Adaptivni sustav za preciznu manipulaciju objekata s robotskom rukom upotrebom promjenjivih kamera i ugradbene računalne platforme

Završki, Luka

Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

ZAVRŠNI RAD

Luka Završki

Zagreb , 2024

Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Filip Šuligoj, mag. ing.

Student:

Luka Završki

Zagreb , 2024

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem mentoru, doc. dr. sc. Filipu Šuligoju, na pomoći, savjetima i komentarima prilikom izrade ovog završnog rada.

Također, zahvaljujem svojim roditeljima i prijateljima na pruženoj podršci tijekom dosadašnjeg studiranja te cimeru Robertu na komentarima i savjetima o programiranju.

Luka Završki

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 24 – 06 / 1**	
Ur.broj: **15 – 24 –**	

ZAVRŠNI ZADATAK

Student: **Luka Završki** JMBAG: **0035235069**

Naslov rada na hrvatskom jeziku: **Adaptivni sustav za preciznu manipulaciju objekata s robotskom rukom upotrebom promjenjivih kamera i ugradbene računalne platforme**

Naslov rada na engleskom jeziku: **Adaptive system for precise manipulation of objects with a robotic arm using exchangeable cameras and embedded computing platform**

Opis zadatka:

Robotske ruke igraju kritičnu ulogu u širokom rasponu aplikacija, uključujući industrijsku proizvodnju, medicinsku kirurgiju i znanstvena istraživanja. Kroz napredak u područjima računalnog vida i umjetne inteligencije, ovakvi roboti su postali sve precizniji i autonomniji. Upravo stoga, ovaj rad nastoji usmjeriti fokus na implementaciju i optimizaciju jednog takvog sustava. U radu razmatrat će se uporaba platforme Jetson Nano za obradu podataka s ciljem realizacije optimizirane 2D detekcije objekata, a koji će u suradnji s robotskom rukom opremljenom hvataljkom obavljati specifične manipulacije objektima. U radu se posebno naglašava važnost kalibracije oko-ruka, procesa koji je presudan za uspostavljanje točne prostorne koordinacije između kamere i robotske ruke.

Zadaci:

- Implementirati komunikaciju između ugradbene Jetson Nano platforme, robotske ruke i različitih tipova kamera.
- Kreirati C++ aplikaciju koja omogućava intrinzičnu i ekstrinzičnu kalibraciju kamere. Također omogućiti oko-ruka kalibraciju putem kalibracijskog uzorka fiksiranog na hvataljku robota.
- Razviti i implementirati algoritme za 2D detekciju objekata.
- Oblikovati i konstruirati sustav koji uključuje robotsku ruku, Jetson Nano platformu i nosače koji su kompatibilni s različitim industrijskim kamerama.
- Provesti analizu performansi detekcije i manipulacije objekata, fokusirajući se na parametre kao što su točnost i robusnost.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
30. 11. 2023.	**1. rok:** 22. i 23. 2. 2024. **2. rok (izvanredni):** 11. 7. 2024. **3. rok:** 19. i 20. 9. 2024.	**1. rok:** 26. 2. – 1. 3. 2024. **2. rok (izvanredni):** 15. 7. 2024. **3. rok:** 23. 9. – 27. 9. 2024.

Zadatak zadao:

Doc. dr. sc. Filip Šuligoj

Predsjednik Povjerenstva:

Prof. dr. sc. Damir Godec

Sadržaj

Popis slika

Popis tablica

Sažetak

U ovom radu integrira se sustava upravljanja robotom i sustav strojnog vida s 2 kamere s Jetson Nano računalom. Odabrano je programiranje aplikacija u C++ programskom jeziku koji je efikasan i ima široku podršku za specijalizirane biblioteke kao što su OpenCV i ur_rtde. Te biblioteke omogućuju obradu i analizu slika, kalibraciju kamera, lokalizaciju predmeta i upravljanje UR robota u stvarnom vremenu.

Tijekom rada izrađuju se i prilagođena kućišta pomoću kojih se Jetson Nano računalo i dvije kamere montiraju na nosač, čime se osigurava kompaktnost i preglednost radnog prostora za kalibraciju i manipulaciju. Proces kalibracije obuhvaća, kalibraciju kamera i kalibraciju robota u konfiguraciji oko-ruka, što omogućuje robotu preciznu lokalizaciju i interakciju s objektima u radnom prostoru.

Nakon što su provedene sve kalibracije, razvijeni sustav omogućuje lokalizaciju predmeta na radnom stolu, korištenjem naprednih algoritama za obradu slike kao što su ORB i SIFT. Nadalje se predmeti vakuumskom hvataljkom odstranjuju iz radnog prostora. Ovaj rad demonstrira kako se napredne tehnologije mogu integrirati u funkcionalan i efikasan sustav za primjene u industriji, pri čemu se postiže robusnost i točnost u izvršavanju zadataka.

Ključne riječi: openCV, robot, ur_rtde, embeded računalo, C++, kamera, kalibracija, lokalizacija

Abstract

In this final thesis, a system integrating robot control and machine vision with two cameras and a Jetson Nano computer is developed. C++ was chosen as the programming language for application development due to its efficiency and strong support for specialized libraries such as OpenCV and UR RTDE. These libraries enable image processing and analysis, camera calibration, object localization, and real-time control of a UR robot.

During the project, custom-made cases were also designed and produced to mount the Jetson Nano computer and two cameras onto a holder, ensuring compactness and a clear workspace for calibration and manipulation. The calibration process covers both camera calibration and robot hand-eye calibration, allowing the robot to precisely localize and interact with objects in the workspace.

After completing all calibrations, the developed system allows for object localization on the table using advanced image processing algorithms such as ORB and SIFT. Objects are then removed from the workspace using a vacuum gripper. This thesis demonstrates how advanced technologies can be integrated into a functional and efficient system for industrial applications, achieving robustness and accuracy in task execution.

Keywords: OpenCV, robot, UR RTDE, embedded computer, C++, camera, calibration, localization.

1 Uvod

1.1 Kontekst i važnost istraživanja

Automatizacija i robotika postali su ključni u transformaciji suvremene industrije, omogućujući značajna poboljšanja u produktivnosti, kvaliteti i učinkovitosti proizvodnih procesa. Razvoj naprednih industrijskih sustava koji integriraju strojni vid i robotsku manipulaciju važan je za postizanje visokopreciznih zadataka, posebno u dinamičnim i kompleksnim okruženjima. Primjer upotrebe takvog integriranog sustava bio bi kod proizvodnje staklenih boca gdje se svakodnevno rade izmjene proizvoda, a gdje strojni vid ima važnu ulogu kod kontrole kvalitete proizvoda.

Jedan od izazova u ovom području je precizna kalibracija robota u konfiguraciji oko-ruka, koja omogućuje robotima točnu lokalizaciju i manipulaciju objekata u radnom prostoru. Jetson Nano računalo, poznato po svojoj snazi u obradi fotografija i niskoj potrošnji energije, predstavlja idealnu platformu za integraciju strojnog vida i robotske manipulacije. Kombinacija Jetson Nano računala s kamerama kao što su IMX477 B0250 i B0223 omogućuje izgradnju sustava koji može precizno lokalizirati objekte na radnom stolu, što je ključno za manipulaciju robotom.

U ovom radu nastoji se razviti sustav koji kombinira napredne algoritme računalnog vida s robotikom, kako bi se omogućila točna lokalizacija i manipulacija objekata.

1.2 Ciljevi rada

Cilj ovog rada je pružiti rješenje koje omogućuje integraciju Jetson Nano računala s robotom i dvjema kamerama, B0250 IMX 477 i B0223. Integracija omogućuje postizanja precizne kalibracije kamere i kalibracije robot u konfiguraciji oko-ruka. Ova kalibracija omogućuje robotskoj ruci točnu lokalizaciju i manipulaciju objekata u prostoru, što je ključno za precizne industrijske aplikacije.

Specifični ciljevi:

- Projektiranje i izrada kučišta za Jetson Nano računalo i dvije kamere, prilagođene za 3D printanje, koje će osigurati zaštitu i stabilnost sustava te omogućiti jednostavnu instalaciju i fleksibilnost u radu.

- Razvoj aplikacija u C++ jeziku, koristeći biblioteku OpenCV i ur-rtde, za robusnu i fleksibilnu integraciju lokalizacije predmeta i robotske manipulacije.
- Demonstracija cjelovitog rješenja koje integrira hardverske komponente i softverske alate u funkcionalni sustav za preciznu lokalizaciju i manipulaciju objekata u stvarnom okruženju.

1.3 Struktura rada

Rad je strukturiran na sljedeći način:

- **Uvod** – U ovom poglavlju može se vidjeti kontekst i važnost završnog rada, ciljevi i pristup rješavanju zadatka.
- **Pregled literature i oprema** – Pregled teorijske podloge i opreme relevantne za temu rada, uključujući, strojni vid i robotsku manipulaciju.
- **Metodologija i tehnički postav** – Opis metodološkog okvira, uključujući projektiranje i izradu kutija, postupke kalibracije sustava oko-ruka i kamera, te algoritme za detekciju i lokalizaciju objekata.
- **Rezultati i evaluacija rezultata** – Prikaz rezultata postignutih implementacijom i testiranjem sustava, te njihova analiza.
- **Zaključak** – Sažetak ključnih nalaza rada, te preporuke za buduće istraživanje i razvoj.

2 Pregled literature i oprema

2.1 Uvod u ugradbena računala [1]

Ugradbeni računalni sustav ili ugradbeno računalo (engl. *embedded system, embedded computers*) je kombinacija računalnog hardvera i softvera, a nekad i dodatnih dijelova, bilo mehaničkih ili elektroničkih dizajnirana za obavljanje određene funkcije. Dobar primjer je mikrovalna pećnica.

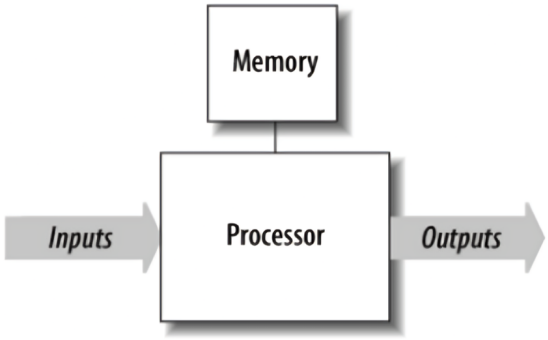
Povijest

Prvi takvi sustavi pojavili su se 1971. godine kada je Intel predstavio prvi mikroprocesor na svijetu s jednim čipom . Taj čip, 4004, dizajniran je za upotrebu u nizu poslovnih kalkulatora japanske tvrtke Busicom. Godine 1969., Busicom je zatražio od Intela da dizajnira set prilagođenih integriranih sklopova, po jedan za svaki od svojih novih modela kalkulatora. 4004 je bio Intelov odgovor. Umjesto dizajniranja prilagođenog hardvera za svaki kalkulator, Intel je predložio sklop opće namjene koji bi se mogao koristiti u cijeloj liniji kalkulatora. Ovaj procesor opće namjene dizajniran je za čitanje i izvršavanje skupa uputa softvera pohranjenih u vanjskom memorijskom čipu. Intelova ideja bila je da će softver svakom kalkulatoru dati jedinstven skup značajki i da će ovaj stil dizajna potaknuti potražnju za njegovom osnovnom djelatnošću u memorijskim čipovima. Mikroprocesor je bio uspjeh preko noći, a njegova se uporaba stalno povećavala tijekom sljedećeg desetljeća.

Neizbježno je da će se broj ugrađenih sustava nastaviti brzo povećavati. Već postoje obećavajuća nova ugradbena računala koja imaju golem potencijal na tržištu: svjetlosni prekidači i termostati koji su umreženi i kojima se može bežično upravljati putem središnjeg računala, inteligentni sustavi zračnih jastuka koji se ne napuhuju kada su prisutna djeca ili male odrasle osobe...

Česte komponente sustava

Da bi postojao software, mora postojati prostor za pohranu izvršnog koda i privremenu pohranu za manipulaciju izvršnim podacima. On je u obliku memorije samo za čitanje (ROM) odnosno memorije s izravnim pristupom (RAM); većina ugrađenih sustava ima neke od njih. Ako je potrebna samo mala količina memorije, može se nalaziti unutar istog čipa kao i procesora. Inače se jedna ili obje vrste memorije nalaze u vanjskim memorijskim čipovima. Svi ugrađeni sustavi također sadrže neku vrstu ulaza i izlaza što se može vidjeti na slici 2.1.



Slika 2.1: Komponente ugradbenog sustava [1]

Na primjer, u mikrovalnoj pećnici ulazni podaci su tipke na prednjoj ploči i temperaturna sonda, a izlazni podaci su zaslon i mikrovalno zračenje. Izlazni podaci ugrađenog sustava gotovo su uvijek funkcija njegovih ulaznih podataka i nekoliko drugih čimbenika (proteklo vrijeme, trenutna temperatura itd.). Ulazni podaci u sustav obično su u obliku senzora i sondi, komunikacijskih signala ili upravljačkih tipki. Rezultati su obično prikaz na zaslonu, komunikacijski signali ili promjene fizičkog svijeta.

2.2 NVIDIA Jetson Nano

Jetson Nano je malo, snažno računalo tvrke Nvidia koje služi za ugrađene aplikacije (eng. *embedded applications*) i UI internet stvari (eng. *AI Internet of Things*). Moguće je brzo krenuti s razvijanjem s JetPack SDK koji pruža potpuno razvojno okruženje s bibliotekama za duboko učenje, strojni vid, grafiku, multimediju i druge [2].

Tehničke specifikacije ugrađenog računala mogu se vidjeti u tablici 2.1.

Tablica 2.1: Tehnički podaci [2]

GPU	NVIDIA Maxwell arhitektura s 128 NVIDIA CUDA jezgrama
CPU	Quad-core ARM Cortex-A57 MPCore procesor
Memorija	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Pohrana	16 GB eMMC 5.1
Video kodiranje	250MP/sec 1x 4K @ 30 (HEVC) 2x 1080p @ 60 (HEVC) 4x 1080p @ 30 (HEVC) 4x720p @ 60 (HEVC) 9x720p @ 30 (HEVC)
Video dekodiranje	500MP/sec 1x4K @ 60 (HEVC) 2x4K @ 30 (HEVC) 4x1080p @ 60 (HEVC) 8x1080p @ 30 (HEVC) 9x720p @ 60 (HEVC)
Kamera	12 traka (3x4 or 4x2) MIPI CSI-2 D-PHY 1.1 (1.5 Gb/s po paru)
Povezivost	Gigabit Ethernet, M.2 Key E
Zaslon	HDMI 2.0 i eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B

2.3 Robotske ruke [3]

Robot (češki *Robot*, prema *robota*: tlaka, kmetski rad), automatizirani stroj višestruke namjene za obavljanje zadataka sličnih ljudskomu djelovanju. Naziv "robot" prvi put se pojavljuje u drami R. U. R. (*Rossum's Universal Robots*) 1920. Karel-a Čapek-a, koji koristi za opis čovjekolikoga stroja sposobnoga za rasuđivanje, a konstruiranoga kako bi zamijenio ljudski rad u tvornicama. Ta se predodžba o robotima zadržala sve do danas ali suvremeni roboti koji se koriste u praksi od nje se razlikuju.

Povijesni razvoj

Početak razvoja robota smješta se u polovicu 20. stoljeća kada je prvog robota projektirao američki izumitelj Georg Devol 1954. godine, a američka tvrtka Unimation proizvela ga je 1961. i nazvala Unimate. Izgled robota može se vidjeti na slici 2.2. Prototip je smješten na

proizvodnu liniju Gerenal Motors u New Jersey-u. Do 1961. Unimate serija 1900 postala je prva masovno proizvedena robotska ruka za automatizaciju tvornica gdje se većina robota koristila za tlačno lijevanje.

Slika 2.2: Unimate [4]

Konstrukcija robota

Robotski mehanizmi (robotska ruka, robot) je sustav koji se sastoji od većeg broja međusobno povezanih krutih tijela[1] (članaka/segmenata). Segmenti su međusobno povezani translacijskim ili rotacijskim zglobovima koje najčešće pokreću servo motori.

Robotima se upravlja kontrolom njihove kinematike[2] i dinamike[3] Za procese upravljanje koriste se brza računala koja mogu računati upravljačke varijable u realnom vremenu.

Primjena robota

Roboti se posebno koriste u okruženjima koja su opasna i nepristupačna za čovjeka. Tako imamo robote za razminiranje, varenje, istraživanje drugih planeta, robote koji rade u radioaktivnoj sredini...

Danas se najviše koriste u automobilskoj industriji za zavarivanje, sastavljanje i bojenje dijelova automobila. Primjena im se proširuje te ih možemo vidjeti kako kose travu, usisavaju prašinu ili pomažu kod operacija. Količina robota raste iz dana u dan, a najviše automatizirane zemlje su Sjeverna Koreja, Singapur i Njemačka.

2.3.1 Universal robot UR5

Universal Robots, danska tvrtka, specijalizirana je za proizvodnju kolaborativnih robota te ima sjedište u Odense-u. Osnovali su je Esben Østergaard, Kasper Støy, i Kristian Kassow 2005. godine. 2008. predstavili su UR5, prvog kobota koji može sigurno raditi uz zaposlenike. Kobot je eliminirao zaštitne ograde i kaveze te omogućio automatizaciju pogona i manjim

[1] Krutom tijelu se zanemaruje elastičnost i deformacije.

[2] Kinematika robota opisuje geometriju njihova gibanja, poziciju i orijentaciju pojedinih segmenata te brzine i ubrzanja.

[3] Dinamika robota opisuje njihovo dinamičko ponašanje što uključuje sile i momente.

proizvođačima s manje promjena u proizvodnim pogonima.Danas su svjetski lider na terenu s više prodanih kobota od svih konkurenata zajedno [5].

Kolaborativni roboti [6]

Koboti su roboti koji nisu ograničeni ogradama i kavezima te rade uz zaposlenike. Opremljeni su senzorima i osjetljivi na neočekivane situacije, imaju mogućnost zaustaviti se ukoliko se nađe na putu zaposlenika. To im omogućuje da rade uz ljude i nisu ograđeni.

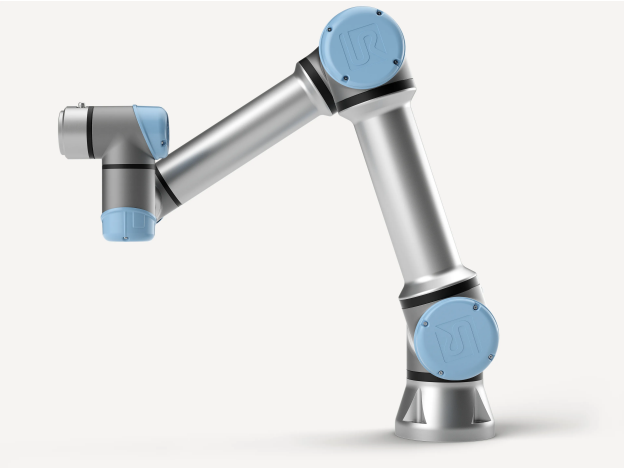
Namijenjeni su za izravnu interakciju s ljudima unutar zajedničkog prostora. Izrađeni su od laganih i oblih dijelova te imaju ograničene brzine i sile. Senzori kojim su opremljeni osiguravaju siguran rad s ljudima

Primjena [6]

Koriste se za doziranje, završnu obradu, sastavljanje, rukovanje materijalom, zavarivanje, itd.

UR5 [7]

UR5 je kolaborativan robot jednostavan za upotrebu te siguran u radu s ljudima, slika 2.3. Tablica 2.2 prikazuje osnovne specifikacije robota.



Slika 2.3: UR5 kobot [7]

Tablica 2.2: Specifikacije UR5 kobota [7]

Parametar	Vrijednost
Domet	850 mm
Nosivost	5 kg
Težina	18,4 kg
Stupnjevi slobode	6
Radna temperatura	0–50 °C
Ponovljivost	±0.1 mm
Komunikacija	TCP/IP 100Mbit, Modbus TCP, Profinet, EthernetIP

Sučelje za razmjenu podataka u stvarnom vremenu (engl. *Real-Time Data Exchange interface, RTDE interface*) omogućava sinkronizaciju vanjskih aplikacije s UR kontrolerom preko standardne TCP/IP veze što je bilo nužno za izradu ovog završnog rada. Više o tome u poglavlju 2.5.1 [8].

2.4 Strojni vid [9]

Strojni vid (engl. *Machine/Computer Vision, CV*) je interdisciplinarno znanstveno područje koje proučava kako računala mogu izvući korisne informacije iz slika ili videozapisa. Omogućuje računalima obavljanje zadataka poput prepoznavanja lica, detekcije, lokalizacije objekata, segmentacije slika. Cilj je razumjeti i automatizirati zadatke koje ljudski vizualni sustav može izvršavati.

Povijest

Larry Roberts 1960. u doktoratu na MIT-u raspravljao je o mogućnostima izdvajanja 3D geometrijskih informacija iz 2D perspektivnih pogleda na blokove poliedra. Istraživači su pratili taj rad te računalni vid proučavali u kontekstu svijeta blokova. Kasnije su shvatili da je potrebno proučavati slike iz stvarnog života te su se bacili na niže razine računalnog vida kao što su otkrivanje rubova i segmentacija.

Veliki utjecaj imao je i David-a Marr-a 1978. koji je predložio pristup "odozdo prema gore" za razumijevanje scene. Pristup je podrazumijevao primjenjivanje algoritama nižeg stupnja na 2D slike kako bi se dobila 2.5D skica. Tehnike visoke razine koriste se za dobivanje 3D modela iz skice.

Marr-ov program je teško provesti no za mnogo aplikacija računalnog vida nije potrebno dobiti potpune 3D modele. Na primjer u nekim aplikacijama potrebno je saznati da li se objekt odmiče ili približava za što nije potreban potpuni 3D model objekta.

Primjena

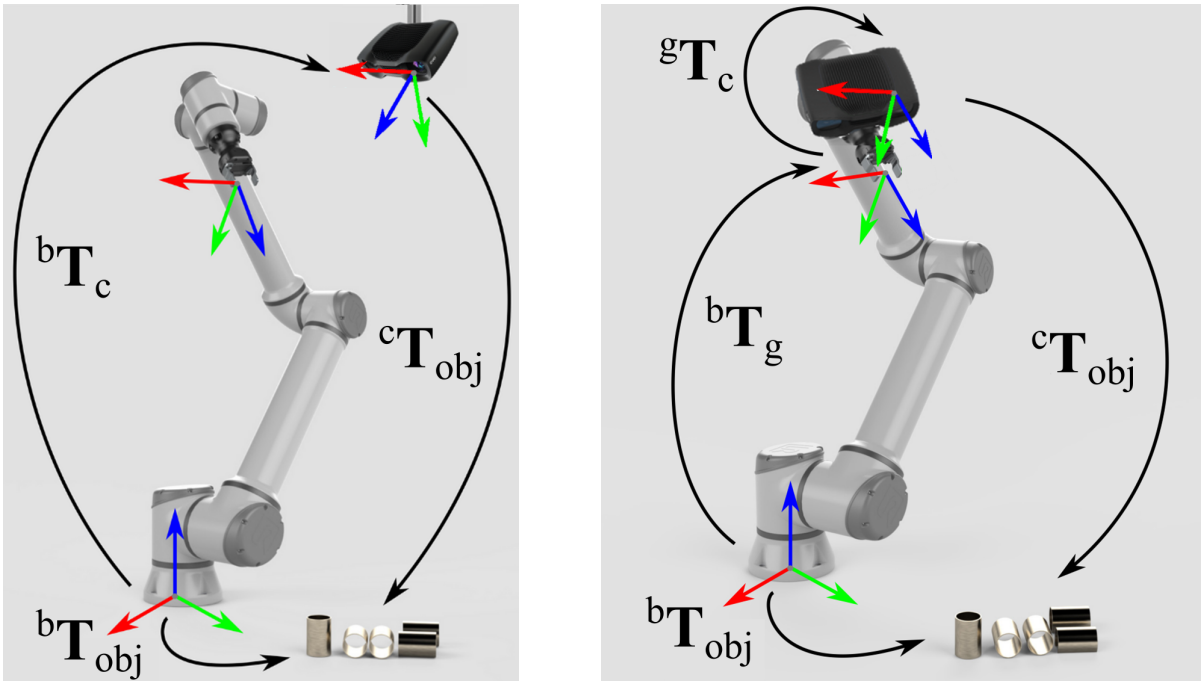
Najistaknutije područje primjene je u medicini kod obrade slika gdje se mogu izvući informacije koje služe za otkrivanje tumora i drugih bolesti. Također se koristi u autonomnim vozilima, kod kontrole kvalitete i montaže proizvoda. Služi i za mjerenje položaja i orijentacije objekata s kojima dalje manipulira robotska ruka.

2.4.1 Robotski vid

Bez robotskog vida roboti striktno izvršavaju kod što nije problem za određene zahvate. Robotski vid (engl. *Robot Vision*) je sustav koji se sastoji od kamere i računala koje obrađuje podatke dobivene kamerom te upravlja robotom i omogućava mu fleksibilniji rad [10].

Razlikujemo **dvije konfiguracije** robota i kamere:

- Oko–ruka (slika 2.4a)
- Oko–u–ruci (slika 2.4b)



(a) Oko ruka (b) Oko u ruci

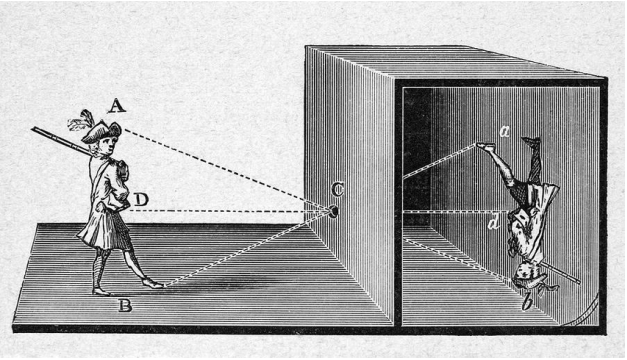
Slika 2.4: Konfiguracije robota i kamere [11]

Kod **oko–ruka** konfiguracije kamera je postavljena na kućište neovisno o robotu, kamera gleda ruku robota, slika 2.4a.
Oko–u–ruci konfiguracija predstavlja konfiguraciju gdje robot sadrži kameru najčešće na zadnjem segmentu, slika 2.4b.
Detaljan opis i kalibracija oko–ruka konfiguracije za potrebe ovog rada obradi će se u poglavlju 3.3.2

2.4.2 Kamera [12]

Kamera predstavlja jedan od najmoćnijih senzora dostupnih robotima. Formiraju 2D slike na senzoru i sadrže informacije o 3D sceni. Te se informacije mogu analizirati pomoću algoritama računalnog vida koje će se razraditi u poglavlju 3.3.1.

Camera obscura (lat. *tamna soba*) je, prethodnica kamere, kutija u koju kroz mali otvor ulazi svjetlost i stvara sliku na drugoj strani kutije. Model se može vidjeti na slici 2.5. Camera obscura fokusira cijelu scenu ali propušta malu količinu svjetlosti te slike koje nastaju su zatamnjene, slika 2.6a. Rješenje tog problema je korištenje objektiva koji prikuplja svjetlost preko većeg promjera i fokusira svjetlost na senzoru slike, slika 2.6b. Nedostatak korištenja objektiva je pojava dubine polja (engl. *depth of field*) što predstavlja gubitak fokusa cijele scene, tj. samo dio scene može biti u fokusu u određenom trenutku. Objektivi također



Slika 2.5: Camera obscura [13]

geometrijski izobličuju sliku, stoga je potrebno primijeniti algoritme za korekciju distorzije kako bi se dobile točne informacije za primjenu u robotici.

(a) Slikano pomoću modela camere obscure (b) Slikano kamerom mobitela

Slika 2.6: Usporedba slika

Najvažnije svojstvo digitalnih kamera je njezina rezolucija koja se najčešće označuje kao $u \times v$ (širina puta visina) u pikselima. Žarišna duljina mjeri se u metrima ili pikselima, a dobiva se iz intrinzičnih parametara kamere. Kada je jednom izračunata može se koristiti za sve slike slikane tom kamerom i s tim fokusom.

Kombiniranjem dvije kamere može se procijeniti 3D informacije iz dvije 2D slike. Takvo kombiniranje nazivamo stereokamera. Stereokamera je kamera s dva jednaka objektiva koja također može sadržavati i druge senzore, slika 2.7 prikazuje stereokameru koja se primjenjuje u robotici. Objektivi su pomaknuti jedan u odnosu na drugog te se njome slikaju dvije slike istog objekta što nazivamo stereoparom. Stereopar se također može izvesti kamerom s jednim objektivom ali je objekt potrebno slikati s dva različita položaja.

Slika 2.7: ZED stereokamera [14]

Geometrija kamere

Točke iz 3D svijeta projiciraju se u točke na 2D ravninu senzora.

Najjednostavniji model geometrijskog formatiranja slike je pinhole kamera (engl. *pinhole camera*). Pinhole kamera je jednostavana kamera bez objektiva koja koristi mali otvor kao blendu. Slika koja nastaje pomoću te kamere rezultat je efekta camera obscura, pri čemu se slika projicira kroz mali otvor.

Standardni koordinatni sustav kamere u strojnom vidu je desnokretni gdje:

- X-os pokazuje u *desno*,
- Y-os pokazuje *dolje*, a
- Z-os pokazuje u *scenu* (slika 2.8).

Slika 2.8: Koordinatni sustav kamere [15]

Mapiranje iz koordinata svijeta u koordinate slike

Model pinhole kamere definira da će se 3D točka $P_w = (X_w, Y_w, Z_w)$ projicirati na ravninu slike (engl. *image plane*) iza kamere. Zbog jednostavnosti ravninu slike možemo pomaknuti u žarište $z = f$ ispred kamere. Slika koja nastaje na toj ravnini je uspravna te olakšava pisanje formula.

Točka iz koordinata kamere $P_c = (X_c, Y_c, Z_c)$ se projicira na ravninu slike $p = (x, y)$ preko:

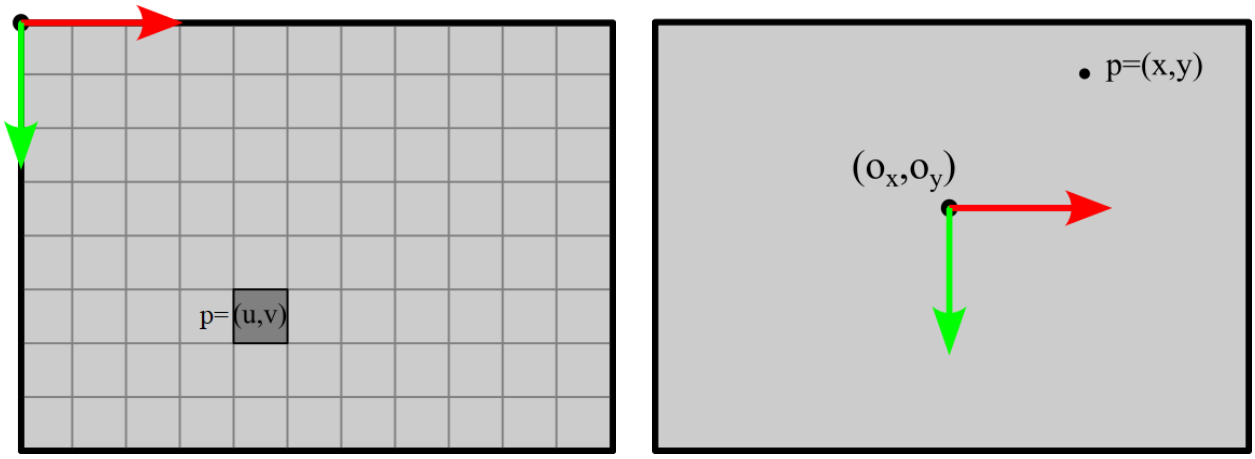
$$x = f\frac{X_c}{Z_c}, \quad y = f\frac{Y_c}{Z_c}. \tag{2.1}$$

Ukoliko želimo projicirati na senzor slike (engl. *image sensor*) $p = (u, v)$ žarište je potrebno izraziti u pikselima te se ono označava (f_x, f_y). Pravokutni oblik piksela je razlog zašto

koristimo x i y način označivanja žarišta. Žarište izraženo u pikselima možemo izračunati preko gustoće piksela (m_x, m_y) [piksel/mm] gdje je $(f_x, f_y) = (m_x f, m_y f)$ [piksel].

$$u = f_x \frac{X_c}{Z_c}, \quad v = f_y \frac{Y_c}{Z_c}. \tag{2.2}$$

Razlika između ravnine slike i senzora slike je u ishodištu koordinatnog sustava. Ishodište koordinatnog sustava senzora slike je u gornjem lijevom kutu (slika 2.9a), te se koordinate točke $p = (u, v)$ izražavaju u pikselima. Važno je napomenuti da pikseli senzora mogu biti pravokutni, a ne kvadratni, što ima utjecaj kod rekonstukcije 3D scene. Ravnina slike ishodište koordinatnog sustava sadrži u optičkom središtu te se koordinate točke $p = (x, y)$ izražavaju u milimetrima (slika 2.9b).



(a) senzor slike (b) ravnina slike

Slika 2.9: Usporedba koordinatnih

Slika 2.10 prikazuje slikovnu reprezentaciju mapiranja točke iz 3D svijeta u 2D sliku. C je koordinatni sustav kamere kojoj Z os (Z_c) ujedno predstavlja optičku os koja prolazi kroz središte ravnine slike tj. optičko središte (o_x, o_y) . I predstavlja koordinatni sustava senzora slike gdje se točka iz svijeta projicira u točku $p = (u, v)$.

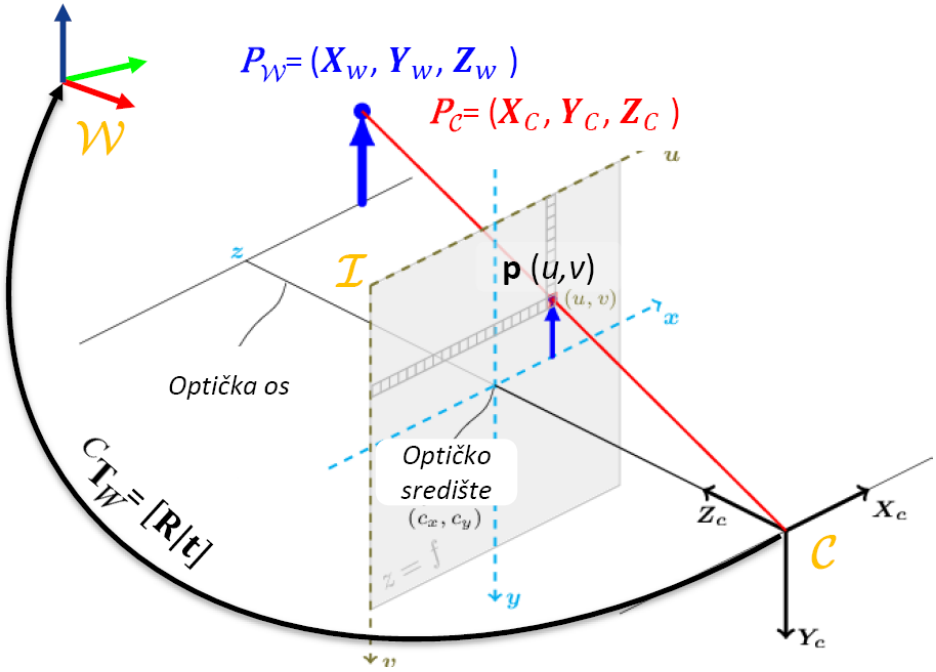
Koordinatni sustav svijeta označen je s W. Koordinatu točke ${}^{C}\mathbf{p}$ u koordinatnom sustavu kamere C možemo dobiti množenjem homogene transformacije[4] ${}^{C}\mathbf{T}_{W}$ s točkom u globalnom koordinatnom sustavu ${}^{W}\mathbf{p}$.

$$ {}^{C}\mathbf{p} = {}^{C}\mathbf{T}_{W} \cdot {}^{W}\mathbf{p} \tag{2.3}$$

Ukoliko želimo dobiti koordinatu točke ${}^{W}\mathbf{p}$ u globalnom koordinatnom sustavu W potrebno je invertirati homogenu matricu ${}^{C}\mathbf{T}_{W}$:

[4]Ovakav zapis homogene transformacije podrazumijeva transformaciju koordinatnog sustava svijeta u koordinatnom sustavu kamere

$$
\begin{aligned}
{}^{C}\mathbf{T}_{W}^{-1} &= {}^{W}\mathbf{T}_{C}, \\
{}^{W}\mathbf{p} &= {}^{W}\mathbf{T}_{C} \cdot {}^{C}\mathbf{p}.
\end{aligned}
\tag{2.4}
$$



Slika 2.10: Model projekcije[16]

Jednadžbi 2.2 pribrajamo optičko središte (o_x, o_y) zbog toga jer glavna točka može biti pomaknuta od idealne pozicije $(0, 0)$, središta senzora. 2.5 jednadžbe predstavljaju nelinearni model perspektivne projekcije.

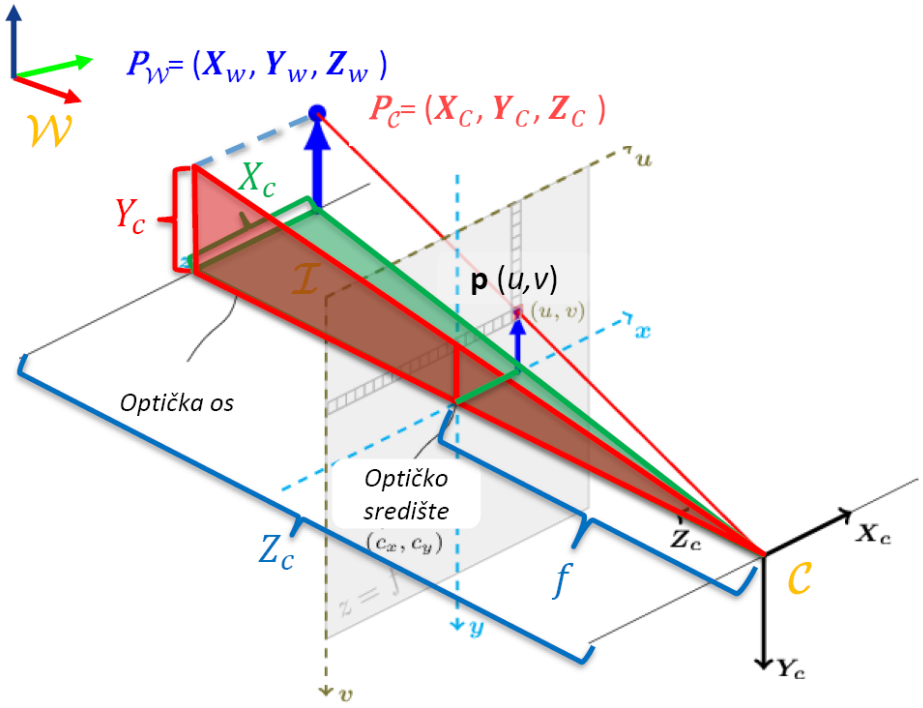
$$
\begin{aligned}
u &= f_x \frac{X_c}{Z_C} + o_x, \\
v &= f_x \frac{Y_c}{Z_C} + o_y
\end{aligned}
\tag{2.5}
$$

Do nesavršenosti dolazi kod instalacije senzora u kameru, nije moguće senzor pozicionirati savršeno u sredinu. Ti su pomaci relativno mali u odnosu na matematičko središte koje se dobiva prepolavljanjem rezolucije senzora. Matematičko središte senzora rezolucije 1920×1080 iznosilo bi $(960, 540)$.

Jednadžbe 2.5 mogu se izvesti iz sličnosti trokuta preko slike 2.11.

Nelinearni model perspektivne projekcije može se linearizirati upotrebom homogenih koordinata koje nam omogućavaju translaciju i rotaciju 2D koordinata:

$$
\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \equiv \begin{bmatrix} Z_c u \\ Z_c v \\ Z_c \end{bmatrix} = \begin{bmatrix} f_x x_c + z_c o_x \\ f_x x_c + z_c o_x \\ z_c \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}. \tag{2.6}
$$



Slika 2.11: Model projekcije[16]

Matrica:

$$\mathbf{M}_{int} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.7}$$

u jednadžbi 2.6 predstavlja **intrinzičnu matricu**, a ukoliko uklonimo 4. stupac dobivamo **matricu kalibracije**. Intrinzični parametri ostaju isti za različite poglede na scenu i specifičnu kameru, a služe za uklanjanje distorzije uzrokovane lećama.

Ekstrinzični parametri kamere su rotacija **R** i translacija **t** koordinatnog sustava kamere u koordinatnom sustavu svijeta.

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.8}$$

Matrica rotacije je ortogonalna matrica što znači da joj je inverz jednak transponiranoj matrici.

Ekstrinzična matrica se stvara od matrice rotacije i vektora translacije:

$$\mathbf{M}_{ekst} \equiv {}^{C}\mathbf{T}_{W} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Upotrebom ovih matrica mogu se izračunati koordinate kamere preko koordinata svijeta i koordinate kamere u piksele.

Kamera u piksele:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \mathbf{M}_{int} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{2.10}$$

Svijet u kameru:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \mathbf{M}_{ekst} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.11}$$

2.4.3 Arducam B0223 OV9281 1MP [17]

Slika 2.12: Arducam B0223 OV29281 1MP [17]

Kamera posjeduje visoke performanse, globalni okidač (endl. *global shutter*) i dobru osjetljivost na slabo osvjetljenje, što je čini pogodnom za različite aplikacije u industriji, slika 2.12.

Na Jetson Nano potrebno je instalirati upravljački program (engl. *driver*) za kameru kako bi kamera ispravno funkcionirala. Važno je napomenuti, ukoliko je instaliran upravljački program za drugu kameru, instalacijom ovoga drugi se briše.

Tablica 2.3: Ključne specifikacije Arducam B0223 OV9281 kamere [17]

Senzor	Monokromatski globalni zatvarač OV9281
Veličina piksela	3 µm x 3 µm
Optička veličina	1/4 inča
Raspon fokusa	30 mm – beskonačno
Izlazno sučelje	2-lane MIPI serijski izlaz
Izlazni formati	8/10-bit crno-bijeli RAW
Rezolucija i broj slika u sekundi	1280×800@80fps, 1280×720@80fps, 640×400@240fps
Maksimalna brzina prijenosa slike	1280 x 800@120 fps

2.4.4 B0250 IMX 477 12 MP [18]

Slika 2.13: IMX477 kamera [18]

B0250 IMX477 – ArduCam kamera sadrži IMX477, senzoru u boji, tvrtke Sony, koji je poznat po svojoj visokoj rezoluciji i oštrini (slika 2.13). To ga čini idealnim za primjene gdje su potrebne jasne i detaljne slike. Kamera dolazi u kompletu s CS lećom i industrijskim metalnim kućištem, što omogućuje jednostavnu upotrebu i montažu. Uz podršku za Nvidia Jetson platforme, ova kamera odabrana je za potrebe izrade ovog rada. Ostale specifikacije mogu se vidjeti u tablici 2.4.

Tablica 2.4: Ključne specifikacije IMX477 kamere [18]

Senzor	IMX477 senzor u boji
Veličina piksela	1.55 µm x 1.55 µm
Optička veličina	1/2.3 inča
Maksimalna rezolucija	4056 x 3040 (12.3 MP)
Tip zatvarača	Rolling shutter
Rezolucija i broj slika u sekundi	4032×3040 @ 10fps, 3840×2160 @ 20fps, 2592×1944 @ 30fps 2560×1440 @ 30fps, 1920×1080 @ 60fps, 1600×1200 @ 50fps 1280×960 @ 100fps, 1280×760 @ 100fps, 640×480 @ 80fps
Izlazno sučelje	4-lane MIPI CSI-2
Filter	IR-cut filter (nije prikladno za infracrvenu fotografiju)

2.5 C++ [19]

C++ je programski jezik visoke razine i opće namjene, kojeg je razvio danski informatičar Bjarne Stroustrup. Prvi put je predstavljen 1985. godine kao proširenje jezika C. Tijekom godina se znatno proširio, a od 1997. godine uključuje objektno orijentirane, generičke i funkcionalne značajke. Pogodan je za izradu operativnih sustava kao što su Linux i Windows i programiranje mikroračunala.

Slika 2.14: C++ logo

Stroustrup je započeo razvoj C++ jezika u Bell Labs-u 1979. godine, želeći stvoriti jezik koji je učinkovit i fleksibilan poput C-a, ali s dodatnim naprednim značajkama za bolju organizaciju programa.

2.5.1 UR rtde biblioteka [8]

Slika 2.15: Ur_rtde logo

RTDE (Real-Time Data Exchange) biblioteka je specijalizirana biblioteka koja omogućava komunikaciju između robota i aplikacija u stvarnom vremenu. Ova biblioteka je dio ekosustava Universal Robots i omogućava korisnicima efikasno razmjenjivanje podataka s robotima, što olakšava razvoj aplikacija za automatizaciju i robotiku.

Real-Time Komunikacija (RTDE) omogućava razmjenu podataka u stvarnom vremenu, što je ključno za aplikacije koje zahtijevaju brze reakcije i precizno upravljanje robotima.

2.5.2 OpenCV biblioteka [20]

OpenCV, skraćeno od Open Source Computer Vision Library, je otvorena biblioteka za računalni vid i strojno učenje koja se koristi za obradu slika i videa. Prvotno razvijena od strane Intela, OpenCV se danas održava od strane zajednice programera pod OpenCV Foundation. Ova biblioteka je postala standardni alat za mnoge aplikacije vezane uz računalni vid i nudi širok spektar funkcionalnosti koje omogućuju prepoznavanje objekata, analizu slika, praćenje pokreta i još mnogo toga.

Slika 2.16: OpenCV logo

Sadrži bogat skup algoritama koji pokrivaju širok spektar aplikacija, uključujući prepoznavanje lica, segmentaciju slika i analizu objekata. Biblioteka je optimizirana za rad u stvarnom vremenu, što je ključno za aplikacije kao što su autonomna vozila, robotske tehnologije i sigurnosni sustavi.

3 Metodologija i tehnički postav

3.1 Postav sustava

Sustav za manipulaciju objekata razvijen na Jetson Nano računalu integrira napredne algoritme za lokalizaciju objekata (ORB i SIFT) pomoću OpenCV biblioteke, koji omogućuju precizno prepoznavanje lokacije objekata na radnom stolu. Jetson Nano komunicira s robotom putem lokalne mreže s TCP/IP protokola, što omogućuje upravljanje UR robotom. Upravljanje robotom realizirano je korištenjem ur_rtde biblioteke, koja osigurava brzu i pouzdanu komunikaciju između Jetson Nano uređaja i UR robota. Aplikacije su razvijene u C++ programskom jeziku, što osigurava visoke performanse i efikasnost u obradi slike i upravljanju robotskim manipulatorom. Postav sustava može se vidjeti na slici 3.1.

Slika 3.1: Postav sustava

3.2 Postavljanje Jetson Nano uređaja

Uz Jetson Nano 4GB, potrebno je imati memorijsku karticu microSD s minimalno 32Gb pohrane i adapterom koji omogućava upotrebu s računalom. Također je potrebno imati miš, tipkovnicu te pristup internetu preko mrežnog kabla budući da uređaj nema bežični mrežni adapter (slika 3.2).

Operativni sustav [21]

Kako bi se instalirao operativni sustav na Jetson Nano, potrebno je:

- skinuti sliku Jetpack-a koji podržava uređaj. Odabere se verzija (Jetpack 4.6.1) i skine na stolno računalo.

Slika 3.2: Jetson Nano

- Formatirati memorijsku karticu koristeći program SD Memory Card Formatter.
- Stvoriti sliku na memorijskoj kartici koristeći program belena Etcher.

Prebacivanjem memorijske kartice u Jetson Nano, postavlja se operativni sustav. Pokretanjem uređaja nužno je uspostaviti internetsku mrežu te krenuti s instalacijom potrebnih programa i paketa.

Programi i paketi

Paket v4l-utils sadrži niz alata za upravljanje kamerama na Linux sustavima. Ti alati su korisni za testiranje, uklanjanje pogrešaka i konfiguriranje video uređaja. Može se instalirati slijedećom naredbom.

```
$ sudo apt-get install v4l-utils
```

Potrebno je instalirati upravljački program za kamere koje se koriste u ovom radu iz poglavlja 2.4.4 i 2.4.4. Kao što je navedeno prije, nije moguće imati upravljački program za obje kamere odjednom jer se oni sukobljavaju te je potebno iznova instalirati upravljački program za kameru koja se koristi.

Aplikacije se programiraju u Qt Creator-u, u C++ jeziku, što je integrirano razvojno okruženje koje pojednostavljuje razvoj aplikacija.

UR rtde biblioteka [22]

Kod instalacije ove biblioteke treba pripaziti koja se verzija operativnoga sustava koristi na Jetson Nano uređaju jer biblioteka trenutno podržava:

- Ubuntu 16.04 (Xenial Xerus),

- Ubuntu 18.04 (Bionic Beaver) i
- Ubuntu 20.04 (Focal Fossa) za Linux.

```
$ sudo add-apt-repository ppa:sdurobotics/ur-rtde
$ sudo apt-get update
$ sudo apt install librtde librtde-dev
```

Proces izgradnje softvera:

```
$ git clone https://gitlab.com/sdurobotics/ur_rtde.git
$ cd ur_rtde
$ git submodule update --init --recursive
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
```

Cmake primjer za pronalaženje biblioteke:

```
cmake_minimum_required(VERSION 3.5)
project(ur_rtde_cmake_example)

find_package(ur_rtde REQUIRED)
add_executable(ur_rtde_cmake_example main.cpp)
target_link_libraries(ur_rtde_cmake_example PRIVATE ur_rtde::rtde)
```

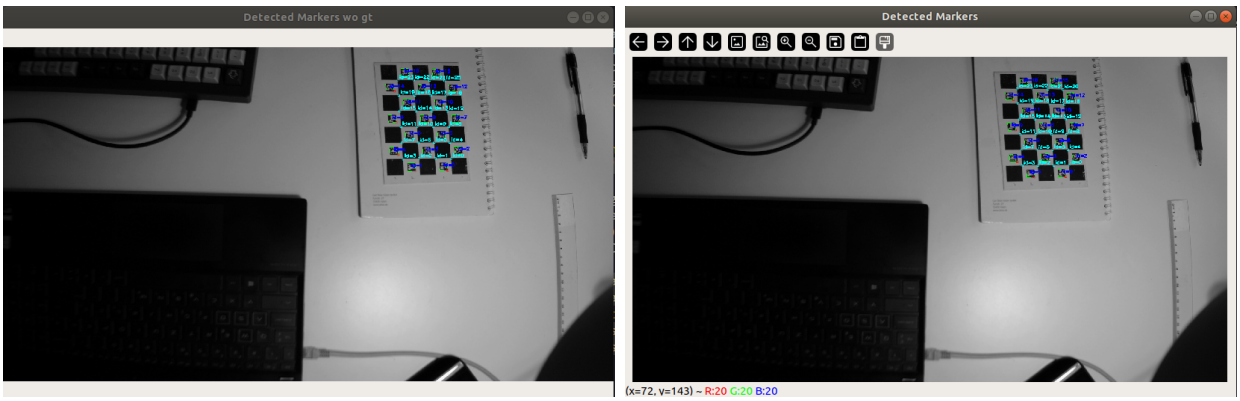
OpenCV biblioteka [23]

Potrebno je kalibrirati kameru što se izvodi s ChAruco pločom za što su nam potrebni dodatni moduli koji ne dolaze sa službenom OpenCV bibliotekom već OpenCV-contrib.

Jetpack 4.6.1 već ima instaliran OpenCV 4.1.1. Ova verzija ne podržava sve potrebne značajke. Za potpunu funkcionalnost potrebno je instalirati OpenCV 4.9 i OpenCV contrib.

Prije instalacije OpenCV-a potrebno je instalirati ostale pakete. Qt 5 je alat za razvoj grafičkog korisničkog sučelja. Služi za stvaranje privlačnijih grafičkih elemenata kao što su klizači i okviri za potvrdu u OpenCV aplikacijama. Nije obavezan za funkcioniranje OpenCV-a te njegova upotreba može dovesti do smanjenja brzine programa. Na slici 3.3 može se vidjeti kako grafičko sučelje izgleda sa i bez Qt5 alata.

Može se instalirati slijedećom naredbom, a u OpenCV uključiti pomoću zastavice -D WITH_QT



(a) Bez Qt5 (b) sa Qt5

Slika 3.3: Usporedba grafičkog sučelja

```
$ sudo apt-get install qt5-default
```

Instalacija OpenCV-a zauzima približno 8.5GB pohrane.

Preuzimanje i raspakiravanje:

```
$ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.9.zip
$ wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.9.zip
$ unzip opencv.zip
$ unzip opencv_contrib.zip
```

Prije izgradnje biblioteke potrebno je napraviti novu mapu ”build” u kojoj će se spremiti sve datoteke izgradnje.

```
$ cd ~/opencv
$ mkdir build
$ cd build
```

Slijedećim naredbama i zastavicama određuje se gdje i kako izraditi OpenCV biblioteku. Ovdje se unese zastavica -D WITH_QT=ON za grafičko korisničko sučelje visoke razine.

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D OPENCV_EXTRA_MODULES_PATH=../opencv_contrib-4.9/modules ../opencv-4.9
-D EIGEN_INCLUDE_PATH=/usr/include/eigen3 \
-D WITH_OPENCL=OFF \
-D WITH_CUDA=ON \
-D CUDA_ARCH_BIN=5.3 \
-D CUDA_ARCH_PTX="" \
```

```
-D WITH_CUDNN=ON \
-D WITH_CUBLAS=ON \
-D ENABLE_FAST_MATH=ON \
-D CUDA_FAST_MATH=ON \
-D OPENCV_DNN_CUDA=ON \
-D ENABLE_NEON=ON \
-D WITH_QT=ON \
-D WITH_OPENMP=ON \
-D BUILD_TIFF=ON \
-D WITH_FFMPEG=ON \
-D WITH_GSTREAMER=ON \
-D WITH_TBB=ON \
-D BUILD_TBB=ON \
-D BUILD_TESTS=OFF \
-D WITH_EIGEN=ON \
-D WITH_V4L=ON \
-D WITH_LIBV4L=ON \
-D WITH_PROTOBUF=ON \
-D OPENCV_ENABLE_NONFREE=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D INSTALL_PYTHON_EXAMPLES=OFF \
-D OPENCV_GENERATE_PKGCONFIG=ON \
-D BUILD_EXAMPLES=OFF ..
```

3.3 Kalibracija

Kalibracija kamere i robota je ključni korak u integraciji vizijskih sustava u robotske aplikacije. Kalibracija kamere uključuje određivanje intrinzičnih (unutarnjih) parametara kamere (žarišna duljina, optički centar) i ekstrinzičnih (vanjskih) parametara (rotacija i translacija kamere u odnosu na koordinatni sustav svijeta). Kalibracija robota eye-to-hand, s druge strane, određuje transformaciju između koordinatnog sustava kamere i prihvata ili baze robota. Ova kalibracija je neophodna za preciznu manipulaciju objektima, kao što je prepoznavanje objekata i njihovo hvatanje.

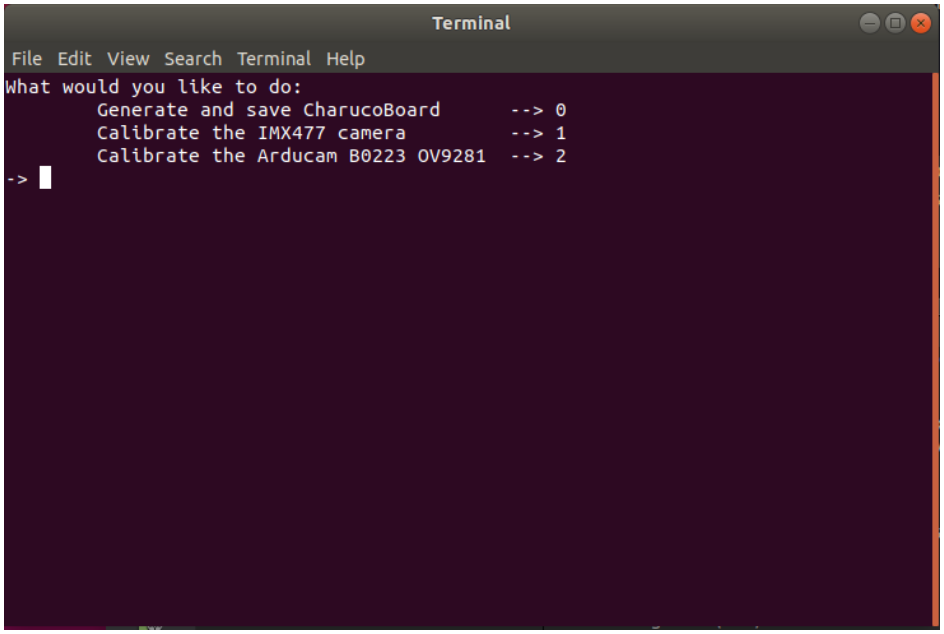
U ovom slučaju kada se pronađu predmeti na stolu primjenom algoritama strojnog vida njihove se koordinate težišta transformiraju u bazu robota. Robot odlazi u izračunate koordinate i hvata predmet.

3.3.1 Kalibracija kamere [24]

Kao što je prije navedeno kalibracijom kamere dobivamo intrinzične i ekstrinzične parametare.

Postupak kalibracije:

Potrebno je izraditi kalibracijski uzorak. Postoje više kalibracijskih uzoraka primjerice šahovska ploča, ploča s kugovima, Aruco ploča, ChAruco ploča itd. Korištena je ChAruco kalibracijska ploča jer ima prednosti nad ostalima. Nije potrebno detektirati cijelu ploču da se algoritam detekcije izvrši. To omogućavaju Aruco markeri, gdje svaki od njih predstavlja jedinstveni ID, koji se dobiva dekodiranjem binarnog koda. U aplikaciji postoji opcija za generiranje i spremanje ChAruco kalibracijske ploče (slika (3.4)) koju isprintamo i postavimo u hvataljku robota.



Slika 3.4: Izgled sučelja aplikacije za kalibraciju

Nakon toga pokrene se aplikacija koja slika slike za kalibraciju. Kalibracijska ploča u robotskoj ruci uperi se u kameru te se kod svakog slikanja pomiče u novi položaj. Robot se postavlja u freedriveMode što omogućuje micanje robota rukom u proizvoljnu poziciju, a tipkom za razmak (engl. *spacebar*) sprema kalibracijsku sliku. Preporuča se uzeti 10 ili više slika s različitih položaja. Postoji mogućnost spremanja koordinata robota što omogućava lakšu kalibraciju druge kamere jer robot "pamti" pozicije u kojima se slike slikaju.

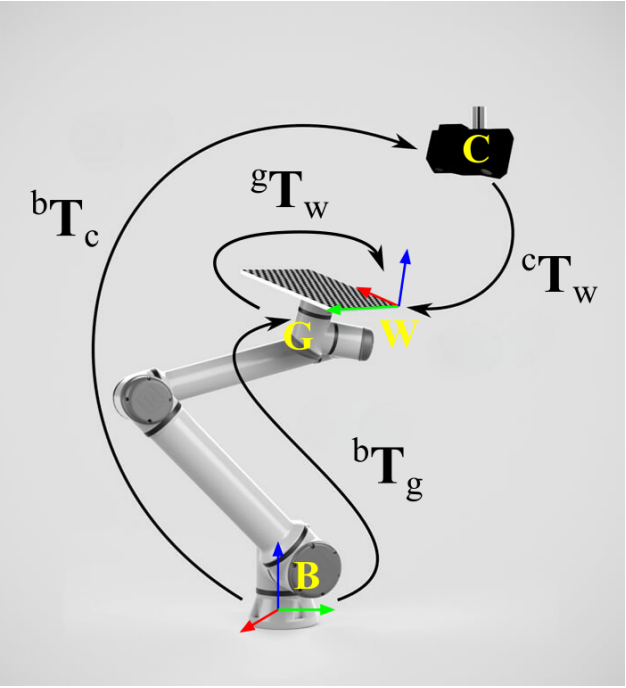
Nakon toga ponovno se otvara aplikacija za kalibraciju i odabire opcija kalibracije željene kamere (IMX477 12MP ili Arducam B0223). Odabirom jedne od njih algoritam učitava kalibracijske slike detektira rubove i ID Aruco markera. Aplikacija sprema matricu kalibracije i distorzije te vektor rotacija i translacija koji predstavljaju transformaciju kalibracijskog uzorka od kamere. Matrica kalibracije i matrica distorzije služe za uklanjanje izobličenja na slikama slikanim tom kamerom. Nakon kalibracije kamere aplikacija izvršava kalibraciju robota oko ruka.

3.3.2 Kalibracija robota oko-ruka

Kalibracija robota oko ruka "eye to hand" podrazumijeva rješavanje jednadžbe

$$\mathbf{AX} = \mathbf{XB} \tag{3.1}$$

Kalibracija oko-ruka uspostavlja točnu vezu između vizijskog sustava (kamere) i kinematičkog sustava (robota). Ova kalibracija je presudna za manipuliranje robotom, poput hvatanja objekata. Kamerom i algoritmom strojnog vida, robot može odrediti točan položaj objekta u prostoru na temelju samo jedne slike. Slika 3.5 prikazuje robota s njegovim pozama, odnosno homogenim transformacijama.



Slika 3.5: "Eye-to-hand" kalibracija [25]

Nadalje u jednadžbama od (3.2) do (3.4) vidi se postupak rješavanja jednadžbe (3.1) s kojom se dobiva homogena transformacija koordinatnog sustava kamere u koordinatnom sustavu baze robota ${}^{b}\mathbf{T}_{c}$.

$$ {}^{b}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{w}^{(1)} = {}^{b}\mathbf{T}_{g}^{(1)}\,{}^{g}\mathbf{T}_{w} \tag{3.2}$$

$$ ({}^{b}\mathbf{T}_{g}^{(1)})^{-1}\;{}^{b}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{w}^{(1)} = {}^{g}\mathbf{T}_{w} = konst. \tag{3.3}$$

$$ ({}^{b}\mathbf{T}_{g}^{(1)})^{-1}\;{}^{b}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{w}^{(1)} = ({}^{b}\mathbf{T}_{g}^{(2)})^{-1}\;{}^{b}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{w}^{(2)} \tag{3.4}$$

$$\vdots$$

$$\mathbf{A}_i\mathbf{X} = \mathbf{X}\mathbf{B}_i \tag{3.5}$$

Sada se vidi da je kalibracija oko-ruka nužna kako bi se mogle koordinate predmeta iz koo-

rdinatnog sustava kamere ${}^{C}\mathbf{p}$ transformirati u koordinatni sustav baze robota ${}^{b}\mathbf{p}$, jednadžba (3.6).

$$ {}^{b}\mathbf{p} = {}^{b}\mathbf{T}_{c}\,{}^{C}\mathbf{p} \tag{3.6} $$

OpenCV biblioteka ima ugrađenu funkciju koja obavlja matematiku u pozadini, *calibrateRobotWorldHandEye()*.

Postupak kalibracije robota oko-ruka:
Slikanjem slika za kalibraciju, aplikacija također sprema vektore pozicija robotske prirubnice u obliku vektora. Taj se vektor može transformirati u matricu rotacije ${}^{g}\mathbf{R}_{b}$ i vektor translacije ${}^{g}\mathbf{t}_{b}$ što su jedni od ulaznih parametri za funkciju kalibracije robota.

estimatePoseCharucoBoard() funkcija estimiranja poze kalibracijske ploče ChAruco iz kalibracijskih slika te vraća vektor rotacije i translacije ${}^{c}\mathbf{t}_{w}$. Matricu rotacije ${}^{c}\mathbf{R}_{w}$ dobivamo pretvorbom vektora rotacije pomoću funkcije *Rodrigues()* koja vraća matricu rotacije ukoliko je ulaz vektor rotacije i obratno.

Te matrice i vektori ulazi su funkcije *calibrateRobotWorldHandEye()* koja vraća matrice ${}^{w}\mathbf{R}_{b}$, ${}^{w}\mathbf{t}_{b}$, ${}^{c}\mathbf{R}_{g}$, ${}^{c}\mathbf{t}_{g}$ koje nam služe za dobivanje koordinata predmeta u koordinatnom sustavu baze robota.

3.4 Lokalizacija predmeta [26]

Lokalizacija predmeta na slikama ključna je u mnogim aplikacijama računalnog vida, kao što su robotika, autonomna vožnja, proširena stvarnost i ostale. Kvaliteta i brzina algoritama za detekciju i opis ključnih točaka [1] izravno utječu na performanse sustava za rad u stvarnom vremenu. U ovom radu korištena su 2 algoritma:

- ORB (Oriented FAST and Rotated BRIEF) i
- SIFT (Scale-Invariant Feature Transform).

Svaki od ovih algoritama ima svoje prednosti i mane, a njihov učinak može značajno varirati ovisno o specifičnim zahtjevima aplikacije. Točnost, brzina obrade, otpornost na promjenu rotacije, osvjetljenja i otpornost na skaliranje ključni su kriteriji kod odabira algoritma. Zbog invarijantnosti na rotaciju i skaliranje odabrani su upravo ORB i SIFT algoritmi za lokalizaciju predmeta u ovom radu.

[1] Ključna točka je karakteristično područje na slici, poput varijacija u intenzitetu.

ORB

Oriented FAST and Rotated BRIEF (ORB) algoritam razvijen je u laboratorijima OpenCV-a, te nudi iznimno brzu obradu i manji broj značajki u usporedbi sa SIFT-om. ORB koristi kombinaciju FAST algoritma za detekciju ključnih točaka i BRIEF algoritma koji uzima ključne točke i pretvara ih u binarni deskriprtor/značajke koja sadrži kombainaciju 0 i 1. ORB se često koristi zbog svoje brzine jer ne zahtjeva GPU i besplatan je za korištenje tj. nije licenciran. Manje je robustan na promjene skale i ne daje zadovoljavajuće rezultate na zamagljenim ili izobličenim slikama.

Postupak:

- učitavanje slike uzorka i slike scene,
- inicijalizacija ORB detektora,
- detektiranje ključnih točaka i računanje deskriptora na slikama uzorka i na slici scene,
- podudaranje ključnih točaka primjenom BFMatchera (Brute Force Matcher),
- filtriranje podudaranja ključnih točaka,
- traženje homografije i mapiranje uzorak na scenu te na kraju
- dobivanje središta lokaliziranog predmeta.

SIFT

Scale-Invariant Feature Transform (SIFT) je algoritam koji je prvi put predstavio David Lowe 1999. godine. SIFT identificira ključne točke unutar slike koje su invariantne na promjene u skali, rotaciji, osvjetljenju i geometrijskim transformacijama.

SIFT detektori pronalaze ključne točake na slici. Proces detekcije počinje izgradnjom višerezolucijske piramide preko ulazne slike, gdje se primjenjuje razlika Gaussiana (DoG) kako bi se identificirale lokalne ekstremne točke unutar prostora skale što su zapravo ključne točke. Zatim se te točke dodatno filtriraju kako bi se uklonile točke koje nisu značajne.

SIFT deskriptori su matematički opisi ključnih točaka, koji ih karakteriziraju na način da su invarijantni na rotaciju i druge promjene u slici. Svaka ključna točka dobiva svoju orijentaciju, koja pomaže u opisivanju te točke bez obzira na rotaciju slike. Na kraju, za svaku ključnu točku se izračunava deskriptor koji se sastoji od 128 elemenata, organiziranih u 16 blokova veličine 4x4. Svaki blok stvara histogram orijentacije, što omogućava precizno opisivanje orijentacije i raspodjele intenziteta oko ključne točke.

Korištenjem SIFT-a, značajke se generiraju s visokom preciznošću, no ovaj algoritam je računalno kompleksan stoga i vremenski zahtjevniji te nije optimalan za aplikacije koje zahtijevaju brzu obradu podataka.

Postupak je sličan ORB algoritmu samo je kod SIFT algoritma korišten algoritam za podudaranje FLANNBASED

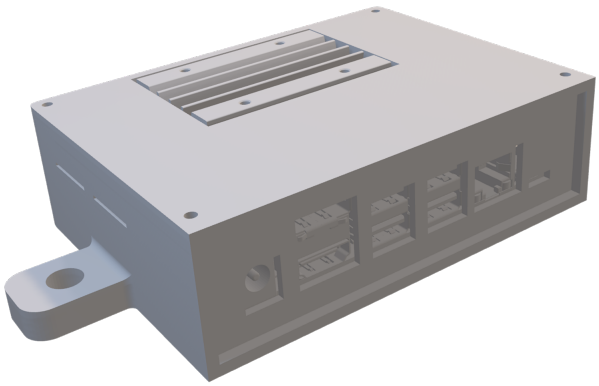
Postupak lokalizacije predmeta: Prvo je potrebno učitati slike uzorka koje se žele detektirati. Zatim se fotografira scena te se ta slika ispravi korištenjem parametara dobivenih iz kalibracije kamere. Na ispravljenim slikama provode se algoritmi za pronalaženje ključnih točaka, koje se potom koriste za izračunavanje deskriptora. Za podudaranje deskriptora između slike scene i slike uzorka koriste se FLANNBASED i BFMatcher algoritmi, koji generiraju niz podudaranja. Ta podudaranja se filtriraju koristeći Loweov omjer kako bi se eliminirali nejasna podudaranja što omogućujue precizniju lokalizaciju predmeta. Nakon filtriranja podudaranja, traži se homografija koja omogućuje mapiranje slike uzorka na sliku scene, iz čega se može izračunati centar detektiranog predmeta.

Pomoću pronađenog centra predmeta, matrica transformacije dobivene iz kalibracije robota i matrice kamere izračunava se točka u koordinatnom sustavu baze robota te se manipulira predmetima.

3.5 Konstrukcija kučišta

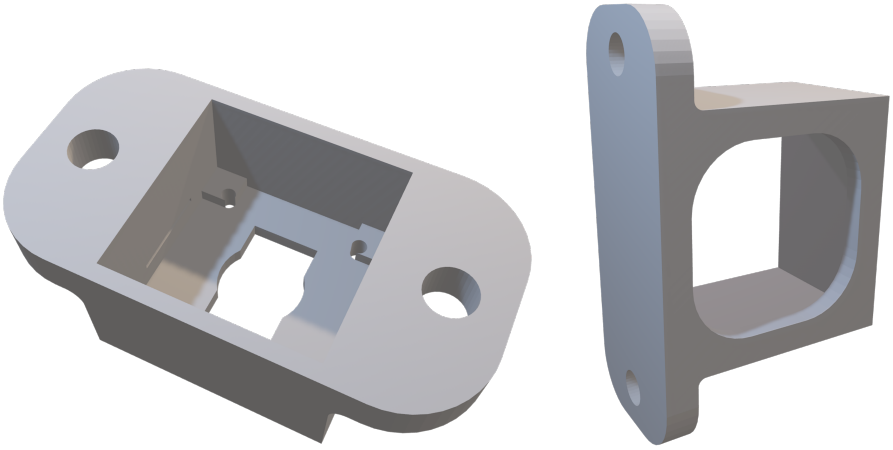
Za potrebe ovog rada modelirano je i 3D printano:

- kučište za Jetson Nano računalo (slika 3.6),
- te kučišta za IMX477 B0250 i B0233 kamere (slika 3.7).



Slika 3.6: Kučište za Jetson Nano računalo

Sva kućišta isprintana su na Prusa MK4S printeru, u jednom printu koji je trajao približno 8 sati. Kao materijal korišten je PETG filament koji je napredna varijanta polietilen-tereftalata (PET), obogaćena glikolom tijekom polimerizacije. Ova dodatna komponenta čini filament izuzetno izdržljivim, manje podložnim pucanju i znatno jednostavnijim za upotrebu u odnosu na standardni PET. PETG je odličan izbor za 3D printanje predmeta koji zahtijevaju visoku čvrstoću, glatku završnu obradu i minimalno skupljanje.



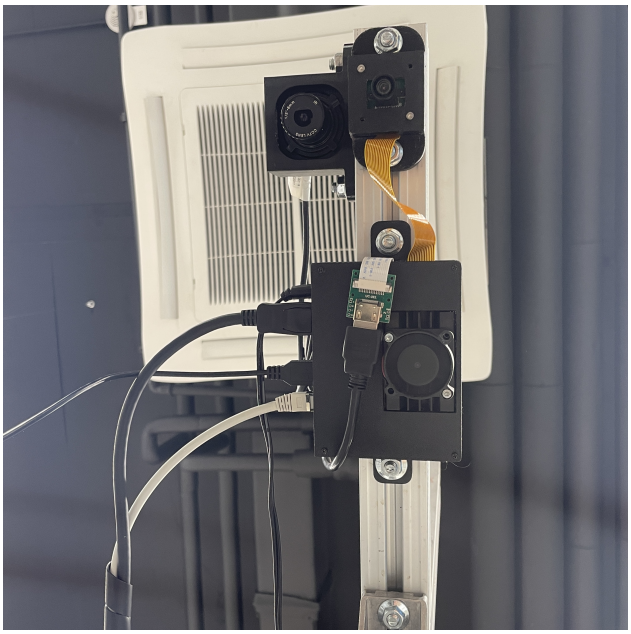
(a) B0233 (b) iMX477 B0250

Slika 3.7: Kučišta za B0233 i B0250 kamere

Kućište za Jetson Nano računalo modelirano je kao dvodijelna struktura, sastavljena od baze i poklopca. Predviđeno je da se računalo i poklopac učvrste pomoću osam samoreznih vijaka za plastiku veličine M2.5x6.

Kamera B0233 pričvršćuje se u svoje kućište koristeći četiri vijka M2.5x12 i matice, dok je B0250 kamera učvršćena trenjem zahvaljujući kućištu oblikovanom u obliku konusa.

Sva su kućišta pričvršćena na standardne aluminijske profile dimenzija 45x45, postavljene u obliku slova L kako bi kamere mogle biti usmjerene prema radnom prostoru. Za montažu kućišta na profile korišteno je šest T-vijaka veličine M8. Montirana kućišta mogu se vidjeti na slici 3.8



Slika 3.8: Montaža kamera i Jetson Nano računala

4 Rezultati i evaluacija rezultata

4.1 Evaluacija rezultata

Nakon što smo izradili cijeli sustav, potrebo je provesti mjerenja točnosti. Inicijalno se uspostavlja greška kalibracije robota u oko-ruka konfiguraciji.

Mjerenja su se provodila za dvije kamere tako što se na sceni označilo 7 točaka razbacanih po cijeloj sceni koju kamera može vidjeti. Fotografirala se scena prvo s jednoma pa onda s drugom kamerom (slika 4.1a i 4.2a) . Izvuku se koordinate točaka u pikselima te provede transformacija iz koordinatnog sustava kamere u koordinatni sustav baze robota te se dobivaju koordinate robota. Te se koordinate uspoređuju sa stvarnim koordinatama koje se dobivaju pomicanjem robota u točku na sceni. Grafički prikaz greške kalibracije robota u konfiguraciji oko-ruka može se vidjeti na slikama 4.1b i 4.2b.

Tablice 4.1 i 4.2 prikazuju odstupanja izračunate točke od stvarne vrijednosti te točke u milimetrima.

Točke	Točka 1	Točka 2	Točka 3	Točka 4	Točka 5	Točka 6	Točka 7
Udaljenosti [mm]	4.85	10.96	0.88	3.37	3.72	7.50	5.45

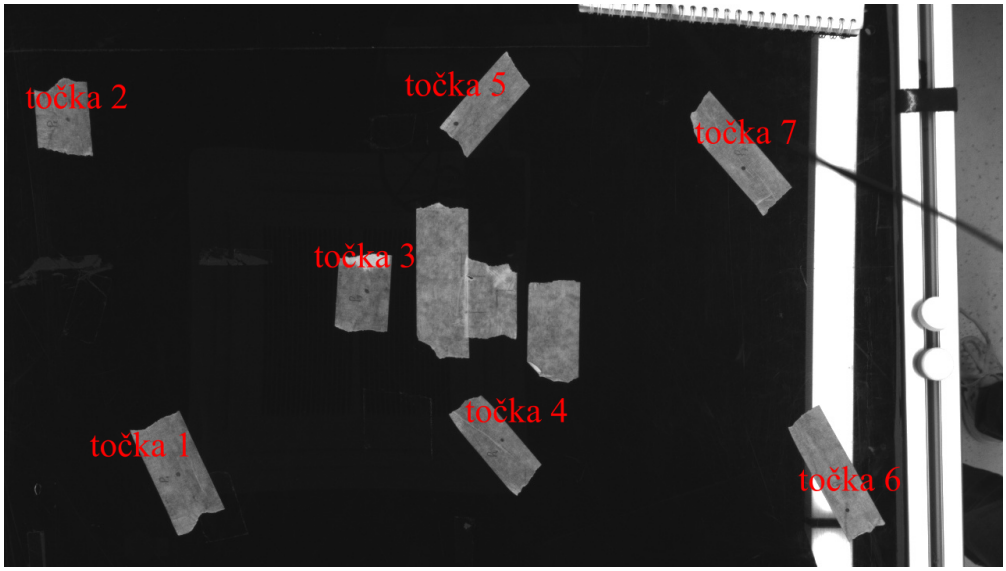
Tablica 4.1: Greška udaljenosti između točaka za B0233 kameru

Točke	Točka 1	Točka 2	Točka 3	Točka 4	Točka 5	Točka 6	Točka 7
Udaljenosti [mm]	4.51	2.00	4.16	3.49	3.63	3.89	3.05

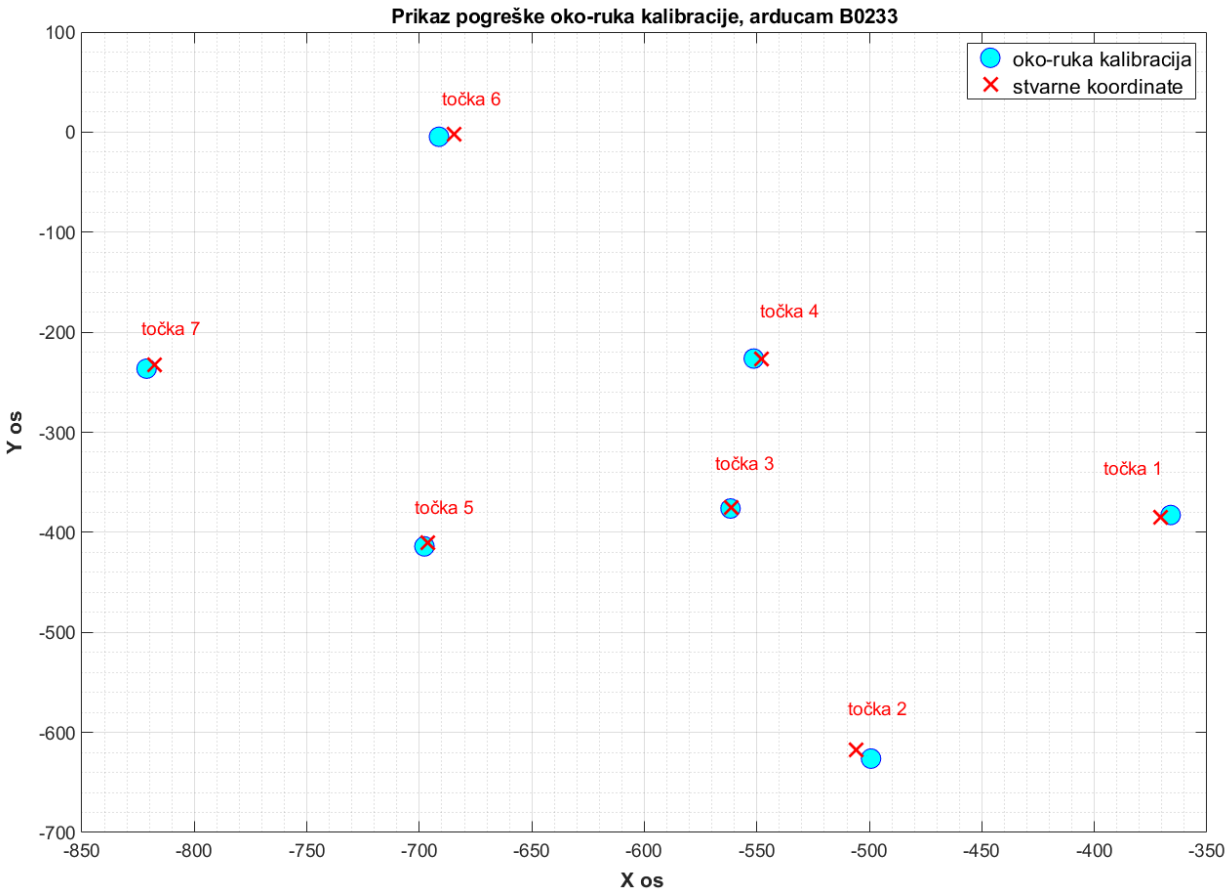
Tablica 4.2: Greška udaljenosti između točaka za IMX477 B0250 kameru

Iz dobivenih rezultata zaključuje se da točke udaljenije od optičkog centra kamere imaju veća odstupanja. Također može se primijetiti da druga kamera (B0250) ima manja odstupanja od B0233 kamere. Do toga je moglo doći zbog bolje kalibracije ili bolje rezolucije B0250 kamere.

Budući da B0233 kamera ima manju rezuluciju, pogreška kod kalibracije kamere ima veći utjecaj na preciznost jer jedan piksel predstavlja više milimetara na sceni nego kod kamere koja ima veću rezoluciju.

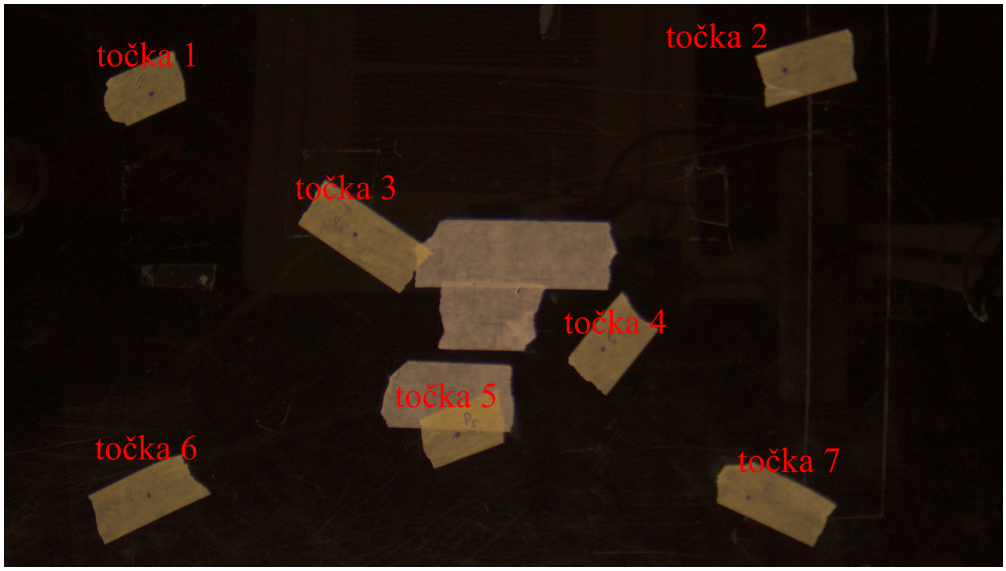


(a) Scena za B0233 kameru

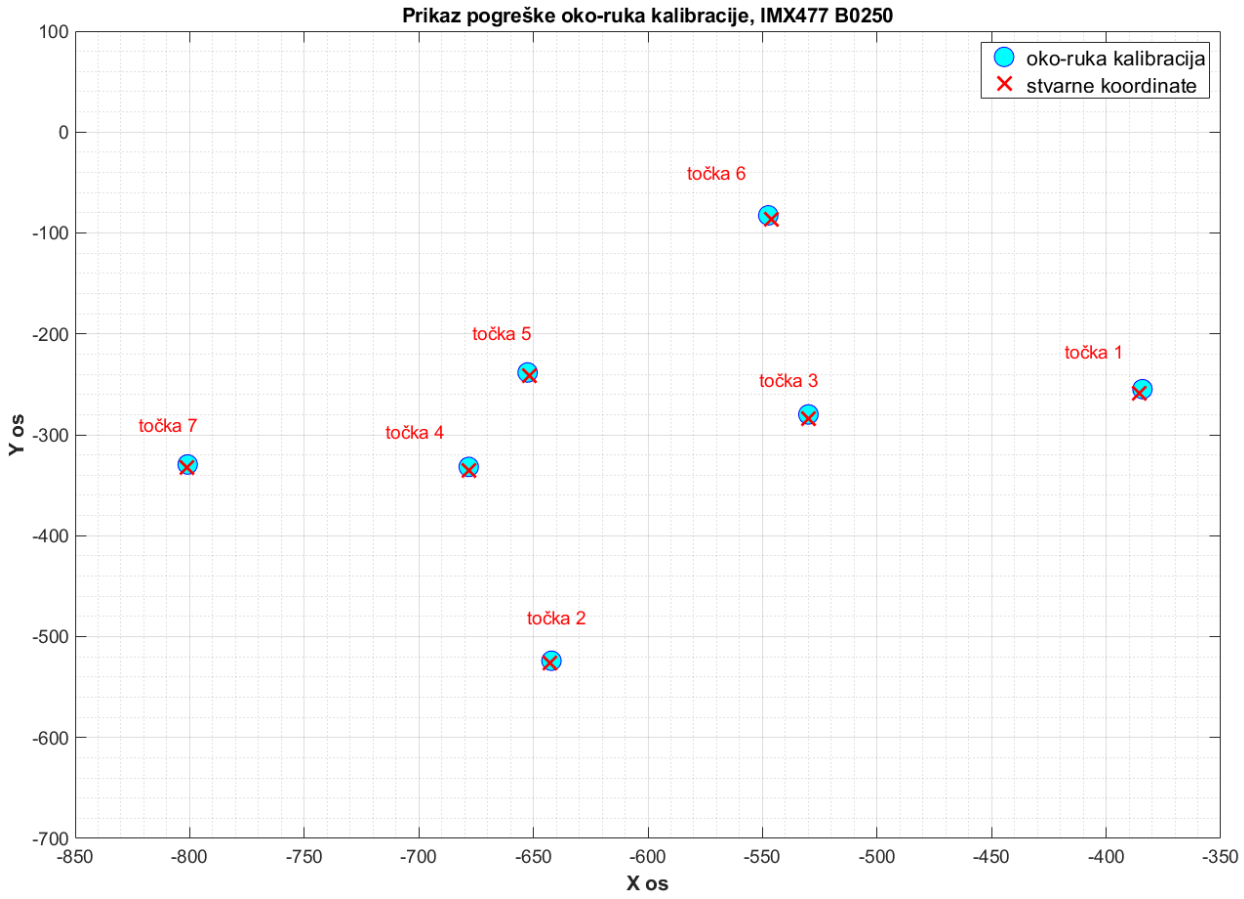


(b) Grafički prikaz greške

Slika 4.1: Greška oko-ruka kalibracija za B0233 kameru



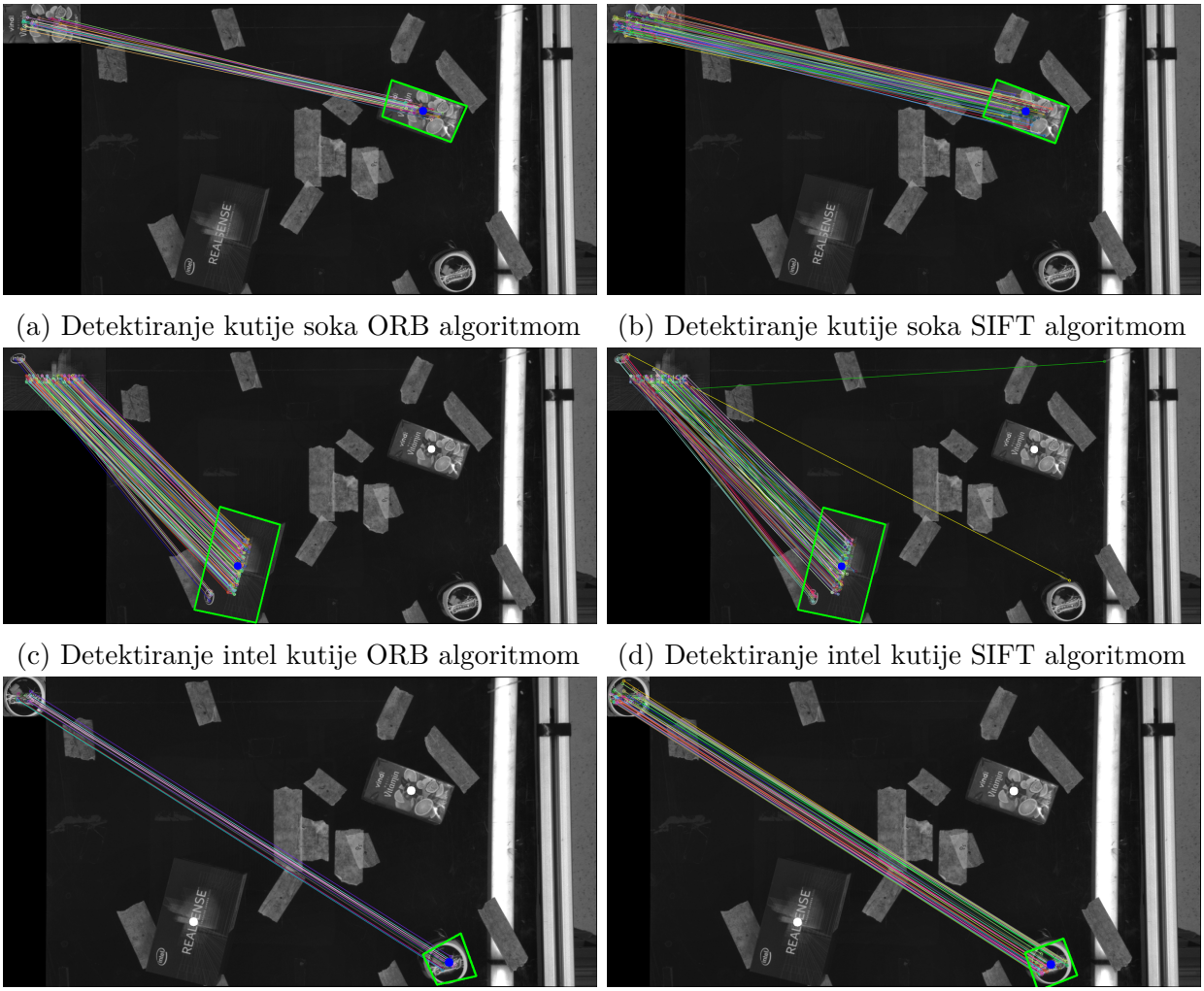
(a) Scena za IMX477 B0250 kameru



(b) Grafički prikaz greške

Slika 4.2: Greška oko-ruka kalibracija za IMX477 B0250 kameru

Nadalje se analizira pogreška detektiranja predmeta na sceni. Nasumično se postave predmeti na scenu 2 puta. Svaki put detektiraju se predmeti jednim od algoritama detekcije (ORB, SIFT) te dobivaju njihova središta koja se opet pretvaraju u koordinatni sustav baze robota. Stvarana pozicija dobiva se pomicanjem robota u središte predmeta na sceni. Uspoređivanjem tih podataka dobivamo ukupnu grešku cijelog sustava.



(a) Detektiranje kutije soka ORB algoritmom (b) Detektiranje kutije soka SIFT algoritmom

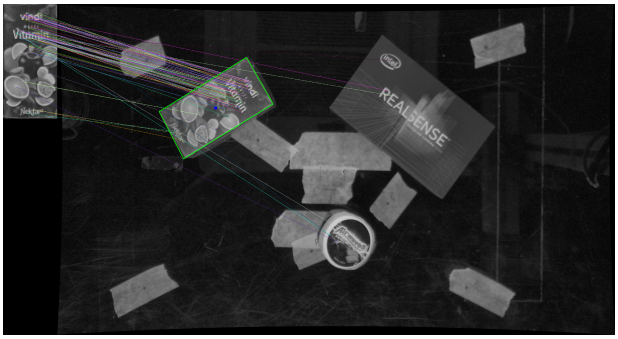
(c) Detektiranje intel kutije ORB algoritmom (d) Detektiranje intel kutije SIFT algoritmom

(e) Detektiranje kutije žvakaća ORB algoritmom (f) Detektiranje kutije žvakaća SIFT algoritmom

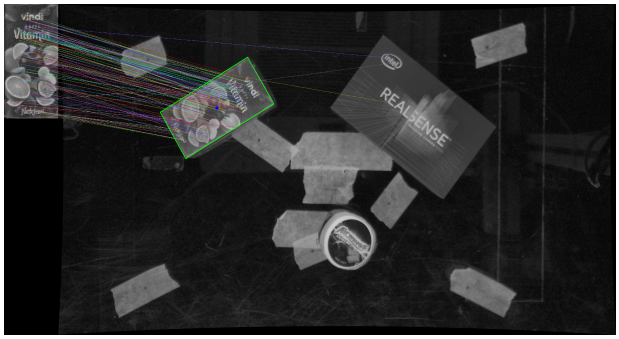
Slika 4.3: Usporedba detekcijskih algoritama na slici snimljenoj B0233 kamerom

Na slikama 4.3 i 4.4 može se primijetiti da je SIFT algoritam precizniji od ORB algoritma te pronalazi puno više ključnih točaka. Uspoređivanjem slike 4.3e i 4.3f vidi se da SIFT bolje prepoznaje manje predmete.

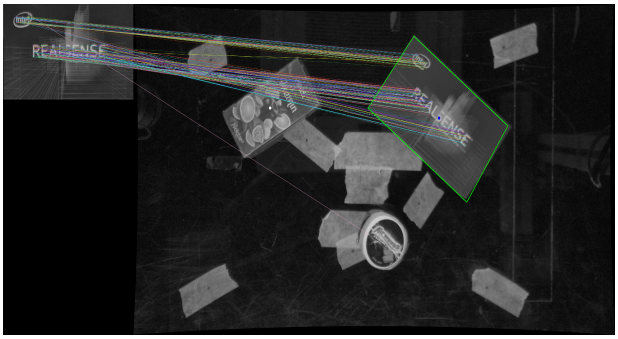
Zbog kompleksnosti SIFT algoritma, vrijeme detekcije značajno raste s povećanjem rezolucije slike. Iako na slikama rezolucije 1280x720 SIFT ne pokazuje uočljivo sporije performanse u usporedbi s ORB algoritmom, detekcija postaje znatno sporija pri obradi slika rezolucije 3840x2160. Stoga je za aplikacije u kojima je brzina detekcije ključna, ORB algoritam bolji izbor.



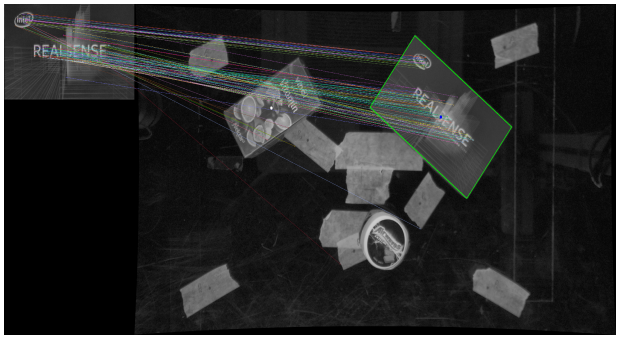
(a) Detektiranje kutije soka ORB algoritmom



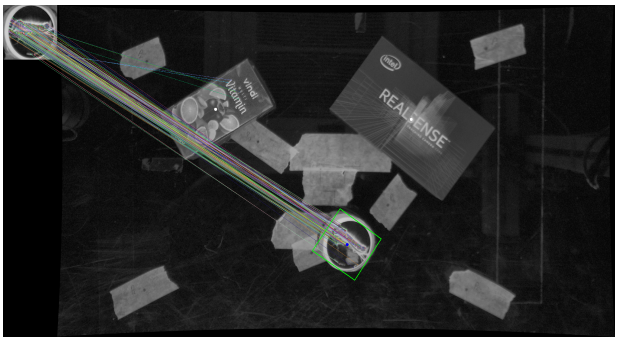
(b) Detektiranje kutije soka SIFT algoritmom



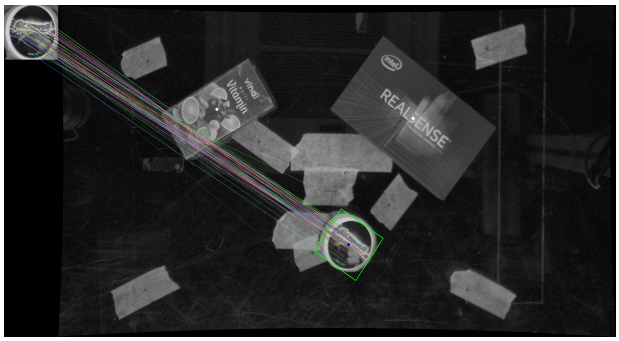
(c) Detektiranje intel kutije ORB algoritmom



(d) Detektiranje intel kutije SIFT algoritmom



(e) Detektiranje kutije žvakaća ORB algoritmom



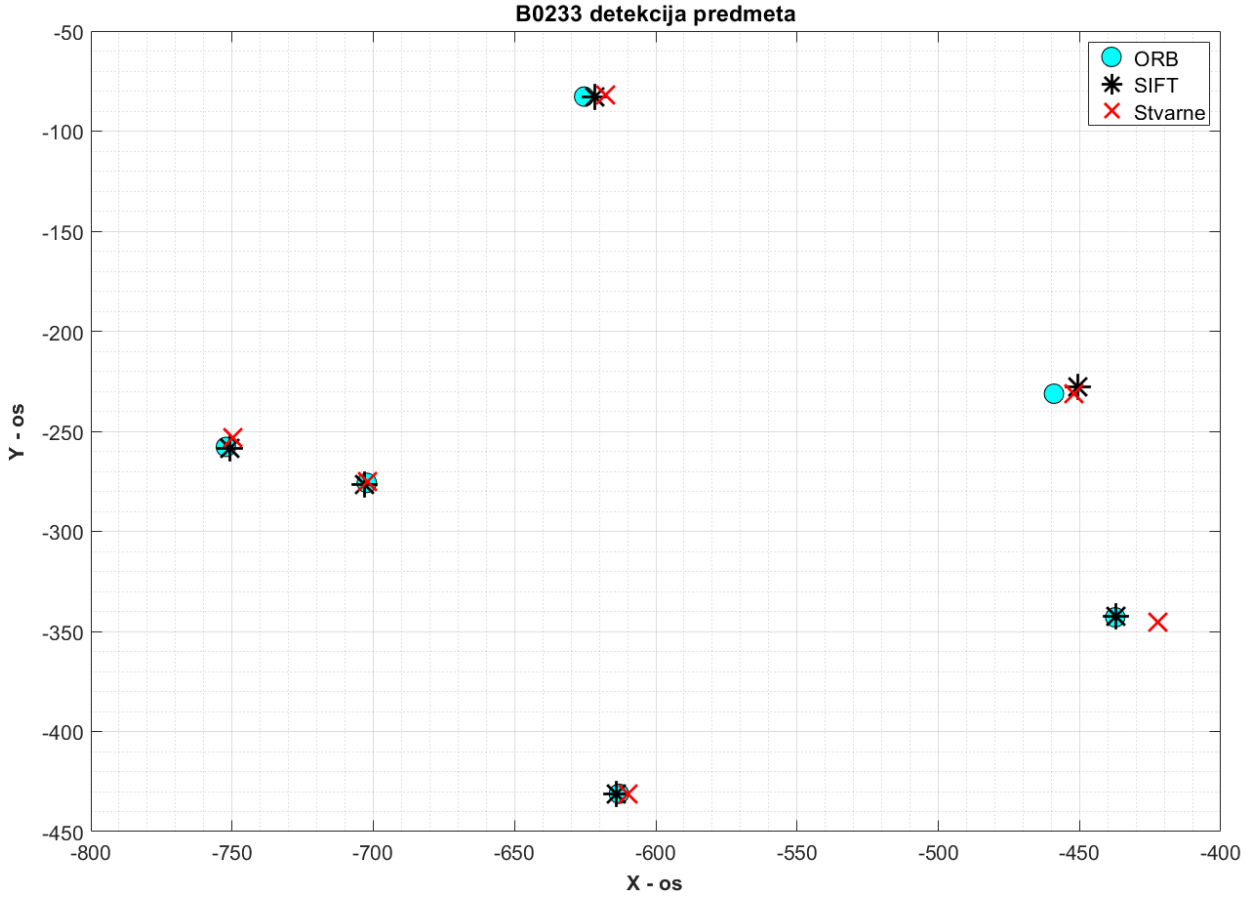
(f) Detektiranje kutije žvakaća SIFT algoritmom

Slika 4.4: Usporedba detekcijskih algoritama na slici snimljenoj IMX477 B0250 kamerom

Grafički prikaz pogrešaka sustava za obje kamere prikazan je na slikama 4.5 i 4.6 i u tablicama 4.3 i 4.4 . Uočava se da se pogreške povećavaju kako se približavamo rubovima slike, što je bilo očekivano zbog ranije pokazane greške kalibracije robota u konfiguraciji oko-ruka. Zaključuje se da ove pogreške nisu značajne za manipulaciju predmetima korištenim u ovom radu, no kod manipulacije manjim predmetima sustav bi mogao imati poteškoća.

Tablica 4.3: Greška sustava za B0233 kameru

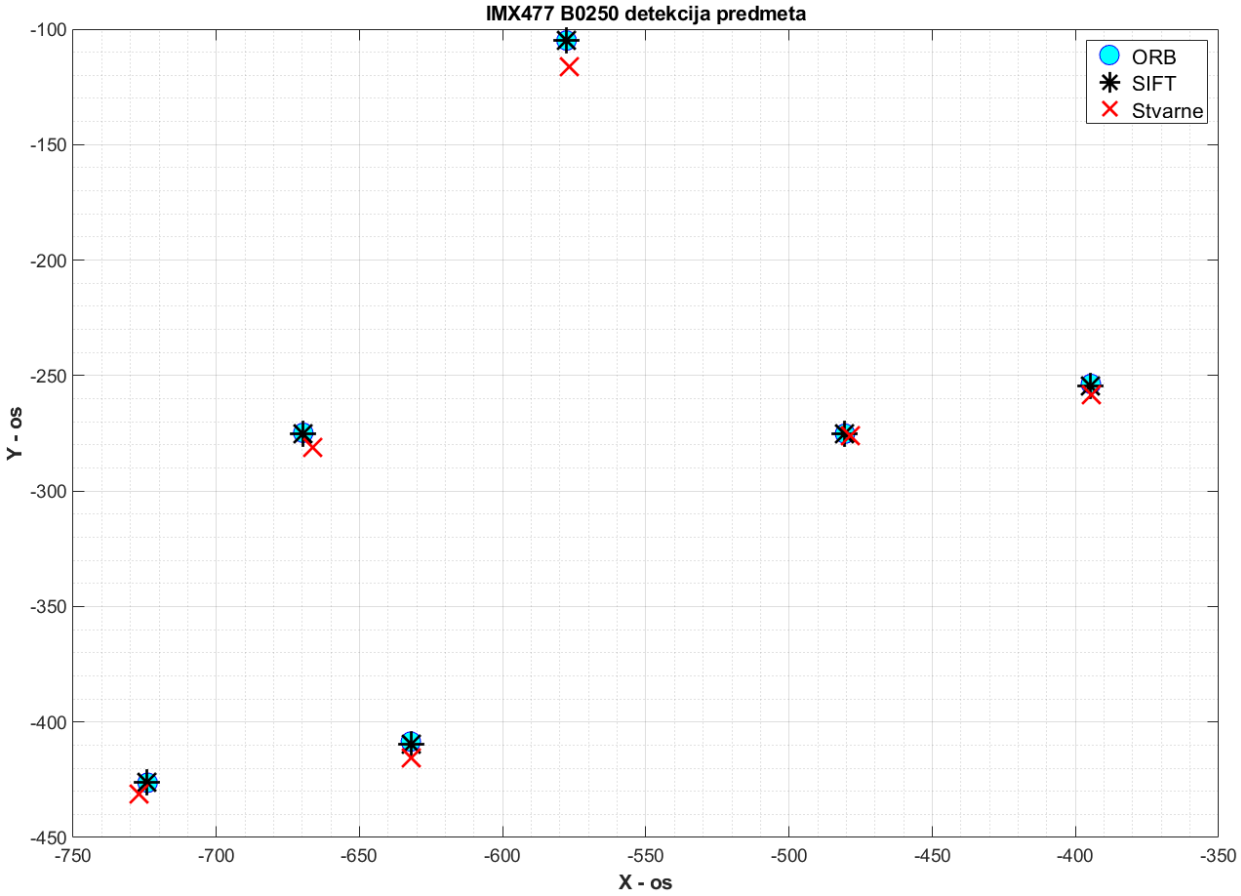
Algoritam	Objekt	Stvarna pozicija x [mm]	Stvarna pozicija y [mm]	Izračunata pozicija x [mm]	Izračunata pozicija y [mm]	Apsolutna greška [mm]
1. Slika						
ORB	sok	-749.9	-253.4	-752.202	-257.793	4.96
SIFT	sok	-749.9	-253.4	-750.649	-258.61	5.26
ORB	intel kutija	-422.3	-345.2	-437.364	-342.939	15.23
SIFT	intel kutija	-422.3	-345.2	-436.965	-342.557	14.90
ORB	žvakaće	-617.8	-82.1	-625.362	-82.750	7.59
SIFT	žvakaće	-617.8	-82.1	-621.672	-82.826	3.94
2. Slika						
ORB	sok	-452.1	-231.2	-459.049	-231.216	6.95
SIFT	sok	-452.1	-231.2	-450.728	-227.851	3.62
ORB	intel kutija	-609.9	-431.1	-613.054	-431.111	3.15
SIFT	intel kutija	-609.9	-431.1	-614.054	-431.111	4.15
ORB	žvakaće	-702.1	-275.1	-702.312	-275.753	0.69
SIFT	žvakaće	-702.1	-275.1	-703.066	-276.476	1.68



Slika 4.5: Grafički prikaz greške sustava za B0233 kameru

Tablica 4.4: Greška sustava za IMX477 B0250 kameru

Algoritam	Objekt	Stvarna pozicija x [mm]	Stvarna pozicija y [mm]	Izračunata pozicija x [mm]	Izračunata pozicija y [mm]	Apsolutna greška [mm]
1. Slika						
ORB	sok	-478.5	-276.0	-480.483	-275.155	2.16
SIFT	sok	-478.5	-276.0	-480.615	-275.290	2.23
ORB	intel kutija	-632.1	-415.5	-632.096	-408.485	7.02
SIFT	intel kutija	-632.1	-415.5	-632.023	-409.781	5.72
ORB	žvakaće	-666.3	-281.1	-669.651	-274.798	7.14
SIFT	žvakaće	-666.3	-281.1	-669.875	-275.072	7.00
2. Slika						
ORB	sok	-394.6	-258.5	-394.629	-253.638	4.86
SIFT	sok	-394.6	-258.5	-394.962	-254.651	3.87
ORB	intel kutija	-727.1	-431.4	-724.007	-426.281	5.98
SIFT	intel kutija	-727.1	-431.4	-724.160	-426.175	6.00
ORB	žvakaće	-576.9	-116.2	-577.721	-104.919	11.31
SIFT	žvakaće	-576.9	-116.2	-577.661	-105.057	11.17



Slika 4.6: Grafički prikaz greške sustava za IMX477 B0250 kameru

5 Zaključak

Ovaj rad predstavlja uspješno razvijen sustav koji integrira strojni vid i robotiku, omogućavajući preciznu lokalizaciju i manipulaciju objekata u stvarnom okruženju. Korištenjem Jetson Nano računala i kamera različitih rezolucija, postignuta je zadovoljavajuća brzina detekcije i razina preciznosti u kalibraciji robota u konfiguraciji oko-ruka, što je ključno za uspješnu manipulaciju predmetima. Greške se mogu smanjiti korištenjem preciznijih kalibracijskih uzoraka i snimanjem kalibracijskih fotografija bliže radnom stolu. Za industrijske primjene, preporučuje se postavljanje kamere tako da fotografira predmet u optičkom središtu kamere, budući da su greške u tom području najmanje.

Rezultati rada potvrđuju da je razvijeni sustav sposoban za zadovoljavajuću lokalizaciju i manipulaciju objektima, čime se otvaraju mogućnosti za unaprijeđene sustava za industrijske primjene. Ovaj sustav predstavlja solidnu osnovu za daljnji razvoj i može se primijeniti u širokom spektru aplikacija.

Daljnji razvoj mogao bi se unaprijediti manipulacijom predmeta koji su bliže robotu i provođenjem algoritma detekcije drugog predmet tijekom manipulacije prvog, što bi moglo smanjiti ukupno vrijeme trajanja procesa. Osim toga, za precizniju detekciju predmeta mogla bi se koristiti stereokamera koja omogućuje snimanja trodimenzionalnih slika gdje se dobiva informaciju o dubini, čime bi se olakšalo manipuliranje različitih predmeta bez potrebe za poznavanjem njihovih dimenzija. Na taj način povećala bi se robusnost i fleksibilnost cijelog sustava. Korištenje naprednijih algoritama za detekciju predmeta koji ne zahtijevaju uzorke za prepoznavanje objekta na slici, dodatno bi unaprijedilo robusnost i fleksibilnost sustava.

Literatura

[1] Michael Barr; Anthony Massa. *Programming Embedded Systems: With C and GNU Development Tools Second Edition*. O'Reilly Media, 2006.

[2] *Jetson Nano*. URL: https://developer.nvidia.com/embedded/jetson-nano. pristupljeno 12. kolovoza 2024.

[3] Hrvatska enciklopedija mrežno izdanje. *Robot*. Leksikografski zavod Miroslav Krleža. 2013–2024. URL: https://www.enciklopedija.hr/clanak/robot. Pristupljeno 12. kolovoza 2024.

[4] *The First Industrial Robot*. URL: https://www.automate.org/robotics/engelberger/joseph-engelberger-unimate. Pristupljeno 12. kolovoza 2024.

[5] Sophie Hand. *A Brief History of Collaborative Robots*. 2020. URL: https://www.mhlnews.com/technology-automation/article/21124077/a-brief-history-of-collaborative-robots. Pristupljeno 12. kolovoza 2024.

[6] *Robotika - Kobot vs Robot*. URL: https://mreza.bug.hr/robotika/robotika-kobot-vs-robot-32282. Pristupljeno 12. kolovoza 2024.

[7] *UR5 collaborative robot*. URL: https://www.universal-robots.com/products/ur5-robot/. Pristupljeno 12. kolovoza 2024.

[8] *Real-Time Data Exchange (RTDE) Guide*. URL: https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/. Pristupljeno 12. kolovoza 2024.

[9] T. Huang. *Computer Vision: Evolution And Promise* -. Teh. izv. CERN, 2009. URL: https://cds.cern.ch/record/400313/files/p21.pdf. Pristupljeno 13. kolovoza 2024.

[10] Alex Owen-Hill. *Robot Vision vs Computer Vision: What's the Difference?* 2016. URL: https://blog.robotiq.com/robot-vision-vs-computer-vision-whats-the-difference. Pristupljeno 13. kolovoza 2024.

[11] *Hand-Eye Calibration Problem*. URL: https://support.zivid.com/en/latest/academy/applications/hand-eye/hand-eye-calibration-problem.html#. Pristupljeno 13. kolovoza 2024.

[12] Peter Corke. *Robotics, Vision and Control*. Sv. 73. Springer Berlin Heidelberg, 2011. ISBN: 978-3-642-20143-1. DOI: 10.1007/978-3-642-20144-8.

[13] Migk. *Dagerotipija*. 2021. URL: https://blog.migk.hr/2021/10/19/dagerotipija/. Pristupljeno 14. kolovoza 2024.

[14] *ZED X Stereo Camera*. URL: https://www.stereolabs.com/en-hr/store/products/zed-x-stereo-camera. Pristupljeno 13. kolovoza 2024.

[15] *2D Machine Vision midiCam2*. URL: https://www.axiomtek.com/Default.aspx?MenuId=Products&FunctionId=ProductView&ItemId=26186&C=2D%20Machine%20Vision%20midiCam2&upcat=378.

[16] Filip Šuligoj. *Prostorne transformacije i konfiguracije vizijskih sustava u robotici*.

[17] *Arducam OV9281 1MP Global Shutter*. URL: https://www.arducam.com/product/arducam-ov9281-mipi-1mp-monochrome-global-shutter-ir-camera-module-for-jetson-nano/. Pristupljeno 16. kolovoza 2024.

[18] *IMX477 - ArduCam Complete High Quality Camera Bundle*. URL: https://dlscorp.com/shop/imx477-arducam-complete-high-quality-camera-bundle/. Pristupljeno 17. kolovoza 2024.

[19] Bjarne Stroustrup. *The C++ programming language*. Addison Wesley Longman, 1997. ISBN: 9812359354.

[20] David. Millán Escrivá, Prateek. Joshi i Vinícius. G. Mendonça. *Building Computer Vision Projects with OpenCV 4 and C++*. Packt Publishing, 2019.

[21] *Jetson Nano 2GB Developer Kit - Get Started*. URL: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit#prepare. Pristupano 18. kolovoza 2024.

[22] *ur rtde 1.5.9 documentation*. URL: https://sdurobotics.gitlab.io/ur_rtde/installation/installation.html. Pristupano 18. kolovoza 2024.

[23] *Install OpenCV on Jetson Nano - Q-engineering*. URL: https://qengineering.eu/install-opencv-on-jetson-nano.html. Pristupano 20. kolovoza 2024.

[24] Frank Dellaert i Seth Hutchinson. *Robotics book*. URL: https://www.roboticsbook.org/S53_diffdrive_sensing.html. Pristupljeno 13. kolovoza 2024.

[25] *The benefits of 3D hand-eye calibration*. URL: https://blog.zivid.com/importance-of-3d-hand-eye-calibration. Pristupano 21. kolovoza 2024.

[26] Monika Bansal, Munish Kumar i Manish Kumar. „2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors". *Multimedia Tools and Applications* 80 (12 svibanj 2021.), str. 18839–18857. ISSN: 1380-7501. DOI: 10.1007/s11042-021-10646-0. Pristupano 7. rujna 2024.

Prilozi

Izvorni kod: https://github.com/l-zavrski/zavrsni-rad/tree/main