[23] Maria Carmo and A.P. Cláudio. 3d virtual exhibitions. *DJLIT- DESIDOC Journal of Library & Information Technology*, 33:222–235, 05 2013.

[24] Kevin Levron. Troisjs. https://github.com/troisjs/trois.

[25] *Virtualna galerija*. https://simply-digitize.herokuapp.com/. Pristupano 20.6.2022.

SAŽETAK

Mirta Čolić, Iva Hrastnik, Luka Karniš, Andro Katanec, Lovro Pliskovac, Dominik Sekula

SimplyDigitize - sustav za digitalizaciju zasnovan na fotogrametriji

Ubrzanim napretkom tehnologije razvila se potreba za digitalizacijom predmeta u tri dimenzije. Tijekom, ali i nakon, pandemije sve popularniji trend postaje trodimenzionalan prikaz predmeta za različite svrhe, pa tako i za virtualne izložbe umjetnina. Upotrebom 3D skenera moguće je predmet iz stvarnog svijeta digitalizirati tako da skener prikupi određen broj fotografija potrebnih fotogrametrijskom softveru za izradu 3D modela. U ovom radu predstavljena je realizacija sustava za digitalizaciju "SimplyDigitize" koji koristi ROS programersko okruženje za povezivanje GoPro kamere i Dynamixel motora za prikupljanje fotografija predmeta iz svih kutova, te Meshroom softver za izradu 3D modela iz niza prikupljenih slika.

Ključne riječi: 3d skener, fotogrametrija, virtualna izložba

ABSTRACT

Mirta Čolić, Iva Hrastnik, Luka Karniš, Andro Katanec, Lovro Pliskovac, Dominik Sekula

SimplyDigitize - digitalization system based on photogrammetry

In the age of advancing technology, the need for digitization of objects in three dimensions developed. During and after the global pandemic, three-dimensional object representation became more popular and is implemented for various use cases such as virtual exhibitions. With the 3D scanner, an object from the real world can be digitized by taking the required number of photographs, then based on taken photographs Meshroom software can create a 3D model of the given object. In this paper implementation of a scanner that uses ROS as an environment that connects the GoPro camera, the Dynamixel motors, and the Meshroom software is presented.

Keywords: 3d scanner, photogrammetry, virtual exhibition