Primjena neuronskih mreža u ekstrinzičnoj kalibraciji robotskog in-hand stereo-vizijskog sustava

Samaržija, Lea

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Lea Samaržija

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Doc. dr. sc. Filip Šuligoj, mag. ing. mech

Student:

Lea Samaržija

Zagreb, 2024.

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

Zahvaljujem se doc. dr. sc. Filipu Šuligoju na zadavanju teme završnog rada, savjetima i pomoći tijekom izrade ovog rada.

Zahvaljujem se obitelji i prijateljima na podršci i strpljenju i motivaciji tijekom studiranja.

Lea Samaržija

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: **Lea Samaržija** JMBAG: **0035225648**

Naslov rada na hrvatskom jeziku: **Primjena neuronskih mreža u ekstrinzičnoj kalibraciji robotskog in-hand stereo-vizijskog sustava**

Naslov rada na engleskom jeziku: **Application of Neural Networks in Extrinsic Calibration of Robotic In-Hand Stereo Vision System**

Opis zadatka:

U cilju razvijanja robusnog i preciznog mehanizma lokalizacije za robotske sisteme s in-hand stereo-vizijskim sustavom, ovaj rad istražuje upotrebu neuronskih mreža u procesu ekstrinzične kalibracije. Stereo-vizijski sustav integriran u robotsku ruku omogućava precizno pozicioniranje u odnosu na objekte, što je od suštinske važnosti za primjene u industriji, medicini i znanstvenim istraživanjima. S obzirom na ograničenja klasičnih metoda kalibracije ekstrinzičnih parametara, ovaj rad predlaže implementaciju neparametarske kalibracije koristeći neuronske mreže. Takav pristup pruža povećanu robusnost u odnosu na šum i nepravilnosti, veću fleksibilnost, prilagodljivost različitim vrstama kamera, te potencijal za automatizaciju procesa kalibracije.
Metodologija:

- modifikacija postojećeg programa laser-kamera modela za simulaciju stereo-vizijskog sustava na robotskoj ruci i pohranu 3D pozicija i koordinata centara sfere
- razvoj strukture neuronske mreže za mapiranje koordinata centara sfera na 3D robotske pozicije
- određivanje optimalne raspodjele kalibracijskih prostornih točaka za efikasno treniranje neuronske mreže
- implementacija i evaluacija modela neuronske mreže koristeći simulirane podatke koji sadrže grešku mjerenja
- analiza točnosti i robusnosti različitih vrsta neuronskih mreža u procesu kalibracije
- provjera točnosti ekstrinzične kalibracije s neuronskom mrežom kroz računanje euklidske greške na pozicijama izvan trening seta.

Rad treba sadržavati pregled literature, detaljan opis korištenih metoda i algoritama te evaluaciju rezultata u kontekstu primjenjivosti u stvarnim aplikacijama.

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

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

Zadatak zadao:

Doc. dr. sc. Filip Šuligoj

Predsjednik Povjerenstva:

Prof. dr. sc. Damir Godec

SADRŽAJ

SADRŽAJ I
POPIS SLIKA III
POPIS TABLICA V
POPIS OZNAKA VI
SAŽETAK VIII
SUMMARY IX
1. UVOD 1
2. MODEL KAMERE 3
2.1. *Pinhole* model kamere 3
2.2. Projekcija na senzor kamere 4
2.2.1. Matrica rotacija 7
3. MODEL STEREOKAMERE 9
3.1. Matematička podloga stereokamera sustava 9
4. NEURONSKA MREŽA 12
4.1. Višeslojni perceptron (MLP) 14
4.1.1. Levenberg-Marquardt (LM) algoritam 15
4.2. Konvolucijska neuronska mreža (CNN) 16
4.2.1. 1D konvolucijske neuronske mreže 17
5. Programska izvedba 18
5.1. Grafičko sučelje 18
5.2. Model stereo kamere 20
5.2.1. Model pinhole kamere 20
5.3. Istaknuti dijelovi koda potrebni za generiranje testnog seta podataka 21
5.3.1. Funkcije homogenosti 21
5.3.2. Matrice rotacije 22
5.3.3. Matrice projekcije 24
5.3.4. Matrica projekcije 24
5.3.5. Generiranje testnih podataka u 3D prostoru 25
5.3.6. Projekcija 3D točke na 2D ravninu slike 26
5.3.7. Uključivanje distorzije i lokalizacijske greške u 2D točku na ravnini slike 27
5.3.8. Generiranje 2D Projiciranih Točaka 28
5.4. Model neuronskih mreža 29
5.4.1. Višeslojni perceptron 30
5.4.2. Višeslojni perceptron treniran Levenberg-Marquardt algoritmom 31
5.4.3. Konvolucijska neuronska mreža 32

6. REZULTATI 34
6.1. Višeslojni perceptron 35
6.2. Višeslojni perceptron treniran Levenberg-Marquardt algoritmom 36
6.3. Konvolucijska neuronska mreža 36
6.4. Usporedba rezultata neuronskih mreža 37
7. ZAKLJUČAK 41
LITERATURA 42

POPIS SLIKA

Slika 1. Projiciranje objekta pinhole modelom [1] ... 3
Slika 2. Projekcija točke na senzor kamere [3] ... 6
Slika 3. Rotacije oko koordinatnih osi [4] ... 7
Slika 4. Prikaz epipolarnog ograničenja [9] ... 9
Slika 5. Disparitet [7] ... 10
Slika 6. Triangulacija paralelne konfiguracije kamera [8] ... 11
Slika 7. Umjetni neuron [12] ... 12
Slika 8. Step funkcija ... 13
Slika 9. Višeslojni perceptron s jednim skrivenim slojem [13] ... 14
Slika 10. Slojevi konvolucijske neuronske mreže [14] ... 17
Slika 11. Grafičko sučelje - glavni prozor ... 20
Slika 12. Funkcije se koristene za pretvorbu koordinata između homogenog i nehomogenog oblika ... 22
Slika 13. Funkcije matrice rotacije oko jedne osi ... 23
Slika 14. Funkcija matrice rotacije ... 24
Slika 15. Funkcija za kreiranje kalibracijske matrice ... 24
Slika 16. Funkcija izračuna projekcijske matrice kamere ... 25
Slika 17. Programski kod za generiranje testnih podataka u 3D prostoru ... 26
Slika 18. Funkcija za Izračun Projekcije 3D Točke na 2D Ravninu Slike ... 27
Slika 19. Funkcija za izračun točke na 2D ravnini slike s distorzijom i lokalizacijskom greškom ... 28
Slika 20. Generiranje 2D projiciranih točaka ... 29
Slika 21. Obrada podataka ... 30
Slika 22. Model višeslojnog perceptrona ... 31
Slika 23. MLP treniran LM algoritmom ... 32
Slika 24. Model konvolucijske neuronske mreže ... 33
Slika 25. Paralelna konfiguracija kamera ... 34
Slika 26. Okomita konfiguracija kamera ... 35
Slika 27. Konfiguracija s osima kamera pod 45° ... 35
Slika 28. Usporedba NN pri okomitoj konfiguraciji kamera ... 38
Slika 29. Usporedba NN pri konfiguraciji kamera pod 44° ... 39

Slika 30. Usporedba NN pri paralelnoj konfiguraciji kamera 40

POPIS TABLICA

Tablica 1. Srednja greška testnih podataka MLP-a 36
Tablica 2. Srednja greška testnih podataka MLP treniran LM 36
Tablica 3. Srednja greška testnih podataka CNN 37

POPIS OZNAKA

Oznaka	Jedinica	Opis
$\boldsymbol{\Theta_x,\Theta_y,\Theta_z}$	°	Eulerovi kutovi, kutovi valjanja, posrtanja i skretanja
$\boldsymbol{\rho_w,\rho_h}$	mm	Širina i visina piksela
λ	/	Faktor prigušenja
b	/	Baza, razmak između optičkih osi kamera
$\mathbf{C}$	/	Ishodište koordinatnog sustava kamere
d	/	Disparitet
$\mathbf{e}$	/	Vektor pogreške
f	mm	Žarišna duljina kamere
$\mathbf{H}$	/	Hessian matrica
$\mathbf{I}$	/	Jedinična matrica
$\mathbf{J}$	/	Jacobian matrica
$\mathbf{K}$	/	Kalibracijska matrica kamere
k	/	Duljina filtera za postupak konvolucije
$k1,k2,k3$	/	Koeficijenti radijalne distorzije
net	/	Suma vrijednosti u jednom neuronu
$\mathbf{p}$	/	Točka na 2D slici
$\mathbf{P}$	/	Točka u prostoru
$\tilde{\mathbf{p}}$	/	Homogena točka slike
$\tilde{\mathbf{P}}$	/	Homogena točka u prostoru
px,py	/	Koordinate glavne točke u x, y ravnini
$\mathbf{P}_{\text{cam}}$	/	Točka u prostornom koordinatnom sustavu kamere
$\mathbf{R}$	/	Matrica rotacije
$\mathbf{R}_x$, $\mathbf{R}_y$, $\mathbf{R}_z$	/	Matrice valjanja, posrtanja i skretanja
u, v	/	Koordinate točke u pikselima
u_0, v_0	/	Koordinate glavne točke u pikselima
$\tilde{u}$, $\tilde{v}$, $\tilde{w}$	/	Homogene koordinate u pikselima
w_i	/	Težina i-tog neurona
x, y	/	Koordinate točke u ravnini
$\tilde{x}, \tilde{y}, \tilde{z}$	/	Homogene koordinate točke slike

X, Y, Z	/	Koordinate točke u prostoru
x_i	/	I-ta ulazna vrijednost u neuron
x_{i+j-1}	/	Element ulazne vrijednosti
x_L, x_R	/	Koordinata točke slike lijeve, desne kamere
z_j	/	Izlazna vrijednost za određeni pomak filtera j

SAŽETAK

U ovom završnom radu obrađuje se primjena neuronskih mreža u procesu ekstrinzične kalibracije stereo-vizijskih sustava unutar robotske ruke. Sustav stereo-vizijskih kamera omogućava preciznu lokalizaciju objekata u prostoru, a kalibracija igra ključnu ulogu u osiguravanju točnosti tih mjerenja. Tradicionalne metode kalibracije često su kompleksne i dugotrajne, stoga se u ovom radu koriste neuronske mreže za automatizaciju i pojednostavljenje procesa kalibracije. U ovom radu setovi podataka napravljeni su simulacijom modela stereo kamere. Stoga ovaj rad obuhvaća pregled teorije kamera i stereo kamera, te prikazuje model stereo kamere korišten za simulaciju potrebnih setova podataka. Nakon toga, rad se fokusira na razvoj i testiranje više modela neuronskih mreža, uključujući višeslojni perceptron (MLP) i konvolucijske neuronske mreže (CNN), s ciljem poboljšanja preciznosti pozicioniranja objekata u stvarnim aplikacijama poput autonomnih vozila i robotske manipulacije.

Ključne riječi: stereo-kamera, neuronska mreža, ekstrinzična kalibracija, MLP, CNN

SUMMARY

This thesis addresses the application of neural networks in the process of extrinsic calibration of stereo vision systems within a robotic arm. The stereo vision camera system enables precise localization of objects in space, with calibration playing a crucial role in ensuring the accuracy of these measurements. Traditional calibration methods are often complex and time-consuming, therefore, this paper employs neural networks to automate and simplify the calibration process. The data sets used in this research were generated by simulating a stereo camera model. Consequently, the thesis includes an overview of camera and stereo camera theory, as well as a presentation of the stereo camera model used for data set simulation. Following this, the work focuses on the development and testing of several neural network models, including a multilayer perceptron (MLP) and convolutional neural networks (CNN), aiming to improve the accuracy of object positioning in real-world applications such as autonomous vehicles and robotic manipulation.

Key words: stereo camera, neural network, extrinsic calibration, MLP, CNN

1. UVOD

Stereo kamera sustav koristi dvije ili više kamera koje su poznato pozicionirane jedna u odnosu na drugu kako bi snimile slike iz različitih perspektiva. Ovaj sustav inspiriran je ljudskim binokularnim vidom i omogućuje dobivanje dubinskih informacija o promatranoj sceni. Pomoću razlike u pozicijama objekata u slikama dobivenim iz pojedine kamere može se izračunati udaljenost promatranih objekata. Kako bi izmjereni rezultati bili precizni potrebno je napraviti kalibraciju stereokamere.

Kalibracija kamere ključni je korak u mnogim računalnim vizijskim sustavima te omogućuje precizno određivanje položaja i orijentacije kamere u prostoru. Dijeli se na intrinzičnu i ekstrinzičnu kalibraciju.

Intrinzična kalibracija odnosi se na postupak određivanja unutarnjih parametara kamere kao što su žarišna duljina, glavna točka, koeficijenti distorzija i veličina piksela. Ovi parametri opisuju karakteristike same kamere koje utječu na način na koji je objekt u stvarnom svijetu projiciran na senzor kamere.

Ekstrinzična kalibracija fokusira se na određivanje vanjskih parametara kamere, uključujući poziciju i orijentaciju kamere u odnosu na određeni koordinatni sustav. U kontekstu stereo kamera kalibrira se odnos između kamera, odnosno definiraju se vrijednosti udaljenosti i kuta između osi kamera. Pravilna ekstrinzična kalibracija od velikog je značaja za omogućavanje točne rekonstrukcije 3D scene pomoću dvije perspektive.

Postoji više metoda kalibracije kamere, geometrijskim uzorcima, kalibracijskim uzorkom i pomoću neuronskih mreža. Tradicionalne metode kalibracije kamera često se oslanjaju na precizno postavljanje i pozicioniranje kalibracijskih objekata, kao što su šahovske ploče. Iako ova metoda može biti vrlo precizna, ona također može biti i složena za implementaciju i zahtijeva značajan trud i vrijeme.

U ovom radu postupak ekstrinzične kalibracije stereo kamere biti će proveden pomoću neuronskih mreža. Korištenje neuronskih mreža za kalibraciju stereokamera predstavlja potencijalno učinkovitiju i robusniju alternativu tradicionalnim metodama. Neuronske mreže imaju sposobnost učenja složenih nelinearnih odnosa iz podataka, što može omogućiti automatizaciju i pojednostavljenje procesa kalibracije.

Cilj ovog rada je razviti model stereo kamere za prikupljanje potrebnih podataka te modele neuronskih mreža. Zatim će se usporediti preciznosti neuronskih mreža u ovisnosti o različitim konfiguracijama stereokamere i veličini mjerne i lokalizacijske greške. Ovaj pristup može značajno smanjiti potrebu za složenim geometrijskim algoritmima i poboljšati učinkovitost kalibracijskih procesa u stvarnim aplikacijama, kao što su autonomna vozila, robotske manipulacije i 3D skeniranje.

2. MODEL KAMERE

Modelom kamere matematički je opisan način na koji se točke iz trodimenzionalnog prostora projiciraju na dvodimenzionalnu ravninu slike. U nastavku je razrađena teorija i jednadžbe potrebne za napravljeni model.

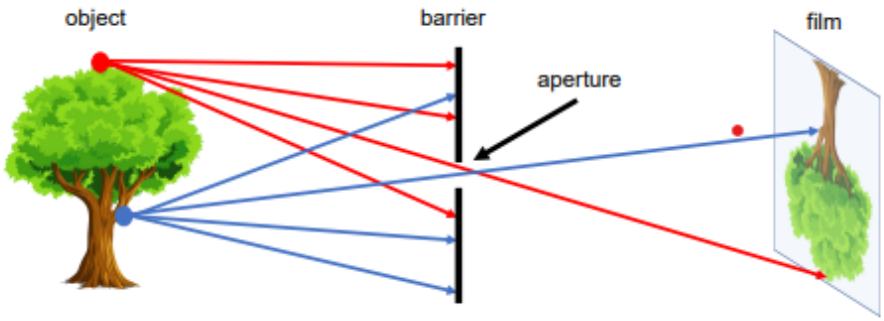
2.1. *Pinhole* model kamere

Pinhole model kamere temeljni je koncept u računalnom vidu i fotogrametriji, koji pruža pojednostavljen, ali učinkovit način opisivanja kako se trodimenzionalna scena projicira na dvodimenzionalnu ravninu slike.

Pinhole model simulira način na koji se svjetlost projicira kroz mali otvor (*pinhole*) i stvara sliku na površini iza njega. Princip *pinhole* modela poznat je iz koncepta *camera obscura*, gdje se uz pomoć malog otvora može dobiti obrnut i umanjen prikaz vanjskog svijeta na unutarnjem zidu zamračene prostorije.

Kada se objekt postavi ispred otvora, svjetlosne zrake koje dolaze s objekta prolaze kroz otvor i preslikavaju točke objekta na zid. Svaka svjetlosna zraka putuje pravocrtno, prolazeći kroz *pinhole* i spajajući određenu točku objekta s točkom na zidu, tako stvarajući projekciju.

Rezultat ovog procesa je obrnuta slika objekta, jer svjetlosne zrake iz gornjeg dijela objekta padaju na donji dio slike, i obrnuto (Slika 1). Oštrina slike ovisi o veličini *pinhole*-a; što je otvor manji, to će slika biti oštrija, ali i manje svjetlosti će proći kroz nju, što rezultira tamnijom slikom. S druge strane, veći otvor dopušta više svjetla, ali zbog difrakcije i prolaska većeg broja zraka sa svake točke objekta stvara zamućeniju sliku. [1]



Slika 1. Projiciranje objekta *pinhole* modelom [1]

2.2. Projekcija na senzor kamere

Kako bi objasnili matematički model *pinhole* kamere potrebno je povezati pojmove *pinhole* modela s pojmovima kamere te geometrijom modela. U srži *pinhole* modela kamere je odnos između 3D točke u prostoru i njezine odgovarajuće 2D točke u ravnini slike. Neka je **P** = (X,Y,Z) točka u trodimenzionalnom prostoru u odnosu na koordinatni sustav kamere, a **p** = (x,y) njezina projekcija na ravninu slike. Geometrijska projekcija može se izraziti koristeći svojstva sličnih trokuta, što dovodi do sljedećih osnovnih jednadžbi:

$$x = \frac{f \cdot X}{Z}, \quad y = \frac{f \cdot Y}{Z}. \tag{1}$$

Ovdje je f žarišna duljina kamere, koja predstavlja udaljenost između otvora i ravnine slike. Proces projekcije opisan ovim jednadžbama naziva se centralna projekcija, gdje se cijeli 3D prostor projicira na 2D ravninu kroz optički centar (*pinhole*).

Radi lakšeg računanja matricama, točke prikazujemo u homogenim koordinatama pa je točka **p** u homogenom obliku zapisana kao $\tilde{\boldsymbol{p}} = (\tilde{x}, \tilde{y}, \tilde{z})$ te vrijedi:

$$\tilde{x} = fX, \quad \tilde{y} = fY, \quad \tilde{z} = Z \tag{2}$$

Ili u matričnom zapisu:

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3}$$

Ako se točka **P** izrazi u homogenom obliku jednadžba (3) se proširuje:

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4}$$

S obzirom da je koordinatni sustav često u nekom kutu ravnine slike, a pretpostavka do sada je bila da se koordinatni sustav nalazi u središtu ravnine slike, potrebno ga je translatirati u kut slike:

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

gdje su p_x i p_y koordinate glavne točke u ravnini slike.

Faktoriziranjem matrice (5) dobije se:

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{6}$$

Što je zapisano u matričnom obliku:

$$\tilde{\mathbf{p}} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]\mathbf{P} \tag{7}$$

Gdje je **K** matrica kalibracije:

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

Ukoliko se koordinatni sustav svijeta i koordinatni sustav kamere ne poklapaju potrebno je prilagoditi jednadžbu (7). U ovom radu koordinatni sustavi obje kamere biti će rotirani i translatirani u odnosu na koordinatni sustav svijeta. Dok je translacija apsolutna, tj. određuje se u odnosu na referentni koordinatni sustav svijeta, rotacija je relativna te se računa oko koordinatnih osi kamera.

Jednadžba pinhole modela kamere s translacijom i rotacijom u odnosu na koordinatni sustav svijeta je:

$$\mathbf{P}_{\text{cam}} = \mathbf{R}(\mathbf{P} - \mathbf{C}), \tag{9}$$

gdje je $\mathbf{P}_{\mathbf{cam}}$ točka u koordinatnom sustavu kamere, **P** ista točka u koordinatnom sustavu svijeta te **R** matrica rotacije i **C** ishodište koordinatnog sustava kamere.

U konačnici prilagođena jednadžba *pinhole* modela kamere (7) postaje:

$$\tilde{\mathbf{p}} = \mathbf{KR}[\mathbf{I} \mid -\mathbf{C}]\,\tilde{\mathbf{P}} \tag{10}$$

Na kraju potrebnog matričnog računa potrebno je homogene koordinate točke $\tilde{\mathbf{p}}$ pretvoriti u nehomogene jednadžbama:

$$x = \frac{\tilde{x}}{\tilde{z}}, \quad y = \frac{\tilde{y}}{\tilde{z}} \tag{11}$$

Do sada navedene jednadžbe potrebno je prikazati pomoću diskretne ravnine slike jer u stvarnim digitalnim kamerama, ravnina slike odgovara senzorskom čipu koji je podijeljen na piksele te kojem se koordinatni sustav u-v se nalazi u kutu ravnine senzora (Slika 2.). Koordinate točaka na slici u pikselima označavaju se s (u,v), a povezane su s koordinatama točaka (x,y) na ravnini slike prema sljedećim jednadžbama:

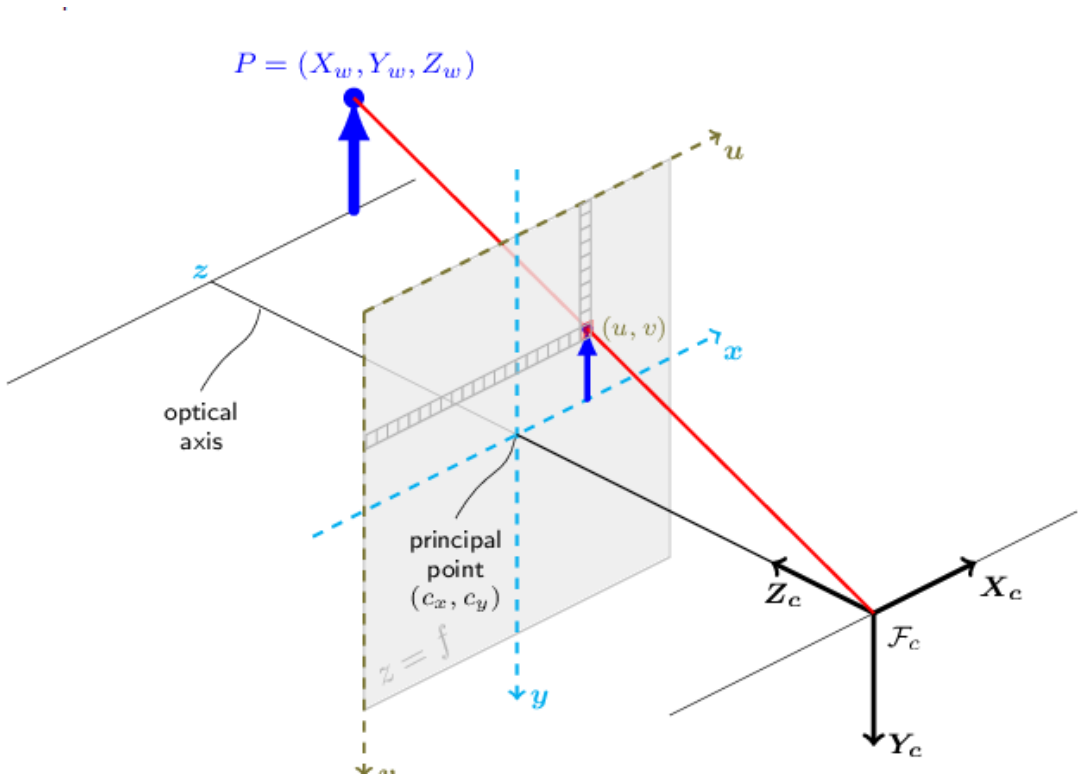
$$u = \frac{x}{\rho_w} + u_0, \quad v = \frac{y}{\rho_h} + v_0 \tag{12}$$

gdje ρ_w i ρ_h predstavljaju širinu i visinu piksela, a (u_0,v_0) koordinate glavne točke u pikselima:

$$u_0 = \frac{p_x}{\rho_w}, \quad v_0 = \frac{p_y}{\rho_h} \tag{13}$$

U tom obliku matrica kalibracije poprima oblik:

$$\mathbf{K} = \begin{bmatrix} f/\rho_w & 0 & u_0 \\ 0 & f/\rho_h & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$



Slika 2. Projekcija točke na senzor kamere [3]

Homogene točke $\tilde{\mathbf{p}}=(\tilde{u},\tilde{v},\tilde{w})$ u diskretnoj ravnini dobivene su pomoću jednadžbe (10) s modificiranom matricom kalibracije (14) te u konačnici su pretvorene u nehomogene:

$$u = \frac{\tilde{u}}{\tilde{w}}, \quad v = \frac{\tilde{v}}{\tilde{w}} \tag{15}$$

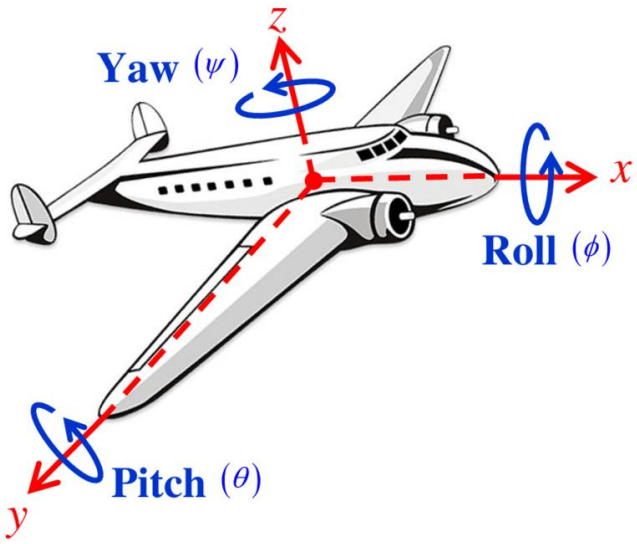
2.2.1. Matrica rotacija

Za rješavanje konačne jednadžbe *pinhole* modela kamere potrebno je definirati matricu rotacija **R**.

U ovom radu korišteni su Tait-Bryan Eulerovi kutovi kod koje se rotacija opisuje oko sve tri koordinatne osi, a odabrani redoslijed rotacija je Z-Y-X. To znači da će se za zadane kuteve prvo provesti rotacija oko z, zatim y te na kraju oko x koordinatne osi. Također, potrebno je naglasiti da je rotacija modela kamere relativna.

Matrice rotacija pisane su pomoću kuteva Θx, Θy i Θz, gdje je Θx kut zakreta oko x osi, tj. kut valjanja (*Roll*), Θy kut zakreta oko y osi, tj. kut posrtanja (*Pitch*) te Θz kut zakreta oko z osi, tj. kut skretanja (*Yaw*). Također, svaka osnovna matrica rotacija opisuje rotaciju oko jedne koordinatne osi te koriste kao varijablu jedan kut.

Smjer pozitivnog smjera kuta određuje se pravilom desne ruke kao što se vidi na Slika 3.



Slika 3. Rotacije oko koordinatnih osi [4]

Matrice rotacija definirane su jednadžbama:

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Theta_x) & -\sin(\Theta_x) \\ 0 & \sin(\Theta_x) & \cos(\Theta_x) \end{bmatrix} \tag{16}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos(\Theta_y) & 0 & \sin(\Theta_y) \\ 0 & 1 & 0 \\ -\sin(\Theta_y) & 0 & \cos(\Theta_y) \end{bmatrix} \tag{17}$$

$$\mathbf{R_z} = \begin{bmatrix} \cos(\Theta_z) & -\sin(\Theta_z) & 0 \\ \sin(\Theta_z) & \cos(\Theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{18}$$

gdje je $\mathbf{R_i}$ za i = x,y,z matrica rotacija oko i-te koordinatne osi.

Konačna korištena matrica rotacije dobivena je množenjem matrica:

$$\mathbf{R_{zyx}} = \mathbf{R_z}\,\mathbf{R_y}\,\mathbf{R_x} \tag{19}$$

$$\mathrm{R_{zyx}} = \begin{bmatrix} \cos(\Theta_y)\cos(\Theta_z) & a & b \\ \cos(\Theta_y)\sin(\Theta_z) & c & d \\ -\sin(\Theta_y) & \sin(\Theta_x)\cos(\Theta_y) & \cos(\Theta_x)\cos(\Theta_y) \end{bmatrix}, \tag{20}$$

gdje je :

$$\begin{aligned} a &= \sin(\Theta_x)\sin(\Theta_y)\cos(\Theta_z) - \cos(\Theta_x)\sin(\Theta_z) \\ b &= \sin(\Theta_x)\sin(\Theta_z) + \cos(\Theta_x)\sin(\Theta_y)\cos(\Theta_z) \\ c &= \cos(\Theta_x)\cos(\Theta_z) + \sin(\Theta_x)\sin(\Theta_y)\sin(\Theta_z) \\ d &= \cos(\Theta_x)\sin(\Theta_y)\sin(\Theta_z) - \sin(\Theta_x)\cos(\Theta_z). \end{aligned} \tag{21}$$

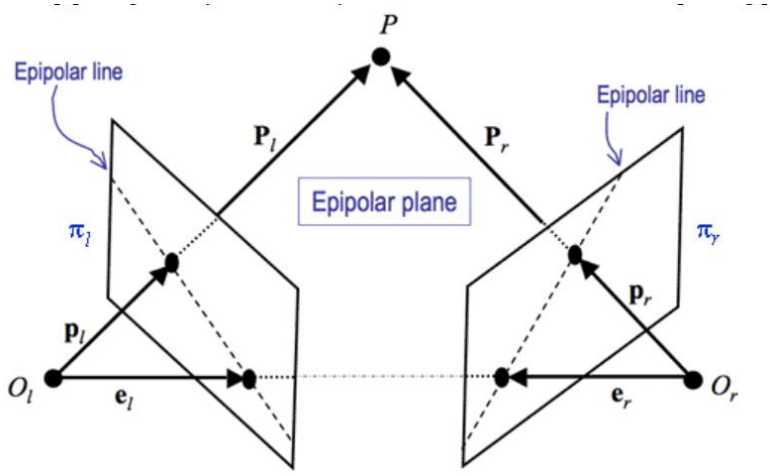
3. MODEL STEREOKAMERE

Stereo kamera ili dubinska kamera je sustav od dvije kamere inspiriran binokularnim ljudskim vidom. Cilj takvog sustava je mogućnost određivanja dubine slike, a poznavanjem udaljenosti objekta moguće je odrediti njegovu veličinu i točnu lokaciju u prostoru. Poznavanje tih informacija o objektu bitno je u robotici jer robotima omogućuje preciznu navigaciju, manipulaciju objektima, prepoznavanje okoline te donošenje autonomnih odluka u stvarnom vremenu.

3.1. Matematička podloga stereokamera sustava

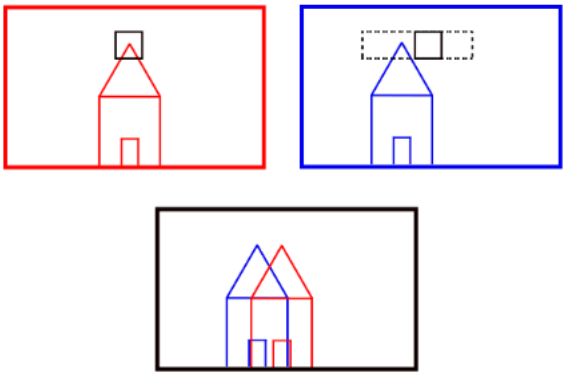
Jedna točka u prostoru preslikava se na različitim mjestima na slikama dvije kamera, ovisno o udaljenosti te točke od ravnina kamera. Kako bi se dobila z koordinata točke potrebno je znati disparitet, razliku u poziciji iste prostorne točke zabilježene u dvije slike. Što je disparitet veći, objekt je bliži; što je manji, objekt je dalje.

Zadatak pronalaženja odgovarajućeg piksela u obje slike znatno olakšava epipolarno ograničenje. Naime, u slučaju da su slike ispravljene, odnosno da nemaju iskrivljenje zbog leće, za piksel na lijevoj slici će se odgovarajući piksel nalaziti na epipolarnom pravcu, odnosno za odgovarajući piksel nećemo morati pretraživati cijelu sliku već samo točno određeni podskup (pravac) piksela slike.



Slika 4. Prikaz epipolarnog ograničenja [9]

Dodatno, ako su optičke osi kamera paralelne epipolarni pravci će postati horizontalni, što znači da piksel koji odgovara vrhu kuće na lijevoj slici biti će u istom retku i na desnoj slici (Slika 5). [7]



Slika 5. Disparitet [7]

Disparitet prostorne točke ovisi o konfiguraciji kamera; u navedenom slučaju (optičke osi kamera su paralelne) računa se prema:

$$d = x_L - x_R\,, \tag{21}$$

gdje je x_L x koordinata točke u koordinatnom sustavu lijeve kamere, a je x_R x koordinata iste točke u koordinatnom sustavu desne kamere.

Stereo kamere sastoje se od dviju kamera koje su postavljene na određenoj udaljenosti jedna od druge, razmak između njih se zove baza, *b*. Obje kamere snimaju istu scenu iz malo drugačije perspektive. Dubina Z točke u prostoru računa se pomoću pravila sličnih trokuta. Za lijevu kameru vrijedi:

$$\frac{x_L}{f} = \frac{X}{Z}, \tag{22}$$

a za desnu kameru:

$$\frac{x_R}{f} = \frac{X-b}{Z}. \tag{23}$$

Množenjem jednadžbi (22) i (23) sa koordinatom Z:

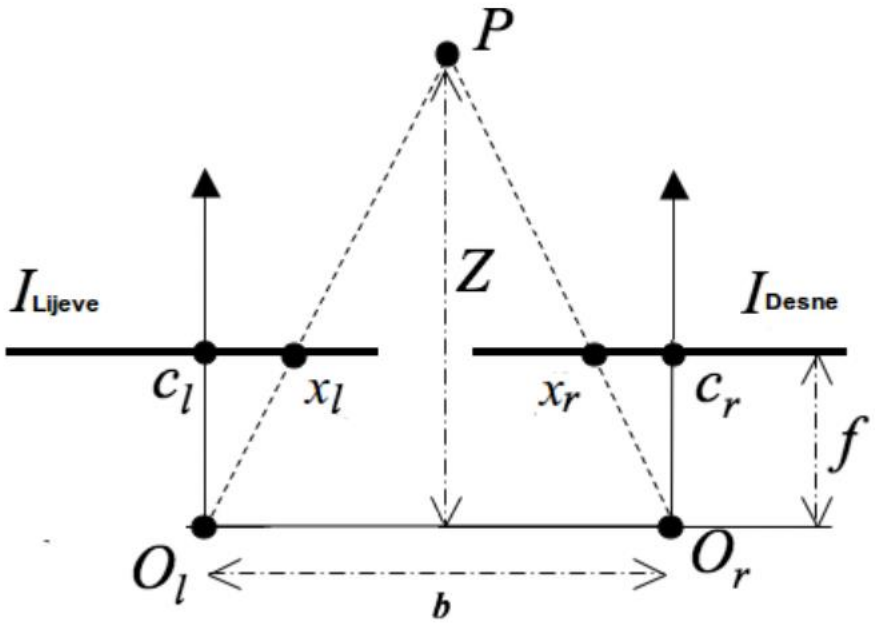
$$x_L \cdot Z = X \cdot f \tag{24}$$

$$x_R \cdot Z = (X - b) \cdot f. \tag{25}$$

Oduzimanjem jednadžbi (24) i (25) dobiva se konačna jednadžba za izračun dubine:

$$Z = \frac{f \cdot b}{d}, \tag{26}$$

gdje je d disparitet, a f žarišna duljina kamera u pikselima (Slika 6).



Slika 6. Triangulacija paralelne konfiguracije kamera [8]

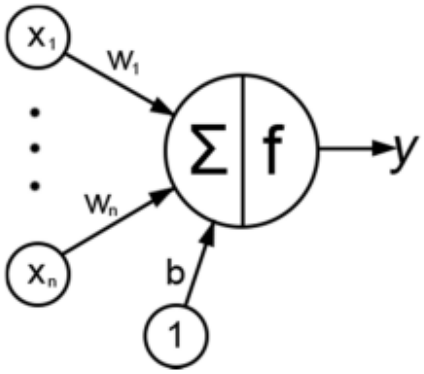
U slučaju da optičke osi kamera nisu paralelne, tj. ravnine slika obje kamere ne nalaze se u istoj ravnini, potrebno je provesti postupak rektifikacije. To je postupak projiciranja slika obje kamere na jednu zajedničku ravninu slike koja je definirana pravcem koji spaja optičke centre kamera. Na taj način slike su ponovno u istoj ravnini te im je moguće odrediti disparitet prema jednadžbi (21) [10]. Za detaljniju obradu rektifikacije čitatelj se upućuje na rad [10].

4. NEURONSKA MREŽA

Neuronska mreža je računalni model inspiriran strukturom i funkcijom ljudskog mozga, posebno neuronima, koristi se za analizu i obradu podataka. Osnovni element umjetne neuronske mreže je umjetni neuron, perceptron. Kao i biološki neuron, perceptron prima ulazne signale, obrađuje ih i generira izlaz ovisno o jačini impulsa, odnosno težini tih signala.

Umjetni neuron prima ulazne podatke koji mogu predstavljati različite vrste informacija, poput piksela u slici, riječi u tekstu ili senzorskih očitanja. Svaki ulaz x_i ima pridruženu težinu w_i koja predstavlja relativnu važnost tog ulaza u procesu donošenja odluke. Ulazni signali se množe s odgovarajućim težinama, a zatim se zbrajaju kako bi se dobila "*neto*" vrijednost ulaza.

Rezultirajuća vrijednost, nazvana "neto ulaz" ili "*net*", prosljeđuje se kroz aktivacijsku funkciju *f*. Aktivacijska funkcija određuje hoće li se neuron aktivirati i poslati signal dalje kroz mrežu. Ova funkcija može biti linearna ili nelinearna, a najčešće korištene su step funkcija, sigmoidna funkcija, ReLU (rectified linear unit), ili tanh (hiperbolička tangenta). Odabir aktivacijske funkcije ovisi o specifičnoj primjeni neuronske mreže i vrsti problema koji se rješava.



Slika 7. Umjetni neuron [12]

Jednadžba *net* vrijednosti je:

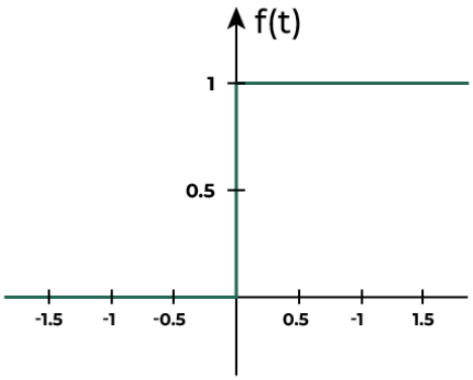
$$net = \sum_{i=1}^{n} w_i \cdot x_i + b, \tag{27}$$

gdje je *b* bias, konstanta koja se dodaje kako bi se prilagodila vrijednost neto ulaza, omogućujući modelu da bolje odgovara podacima. Bias pomaže pomicanju izlazne funkcije

lijevo ili desno, što može biti važno za optimalno učenje modela. Broj ulaznih neurona označen je sa *n*, sa x_i i-ta vrijednost ulaznog podatka te w_i težina i-tog ulaza.

Na izračunatoj vrijednosti *net* koristi se aktivacijska funkcija kako bi se odredio izlaz neurona. Na primjer, step funkcija odlučuje o izlazu prema sljedećem pravilu:

$$y = f(net) = \begin{cases} 1, \text{ ako je } net \geq 0 \\ 0, \text{ ako je } net < 0 \end{cases} \tag{28}$$



Slika 8. Step funkcija

Različite aktivacijske funkcije omogućuju umjetnoj neuronskoj mreži da modelira kompleksne nelinearne odnose između ulaza i izlaza. Kombiniranjem više neurona u slojevima (ulaznim, skrivenim i izlaznim) neuronske mreže mogu učiti karakteristike podataka na više razina apstrakcije, što ih čini vrlo moćnim alatima za zadatke poput prepoznavanja uzoraka, klasifikacije i regresije.

Umjetne neuronske mreže koriste različite algoritme učenja, od kojih je algoritam povratnog širenja pogreške (*backpropagation*) najpoznatiji i najčešće korišten. Algoritam *backpropagation* radi tako da minimizira pogrešku modela prilagođavanjem težina w_i i bias-a *b* kroz iterativni proces optimizacije.

Prvo, mreža prolazi kroz unaprijednu (*feedforward*) propagaciju, gdje se ulazni podaci provlače kroz mrežu i izračunava se izlaz na temelju trenutnih vrijednosti težina i bias-a koje su na početku nasumično odabrane vrijednosti. Zatim se izračunava pogreška, odnosno razlika između stvarnog izlaza (ciljne vrijednosti) i predviđenog izlaza mreže.

U koraku povratnog širenja pogreške, najčešće se koristi gradijentni spust za izračun promjena težina mreže, pri čemu se minimizira funkcija pogreške prilagođavanjem težina u smjeru negativnog gradijenta, omogućujući mreži da uči iz razlika između stvarnih i predviđenih vrijednosti. Na temelju tih gradijenata, težine i bias-evi se ažuriraju kako bi se smanjila ukupna pogreška mreže.

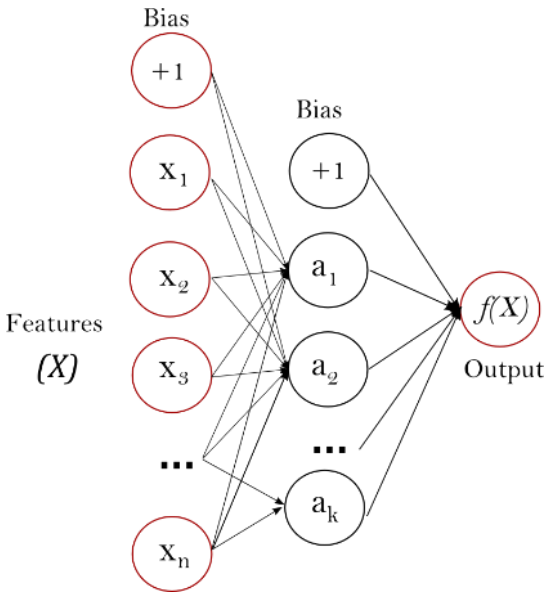
Backpropagation se ponavlja sve dok pogreška ne postane minimalna ili dok se ne postigne unaprijed definirani broj iteracija. Cilj algoritma je pronaći optimalne vrijednosti težina i bias-a koje minimiziraju razliku između stvarnog i predviđenog izlaza za skup podataka, što poboljšava performanse mreže u zadatku za koji je trenirana.

Kroz ovakve procese učenja, neuronske mreže se uspješno primjenjuju u mnogim područjima, uključujući prepoznavanje govora, obradu prirodnog jezika, računalni vid, biomedicinsku dijagnostiku, autonomna vozila, i mnoge druge.

4.1. Višeslojni perceptron (MLP)

Višeslojni perceptron (MLP) je vrsta umjetne neuronske mreže koja se sastoji od više slojeva perceptrona međusobno povezanih prema Slika 9. Sastoji se od najmanje tri sloja: ulaznog sloja, barem jednog skrivenog sloja, te izlaznog sloja. U ulaznom sloju nalaze se n ulaznih podataka, koji se prije ulaza u svaki perceptron množe s težinom w_i. Skriveni sloj sadrži k perceptrona, pri čemu svaki perceptron prije slanja izlazne vrijednosti u sljedeći sloj provodi dobivenu sumu kroz aktivacijsku funkciju. Iz skrivenog sloja podaci se zatim šalju ili u sljedeći skriveni sloj ili u izlazni sloj.

Zahvaljujući većem broju neurona i korištenju aktivacijskih funkcija, MLP je sposoban učiti nelinearne modele, što ga čini vrlo moćnim alatom za zadatke poput klasifikacije i regresije.



Slika 9. Višeslojni perceptron s jednim skrivenim slojem [13]

4.1.1. Levenberg-Marquardt (LM) algoritam

Treniranje mreže se može provoditi i pomoću Levenberg-Marquardt algoritma, koji je metoda optimizacije specifična za mreže kao što su MLP neuronske mreže. Ovaj algoritam je posebno koristan za probleme regresije ili klasifikacije gdje je potreban brz proces konvergencije.

Algoritam koristi iterativni pristup koji se temelji na sljedećim koracima:

- Izračunavanje gradijenta pogreške: Algoritam započinje izračunavanjem gradijenta funkcije gubitka s obzirom na težine mreže. To se postiže pomoću povratnog širenja pogreške (*backpropagation*).
- Prilagodba težina: LM algoritam koristi Hessian matricu (drugi derivati pogreške u odnosu na težine) kako bi bolje procijenio koliko treba prilagoditi težine da bi se minimizirala funkcija gubitka. Međutim, umjesto izravnog izračunavanja Hessian matrice, LM koristi aproksimaciju:

$$\mathbf{H} \approx \mathbf{J}^{\mathbf{T}} \cdot \mathbf{J}, \tag{29}$$

  gdje je **J** Jacobian matrica prvih derivata funkcije pogreške u odnosu na težine. Algoritam također koristi faktor prigušenja (*damping factor*) λ, koji omogućuje kontrolu između brzine konvergencije i stabilnosti.
- Ažuriranje parametara: Težine mreže se ažuriraju prema sljedećoj jednadžbi:

$$\mathbf{w_{new}} = \mathbf{w_{old}} - (\mathbf{J^T J} + \lambda \mathbf{I})^{-1} \mathbf{J^T e}, \tag{30}$$

gdje je:

w - vektor težina,

λ - *damping* faktor koji kontrolira veličinu koraka,

I - jedinična matrica,

e - vektor pogreške.

Ako se prilikom ažuriranja težina pogreška smanji, vrijednost λ se smanjuje, što omogućava veći korak u sljedećoj iteraciji. Ako se pogreška poveća, λ se povećava, čineći algoritam konzervativnijim. Levenberg-Marquardt algoritam kombinira Gauss-Newtonovu metodu i gradijentni spust kako bi postigao bržu konvergenciju od jednostavnog gradijentnog spusta. *Damping* faktor λ pruža stabilnost algoritmu, smanjujući rizik od oscilacija ili divergentnih rješenja, čak i u slučaju loše postavljenih problema. Ovaj algoritam je posebno učinkovit za

treniranje manjih neuronskih mreža, jer omogućuje optimizaciju modela uz visoku preciznost i stabilnost. Međutim, algoritam zahtijeva velike memorijske resurse i može zapeti u lokalnim minimumima, što ga čini neprikladnim za velike mreže ili vrlo složene probleme.

4.2. Konvolucijska neuronska mreža (CNN)

Konvolucijske neuronske mreže (CNN, engl. *Convolutional Neural Networks*) su vrsta dubokih neuronskih mreža dizajnirana za obradu podataka sa strukturom poput slika, audio zapisa ili vremenskih serija. CNN su inspirirane načinom na koji vizualni korteks ljudskog mozga obrađuje vizualne informacije. Vizualni korteks se sastoji od neurona koji su organizirani tako da prepoznaju osnovne vizualne značajke, kao što su rubovi, linije, kutevi, te složenije oblike. Na isti način, CNN koristi "konvolucijske slojeve" za prepoznavanje osnovnih uzoraka u podacima, poput rubova ili tekstura u slikama. Kako mreža prolazi kroz više slojeva, "uči" sve složenije i apstraktnije značajke.

CNN automatski uče hijerarhijske značajke iz podataka, što znači da se svaki sloj u mreži specijalizira za prepoznavanje različitih razina apstrakcije. Na primjer, prvi konvolucijski sloj može prepoznati jednostavne rubove u slici, dok se slojevi dublje u mreži mogu koristiti za prepoznavanje složenijih struktura, kao što su oblici ili objekti.

CNN se obično sastoje od nekoliko ključnih slojeva (Slika 10):

Konvolucijski slojevi (*Convolution layer*) - Ovi slojevi koriste male "filtere" ili "jezgre" koje klize preko ulaznih podataka (npr. slike) i traže specifične značajke, poput rubova, oblika ili tekstura. Kako se filter pomiče po podacima, stvara se "mapa značajki" koja prikazuje gdje su te značajke prisutne.

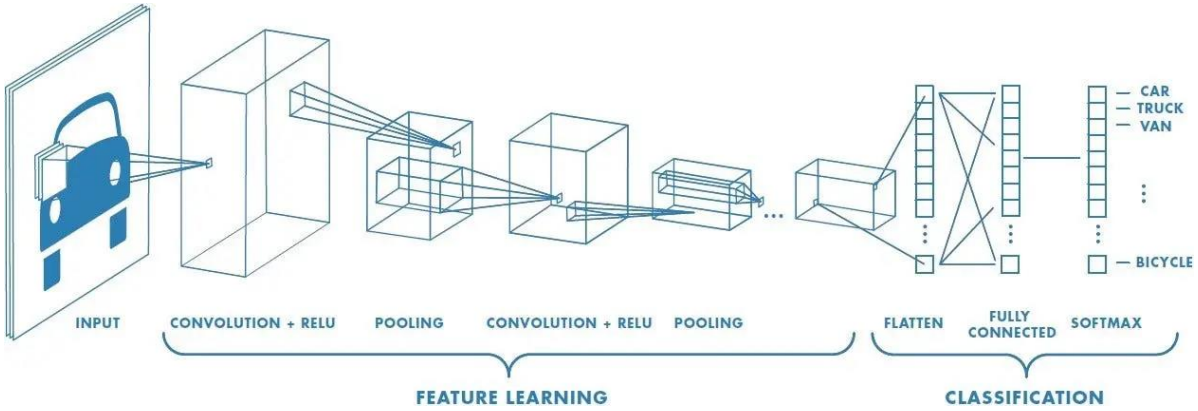
Slojevi za smanjivanje dimenzija (*Pooling Layers*) - Ovi slojevi smanjuju veličinu mape značajki kako bi smanjili broj podataka koje mreža mora obraditi, a pritom zadržali važne informacije. Najčešća metoda je *max pooling*, gdje se unutar svakog manjeg područja odabire najveća vrijednost, što omogućava modelu da zadrži ključne značajke dok smanjuje dimenzije.

Potpuno povezani slojevi (*Fully Connected Layers*) - Na kraju mreže, sve značajke iz prethodnih slojeva ulaze u jedan ili više slojeva u kojima je svaki neuron povezan sa svim neuronima iz prethodnog sloja. Ovi slojevi kombiniraju sve naučene značajke kako bi donijeli konačnu odluku ili klasifikaciju (npr. prepoznali što je na slici).

Kao i kod višeslojnog perceptrona (MLP), konvolucijske neuronske mreže (CNN) koriste aktivacijske funkcije kako bi 'naučile' nelinearne odnose u podacima. Međutim, dok se u MLP-

u treniranjem određuju optimalne težine između slojeva perceptrona, u CNN-u se optimiziraju filteri (jezgre) koji se koriste u konvolucijskim slojevima.

Zbog optimizacije filtera, CNN je iznimno učinkovit u prepoznavanju složenih uzoraka u podacima i rješavanju problema kao što su klasifikacija slika, analiza tekstova i vremenskih serija, te drugi zadaci koji zahtijevaju otkrivanje skrivenih značajki u ulaznim podacima.



Slika 10. Slojevi konvolucijske neuronske mreže [14]

4.2.1. 1D konvolucijske neuronske mreže

1D konvolucijske neuronske mreže su vrsta CNN-a koja se koristi za analizu jednodimenzionalnih podataka, poput vremenskih serija, audio signala i sekvenci riječi u tekstu, koji su organizirani kao nizovi ili sekvence.

1D CNN koristi konvolucijske operacije na jednoj dimenziji podataka kako bi detektirala lokalne obrasce u sekvencijalnom kontekstu. Svaki filter se primjenjuje duž jedne dimenzije ulaznog niza, izračunavajući skalarni umnožak težina filtera i dijelova ulaznog podatka:

$$z_j = \sum_{i=1}^{k} x_{i+j-1} \cdot w_i + b, \tag{29}$$

gdje je:

z_j - izlazna vrijednost za određeni pomak filtera j,

k - duljina filtera,

x_{i+j-1} - element ulazne vrijednosti,

w_i - težine filtera,

b - bias,

5. Programska izvedba

U ovom radu svi korišteni modeli izrađeni su u programskom jeziku *Python*. *Python* je izabran zbog svoje velike i aktivne zajednice koja nudi razne javno dostupne biblioteke i okvire koji podržavaju različite vrste primjena uključujući analizu podataka i strojno učenje.

U izradi ovog rada korištene su biblioteke:

- Tkinter - Služi za izradu grafičkog korisničkog sučelja (GUI) u *Python*-u.
- CustomTkinter - Omogućava veću mogućnost prilagođavanja izgleda korisničkog sučelja
- Numpy - Biblioteka za znanstveno računanje koja pruža podršku za rad s višedimenzionalnim nizovima i matricama, zajedno s velikim skupom matematičkih funkcija za operacije na njima
- Matplotlib - Omogućava stvaranje različitih vrsta grafova i dijagrama kako bi se ostvarila vizualizacija podataka
- Pandas - Pruža alate za manipulaciju, analizu i transformaciju podataka
- Tensorflow - Koristi se za izradu, treniranje i implementaciju modela strojnog učenja
- Datetime - Modul za rad s datumima i vremenima u Pythonu. Omogućava manipulaciju, usporedbu i formatiranje datuma i vremena.
- Pyrenn - Biblioteka otvorenog koda u Pythonu koja omogućuje jednostavno definiranje i treniranje *feedforward* neuronskih mreža, s posebnim fokusom na Levenberg-Marquardt algoritam optimizacije. Nudi fleksibilnu arhitekturu mreže, podršku za prilagođene funkcije aktivacije, automatsku normalizaciju podataka te alate za vizualizaciju performansi treniranja.
- Pickle - Modul za serijalizaciju i deserijalizaciju *Python* objekata. Omogućava spremanje *Python* objekata u datoteku ili prijenos preko mreže u binarnom obliku, te njihovo kasnije ponovno učitavanje i korištenje

5.1. Grafičko sučelje

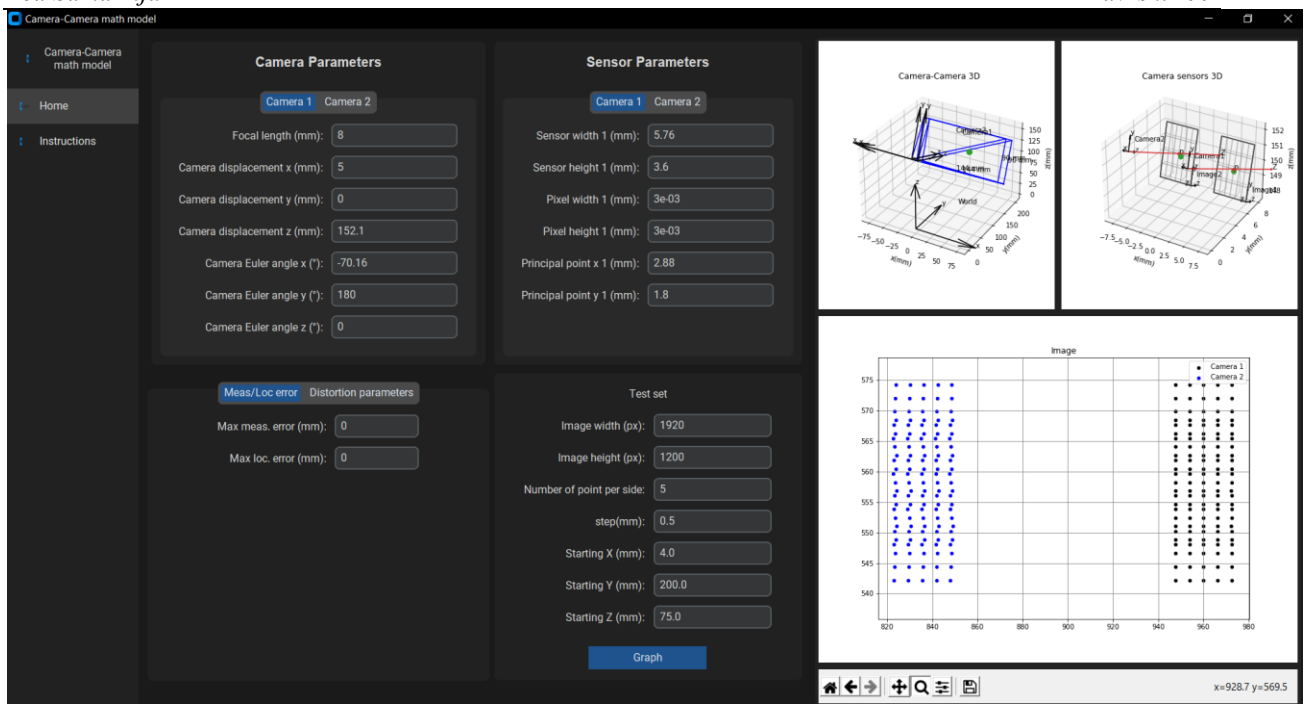
Kostur grafičkog sučelja preuzet je iz rada [6], a zatim je prilagođen i nadograđen kako bi zadovoljio zahtjeve ovog rada. Sučelje se sastoji od tri glavna dijela. Na Slika 11 prikazan je izgled grafičkog sučelja. Na lijevoj strani nalazi se navigacijska traka s koje se može birati

između uputa za uporabu programa i glavnog zaslona na kojem se izvršavaju simulacije (Slika 11). Središnji dio korisničkog sučelja podijeljen je u četiri segmenta:

- Parametri kamera - Ovaj segment sadrži dva dijela (jedan za svaku kameru) i omogućuje prilagodbu parametara kamere, kao što su njezin položaj i orijentacija u prostoru te žarišna duljina.
- Parametri senzora kamera - Sastoji se od dva prozora, svaki za pojedinu kameru, koji omogućuju promjenu dimenzija senzora i piksela te položaja glavne točke (*principal point*).
- Parametri distorzije i grešaka - Kao i prethodna dva segmenta i ovaj segment se sastoji od dva podprozora. U prvom prozoru mogu se mijenjati parametri distorzije leća, dok se u drugom podešavaju iznosi greške lokalizacije i greške mjerenja.
- Parametri 3D točaka - Ovaj segment omogućuje izmjenu parametara simuliranih mjernih točaka, uključujući početnu točku, broj točaka i razmak između njih. Također sadrži gumb za pokretanje simulacije.

Desni segment grafičkog sučelja prikazuje vizualizirana rješenja, koja se sastoje od tri grafa:

- Sustav kamera-kamera u 3D-u
- Senzori kamera u 3D-u
- Slika koju kamere vide u 2D prikazu



Slika 11. Grafičko sučelje - glavni prozor

5.2. Model stereo kamere

Model stereo kamere u ovom je radu napravljen pomoću dva modela *pinhole* kamere kojima se u grafičkom sučelju mogu proizvoljno mijenjati intrinzični parametri kamera, distorzijski parametri, mjerna i lokalizacijska greška te pozicije i orijentacije kamera.

Radi potrebe za velikim skupom podataka, za treniranje i testiranje neuronskih mreža, u grafičkom sučelju moguće je odrediti broj točaka po x, y i z koordinatnim osima, odnosno broj prostornih točaka koje će se projicirati na ravnine kamera. Također, u grafičkom sučelju moguće je i odrediti početnu poziciju točke u prostoru. Na taj način izbjegnuta je uporaba modela stereo kamera sustava iz 3.1.

5.2.1. Model pinhole kamere

Model *pinhole* kamere preuzet je s [2], uz to je nadograđen i izmjenjen kako bi zadovoljio zahtjeve rada. Model se sastoji od glavnog programa i nekoliko dodatnih modula koji služe za grupiranje različitih funkcionalnosti modela. Dani moduli su:

_frames - modul koji služi za stvaranje 3D koordinatnih sustava

_homogenus - modul koji omogućuje pretvorbu nehomogenih koordinata u homogene i obrnuto

_images - modul koji služi za generiranje ravnine slike

_matrices - modul koji omogućuje stvaranje potrebnih matrica:

- Matricu rotacije
- Kalibracijsku matricu
- Matricu projekcije
- Matricu ravnine.

_points - modul pomoću kojeg se dobiju koordinate projiciranih 3D točaka na ravninu slike kamere. Uz to, omogućuje vizualizaciju navedenih 3D i 2D točaka.

_principal axis - modul koji omogućuje inicijalizaciju i 3D vizualizaciju glavne osi kamere.

_utils - modul koji služi za crtanje 3D vektora.

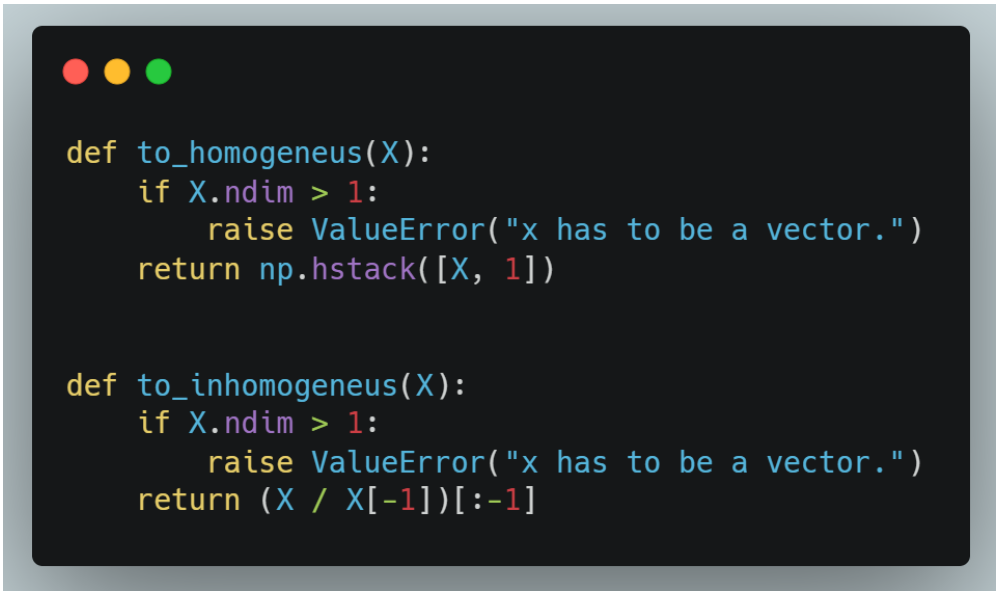
5.3. Istaknuti dijelovi koda potrebni za generiranje testnog seta podataka

U nastavku su opisani ključni dijelovi programskog koda potrebni za generiranje testnog skupa podataka u svrhu treniranja i testiranja neuronskih mreža.

5.3.1. Funkcije homogenosti

Slika 12 prikazuje programski kod u kojem se nalaze dvije Python funkcije koje služe za pretvorbu koordinata između homogenog i nehomogenog oblika, to su:

- to_homogeneous(X) - Ova funkcija uzima vektor koji predstavlja nehomogene koordinate i pretvara u homogene koordinate. Ovaj postupak olakšava računske manipulacije s točkama u različitim koordinatnim sustavima.
- to_inhomogeneous(X) - Ova funkcija služi za pretvorbu homogenih koordinata natrag u nehomogeni oblik. To postiže dijeljenjem svakog elementa vektora homogenizirajućom komponentom (zadnjim elementom) te uklanjanjem te komponente iz rezultirajućeg vektora. Ovaj proces omogućuje vraćanje koordinata u standardni oblik koji se koristi za daljnje računske operacije.



```python
def to_homogeneus(X):
    if X.ndim > 1:
        raise ValueError("x has to be a vector.")
    return np.hstack([X, 1])


def to_inhomogeneus(X):
    if X.ndim > 1:
        raise ValueError("x has to be a vector.")
    return (X / X[-1])[:-1]
```

Slika 12. Funkcije se korištene za pretvorbu koordinata između homogenog i nehomogenog oblika

5.3.2. Matrice rotacije

Funkcije prikazane na Slika 13 služe za generiranje matrica rotacije koje omogućuju rotaciju točaka ili objekata u trodimenzionalnom prostoru, one su

- _get_roll_matrix(theta_x)
- _get_pitch_matrix(theta_y)
- _get_yaw_matrix(theta_z).

Definiraju tri osnovne matrice rotacije: roll (nagib), pitch (naginjanje) i yaw (skretanje), koje predstavljaju rotacije oko X, Y i Z osi. Svaka od ovih funkcija prima odgovarajući kut rotacije (theta_x, theta_y ili theta_z) kao ulaz i koristi trigonometrijske funkcije kosinusa i sinusa iz NumPy biblioteke kako bi izračunala pripadajuću matricu rotacije.

```python
# Roll matrix Rx
def _get_roll_matrix(theta_x):
    Rx = np.array(
        [
            [1.0, 0.0, 0.0],
            [0.0, np.cos(theta_x), -np.sin(theta_x)],
            [0.0, np.sin(theta_x), np.cos(theta_x)]
        ]
    )
    return Rx


# Pitch matrix Ry
def _get_pitch_matrix(theta_y):
    Ry = np.array(
        [
            [np.cos(theta_y), 0.0, np.sin(theta_y)],
            [0.0, 1.0, 0.0],
            [-np.sin(theta_y), 0.0, np.cos(theta_y)]
        ]
    )
    return Ry


# Yaw matrix Rz
def _get_yaw_matrix(theta_z):
    Rz = np.array(
        [
            [np.cos(theta_z), -np.sin(theta_z), 0.0],
            [np.sin(theta_z), np.cos(theta_z), 0.0],
            [0.0, 0.0, 1.0]
        ]
    )
    return Rz
```

Slika 13. Funkcije matrice rotacije oko jedne osi

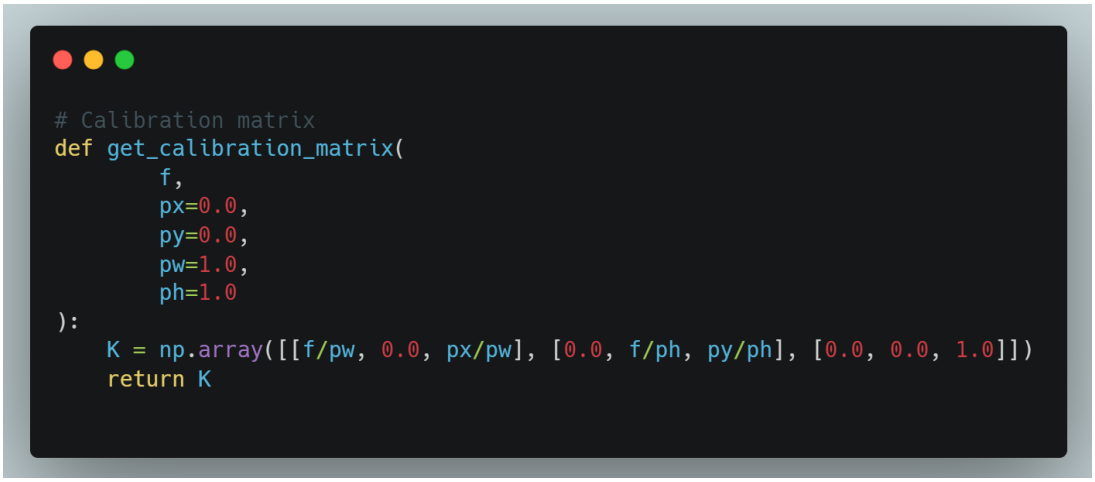
Na Slika 14 prikazana je funkcija get_rotation_matrix(theta_x, theta_y, theta_z). Ova funkcija poziva prethodno definirane funkcije kako bi dobila matrice rotacija oko X, Y i Z osi, a zatim ih množi u redoslijedu **Rz**, **Ry** pa **Rx** kako bi stvorila konačnu matricu rotacije **R** koja se koristi za izvođenje složenih operacija u prostoru.

```python
# Rotation matrix
def get_rotation_matrix(theta_x=0.0, theta_y=0.0, theta_z=0.0):
    # Roll matrix
    Rx = _get_roll_matrix(theta_x)
    # Pitch matrix
    Ry = _get_pitch_matrix(theta_y)
    # Yaw matrix
    Rz = _get_yaw_matrix(theta_z)
    return Rz @ Ry @ Rx
```

Slika 14. Funkcija matrice rotacije

5.3.3. Matrice projekcije

Na Slika 15. prikazana je funkcija get_calibration_matrix, koja se koristi za generiranje kalibracijske matrice kamere. Ova funkcija uzima nekoliko ulaznih parametara, uključujući žarišnu duljinu kamere *f*, koordinate glavne točke na slici *px* i *py*, te širinu i visinu piksela *pw* i *ph*. Pomoću navedenih ulaznih parametara stvara matricu **K** kojom su definirani unutarnji parametri kamere.



```python
# Calibration matrix
def get_calibration_matrix(
        f,
        px=0.0,
        py=0.0,
        pw=1.0,
        ph=1.0
):
    K = np.array([[f/pw, 0.0, px/pw], [0.0, f/ph, py/ph], [0.0, 0.0, 1.0]])
    return K
```

Slika 15. Funkcija za kreiranje kalibracijske matrice

5.3.4. Matrica projekcije

Na Slika 16. prikazana je funkcija get_projection_matrix koja se koristi za generiranje idealne projekcijske matrice kamere. Funkcija prvo poziva get_calibration_matrix za stvaranje kalibracijske matrice **K**, a zatim get_rotation_matrix za generiranje matrice rotacije **R**. Nakon

toga, funkcija koristi kalibracijsku matricu **K** i matricu rotacije **R** kako bi stvorila projekcijsku matricu **P**, koja uključuje i intrinzične i ekstrinzične parametara kamere. Projekcijska matrica **P** je ključna za transformaciju 3D koordinata točaka iz prostora u 2D koordinate na slici.

```python
# Projection matrix - ideal
def get_projection_matrix(
        f,
        px=0.0,
        py=0.0,
        pw=1.0,
        ph=1.0,
        C=(0.0, 0.0, 0.0),
        theta_x=0.0,
        theta_y=0.0,
        theta_z=0.0
):
    K = get_calibration_matrix(f=f, px=px, py=py, pw=pw, ph=ph)
    R = get_rotation_matrix(theta_x=theta_x, theta_y=theta_y,
theta_z=theta_z)
    P = K @ R @ np.c_[np.eye(3), -np.asarray(C)]
    return P
```

Slika 16. Funkcija izračuna projekcijske matrice kamere

5.3.5. Generiranje testnih podataka u 3D prostoru

Prostorne točke su generirane trostrukom *for* petljom, kreirajući pritom skup podataka oblika kocke s k (u navedenom programskom kodu dana varijabla je nazvana stranica) točaka po stranici, odnosno ukupno k^3 točaka gdje je k proizvoljno podesiv iz korisničkog sučelja. Također, iz sučelja je moguće prilagoditi početne koordinate na x, y i z osi te razmak između točaka [Slika 11]. Odabirom $k = 15$ postignut je velik skup prostornih točaka koje će svojim projekcijama na 2D slike kamera činiti skup od 3375 mjerenja. Velik skup podataka potreban je za uspješno treniranje neuronskih mreža.

Tijekom generacije točaka omogućeno je u grafičkom sučelju dodati mjernu grešku, Mes_{error} koja ima uniformnu slučajnu razdiobu oko zadane vrijednosti. Dodavanje te greške uvedeno je kao opcija jer stvarni sustavi uvode određene nepreciznosti u mjerenje, što ovu opciju čini korisnom za simuliranje stvarnih uvjeta. Također, ta greška unosi nelinearnost prema kojoj se neuronske mreže moraju prilagoditi.

Slika 17 prikazuje programsku izvedbu generiranja testnih podataka, uključujući prilagodljive parametre i dodanu grešku mjerenja.

Ovaj pristup omogućuje fleksibilno generiranje velikih skupova podataka potrebnih za učinkovito treniranje neuronskih mreža uz mogućnost prilagodbe parametara prema specifičnim zahtjevima korisnika.

```python
# Generate 3D Points
points_3D = []
for i in range(0, stranica):
    for j in range(0, stranica):
        for k in range(0, stranica):
            points_3D.append(np.array([
                x + i * step + random.uniform(-mes_error, mes_error),
                y + j * step + random.uniform(-mes_error, mes_error),
                z + k * step + random.uniform(-mes_error, mes_error)
            ]))

# Convert to DataFrame and Save
points_3D_df = pd.DataFrame(data=points_3D)
points_3D_df = points_3D_df.rename(columns={0: 'x', 1: 'y', 2: 'z'})
points_3D_df.to_csv("rezultati.csv")
```

Slika 17. Programski kod za generiranje testnih podataka u 3D prostoru

5.3.6. Projekcija 3D točke na 2D ravninu slike

Na Slika 18 prikazana je funkcija calculate_2D, koja služi za izračunavanje 2D koordinata točke u prostoru na temelju zadanih parametara kamere. Funkcija najprije poziva get_projection_matrix kako bi generirala projekcijsku matricu **P** koristeći zadane parametre. Zatim se 3D koordinate dane točke homogeniziraju pomoću funkcije to_homogeneous, te se potom transformiraju pomoću projekcijske matrice **P** kako bi se dobile homogene 2D koordinate. Ove homogene koordinate se zatim pretvaraju natrag u nehomogeni oblik koristeći funkciju to_inhomogeneous. Dobivene 2D koordinate predstavljaju projekciju 3D točke na dvodimenzionalnu ravninu slike.

```python
def calculate_2D(
        point,
        f,
        px=0.0,
        py=0.0,
        pw=1.0,
        ph=1.0,
        C=(0.0, 0.0, 0.0),
        theta_x=0.0,
        theta_y=0.0,
        theta_z=0.0
):
    P = get_projection_matrix(
        f,
        px=px,
        py=py,
        pw=pw,
        ph=ph,
        theta_x=theta_x,
        theta_y=theta_y,
        theta_z=theta_z,
        C=C,
    )

    x = to_inhomogeneus(P @ to_homogeneus(point))

    return x
```

Slika 18. Funkcija za Izračun Projekcije 3D Točke na 2D Ravninu Slike

5.3.7. Uključivanje distorzije i lokalizacijske greške u 2D točku na ravnini slike

Na Slika 19 prikazana je funkcija calculate_2D_with_distortion, koja omogućuje izračunavanje 2D koordinata točke na ravnini slike uzimajući u obzir distorziju objektiva kamere i lokalizacijsku grešku. Funkcija prima parametre kao što su:

- koeficijenti radijalne distorzije (*k1*, *k2*, *k3*)
- tangencijalne distorzije (*p1*, *p2*)
- žarišnu duljinu (*f*)
- koordinate glavne točke izračunate funkcijom calculate_2D (*px*, *py*)
- širina i visina piksela (*pw*, *ph*)
- Maksimalni iznos lokalizacijske greške (*loc_error*)

Funkcija najprije računa normalizirane koordinate točke (x_n, y_n) u odnosu na glavnu točku kamere i određuje radijalnu udaljenost (r2). Na to se primjenjuju koeficijenti distorzije kako bi se izračunale iskrivljene koordinate (x_dist, y_dist) koje uzimaju u obzir radijalne i tangencijalne utjecaje distorzije. Zatim se na ove koordinate dodaje slučajna lokalizacijska greška u zadanom rasponu sa unifornom razdiobom, čime se dobivaju konačne iskrivljene 2D koordinate. Lokalizacijska greška je uvedena zbog nelinearnosti stvarnih sustava i optičkih distorzija te se uzima u obzir kako bi se realno modelirale i ispravile nesavršenosti.

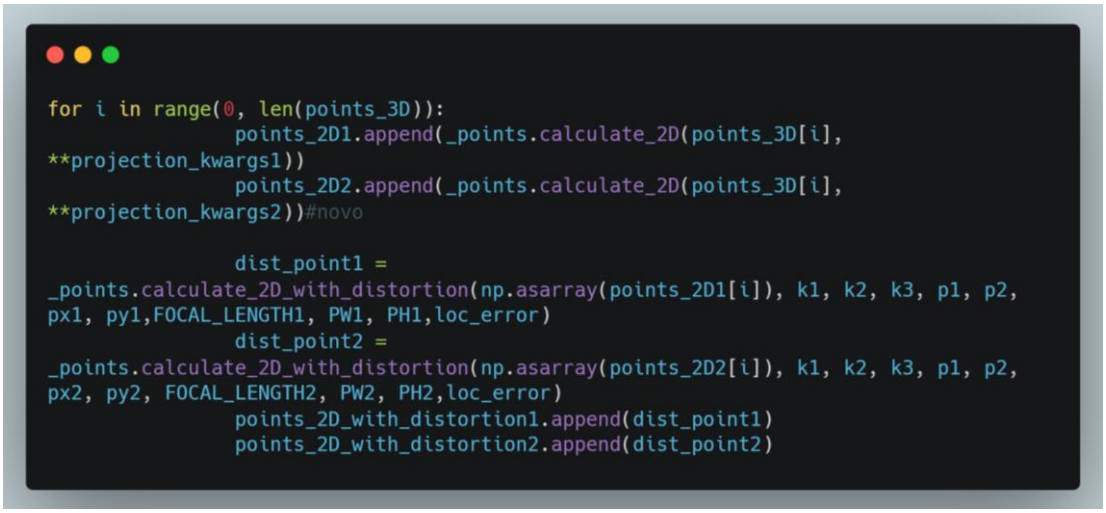
```
def calculate_2D_with_distortion(
        point,
        k1,
        k2,
        k3,
        p1,
        p2,
        px,
        py,
        f,
        pw,
        ph,
        loc_error,
):

    x_n = (point[0] - px)/(f/pw)
    y_n = (point[1] - py)/(f/ph)
    r2 = x_n**2 + y_n**2

    x_dist = x_n*(1 + k1*r2 + k2*(r2**2) + k3*(r2**3)) + (2*p1*x_n*y_n + p2*(r2 + 2*
(x_n**2)))
    y_dist = y_n*(1 + k1*r2 + k2*(r2**2) + k3*(r2**3)) + (p1*(r2 + 2*(y_n**2)) +
2*p2*x_n*y_n)

    x_d = (x_dist * (f/pw)) + px+random.uniform(-loc_error,loc_error)
    y_d = (y_dist * (f/ph)) + py+ +random.uniform(-loc_error,loc_error)

    point_dist = np.array([x_d, y_d])

    return point_dist
```

Slika 19. Funkcija za izračun točke na 2D ravnini slike s distorzijom i lokalizacijskom greškom

5.3.8. Generiranje 2D Projiciranih Točaka

Za treniranje neuronskih mreža uz originalnu poziciju točke u prostoru, potrebni su i podaci projekcije te točke na ravninu slike obije kamere, pošto se radi o stereo kameri. Slika 20 prikazuje kod korišten za generiranje 2D projekcijskih točaka. *For* petljom duljine liste points_3D (lista points_3D sadrži sve testne točke u 3D prostoru, u ovom radu 3375) vrši se iteracija po svakoj točki u listi kako bi se izračunale njene 2D projekcije pomoću funkcije

calculate_2D. Dobivene vrijednosti 2D projekcija potom se prosljeđuju funkciji calculate_2D_with_distortion, zajedno s prije navedenim parametrima. Ova funkcija generira iskrivljene 2D koordinate. Na taj način se dobivaju svi potrebni podaci za treniranje neuronskih mreža, omogućujući realističnu simulaciju projekcija u uvjetima stvarnog svijeta.



```
for i in range(0, len(points_3D)):
                points_2D1.append(_points.calculate_2D(points_3D[i],
**projection_kwargs1))
                points_2D2.append(_points.calculate_2D(points_3D[i],
**projection_kwargs2))#novo

                dist_point1 =
_points.calculate_2D_with_distortion(np.asarray(points_2D1[i]), k1, k2, k3, p1, p2,
px1, py1,FOCAL_LENGTH1, PW1, PH1,loc_error)
                dist_point2 =
_points.calculate_2D_with_distortion(np.asarray(points_2D2[i]), k1, k2, k3, p1, p2,
px2, py2, FOCAL_LENGTH2, PW2, PH2,loc_error)
                points_2D_with_distortion1.append(dist_point1)
                points_2D_with_distortion2.append(dist_point2)
```

Slika 20. Generiranje 2D projiciranih točaka

5.4. Model neuronskih mreža

Za izradu neuronskih mreža odabrana je *Python* biblioteka *Keras* koja je integrirana u *TensorFlow*. *Keras* je odabran zbog jednostavnosti korištenja i brze izrade modela, dok mu *TensorFlow* pruža snažnu pozadinsku podršku za izvođenje složenih matematičkih operacija, optimizaciju i skalabilnost na različitim hardverskim platformama.

U svim neuronskim mrežama podatci su učitani i obrađeni prema Slika 21. Koristeći biblioteku pandas podatci su učitani iz CSV dokumenta te raspoređeni u X i y liste, tj. raspodjeljeni su na ulazne i izlazne podatke. Nadalje, podatci su normalizirani radi poboljšanja učinkovitosti i brže konvergencije algoritma treniranja. Konačne liste podataka su razdvojene na podatke za treniranje i testiranje neuronskih mreža.

```
# Load data from CSV file
data = pd.read_csv('90_kamere_meas_loc_0.1.csv')

# Extract input features (cameras) and target values (3D coordinates)
X = data.iloc[:, 4:8].values
y = data.iloc[:, 1:4].values

#normalizing
scaler = StandardScaler()
X_scaled =  scaler.fit_transform(X)
y_scaled =  scaler.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.1, random_state=42)
```

5.4.1. Višeslojni perceptron

Prvo je inicijaliziran sekvencijalni (*Sequential*) model koji omogućava dodavanje slojeva i prihvaća podatke u obliku tenzora. Zatim je modelu dodan prvi skriveni sloj (*Dense*) sa 30 neurona u kojem je također definiran broj ulaznih neurona (dimenzija 4, što odgovara 2D koordinatama obje kamere). Ovaj sloj koristi ReLU (Rectified Linear Unit) kao aktivacijsku funkciju:

$$\text{ReLU}(net) = \begin{cases} net\ ako\ je\ net\ \geq\ 0 \\ 0\ ako\ je\ net\ <\ 0 \end{cases} \tag{30}$$

Sljedeći sloj dodaje 20 neurona i također koristi ReLU aktivacijsku funkciju [15]. Na kraju se dodaje izlazni sloj s 3 neurona, koji daje predviđene 3D koordinate točke.

Slika 21. Obrada podataka

Nakon definiranja strukture modela, definiran je algoritam optimizacije (*Adam, Adaptive Moment Estimation*) i funkcija gubitka (*MAE, Mean Absolute Error*), koja minimizira srednju apsolutnu pogrešku između predviđenih i stvarnih vrijednosti. Srednja greška je definirana jednadžbom:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{31}$$

gdje je:

n – veličina skupa podataka

y_i - stvarna vrijednost za i-ti primjer

$\hat{y}_i$ - predviđena vrijednost modela za i-ti primjer.

Model se zatim trenira koristeći skup podataka za treniranje (X_train, y_train) s definiranim brojem epoha i veličinom serije (batch_size). Parametar verbose=1 omogućava ispis detalja o napretku treniranja.

Na kraju, model se evaluira na testnom skupu podataka (X_test, y_test), a na kraju je izračunata vrijednost gubitka, što omogućava procjenu uspješnosti modela na testnim podacima.

```
# Create a Sequential model
model = Sequential()

# Add layers to the model
model.add(Dense(30, input_dim=4, activation='relu'))  # Input: 4 (2 coordinates for each camera)
model.add(Dense(20, activation='relu'))
model.add(Dense(3))  # Output: 3 coordinates for the 3D point

# Compile the model
model.compile(optimizer='Adam', loss='MAE')

# Training the model
epochs = 1000
batch_size = 32

model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1)

# Evaluate the model on the testing set
loss = model.evaluate(X_test, y_test, batch_size=batch_size)
print(f'Test Loss: {loss}')
```

Slika 22. Model višeslojnog perceptrona

5.4.2. Višeslojni perceptron treniran Levenberg-Marquardt algoritmom

Na Slika 23 vidi se model višeslojnog perceptrona s jednim skrivenim slojem kreiran pomoću biblioteke *Pyrenn*. Model je inicijaliziran korištenjem funkcije CreateNN, gdje je definiran ulazni sloj s 4 neurona, skriveni sloj s 17 neurona te izlazni sloj s 3 neurona. Za treniranje modela primijenjen je Levenberg-Marquardt (LM) optimizacijski algoritam, koji je poznat po brzoj konvergenciji u problemima nelinearne regresije i dobroj optimizaciji mrežnih parametara. Levenberg-Marquardt algoritam je metoda optimizacije koja kombinira pristupe Newtonove metode i gradijentnog spusta (*gradient descent*) za rješavanje nelinearnih problema najmanjih kvadrata. Treniranje modela izvedeno je pomoću funkcije train_LM, s maksimalnim brojem iteracija postavljenim na 1000 (k_max=1000) i kriterijem zaustavljanja (E_stop=1e-5), uz omogućenu detaljnu izlaznu informaciju o procesu treniranja (verbose=True).

Nakon treniranja, mreža je testirana na skupu testnih podataka pomoću funkcije NNOut, koja je korištena za procjenu izlaznih vrijednosti modela (y_test_est i y_train_est) na temelju ulaznih

podataka testnog skupa (X_test.T) i skupa za treniranje (X_train.T). Rezultati pokazuju procijenjene vrijednosti modela, što omogućuje daljnju analizu performansi mreže.

```
# Create the model with 1 hidden layer with 17 neurons
net = prn.CreateNN([4, 17, 3])

# Train the model using Levenberg-Marquardt algorithm
net = prn.train_LM(X_train.T, y_train.T, net, verbose=True, k_max=1000, E_stop=1e-5)

# Test the model
y_test_est = prn.NNOut(X_test.T, net)
y_train_est = prn.NNOut(X_train.T, net)
```

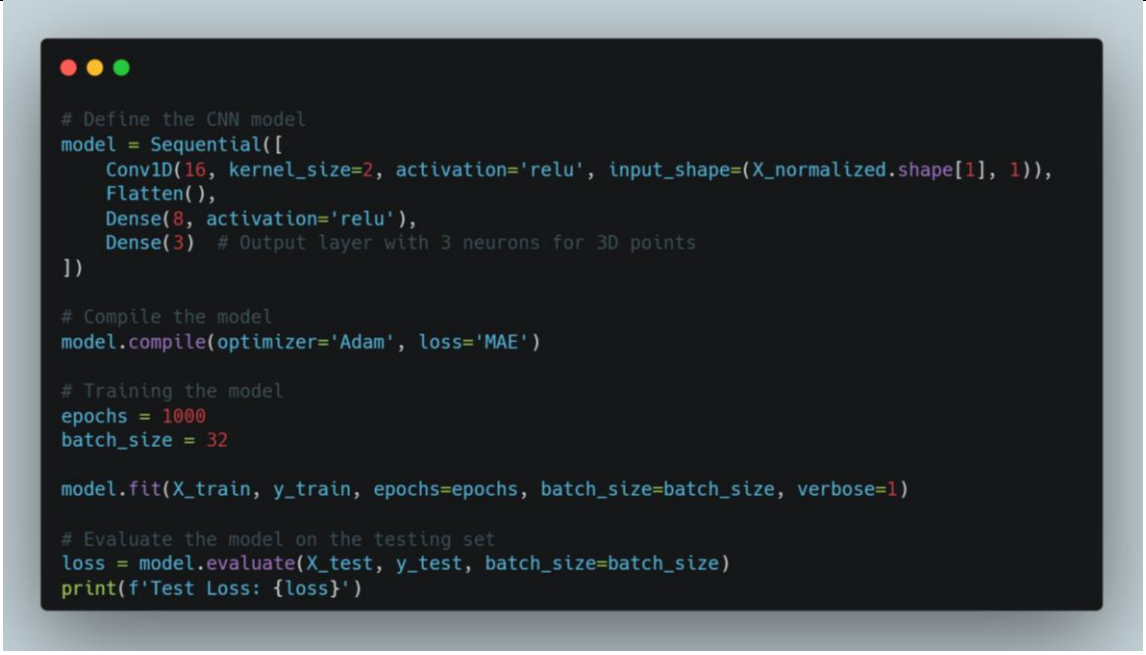
Slika 23. MLP treniran LM algoritmom

5.4.3. Konvolucijska neuronska mreža

Na Slika 24 je model konvolucijske neuronske mreže (CNN) definiran pomoću *Keras* biblioteke. Model je sastavljen od nekoliko slojeva, počevši s 1D konvolucijskim slojem (Conv1D) koji sadrži 16 filtera s veličinom kernela 2 i ReLU aktivacijskom funkcijom. Ulazni oblik modela definiran je pomoću parametra input_shape koji se temelji na dimenzijama normaliziranih ulaznih podataka. Nakon konvolucijskog sloja, model koristi sloj Flatten za pretvaranje višedimenzionalnih podataka u jednoslojni vektor, koji se zatim prosljeđuje kroz dva potpuno povezana (Dense) sloja - jedan sa 8 neurona s ReLU aktivacijom i izlazni sloj sa 3 neurona, koji predstavlja 3D točke.

U ovom modelu je također korišten Adam optimizacijski algoritam i gubitak definiran kao srednja apsolutna greška (MAE). Treniranje modela izvedeno je kroz 1000 epoha s veličinom grupe podataka (batch_size) postavljenom na 32. Funkcija fit korištena je za treniranje modela s ulaznim podacima X_train i odgovarajućim izlaznim podacima y_train, uz detaljan izlaz (verbose=1).

Na kraju, model je evaluiran na testnom skupu podataka pomoću funkcije evaluate, koja računa i ispisuje vrijednost gubitka (loss) na testnom skupu (X_test, y_test).



```
# Define the CNN model
model = Sequential([
    Conv1D(16, kernel_size=2, activation='relu', input_shape=(X_normalized.shape[1], 1)),
    Flatten(),
    Dense(8, activation='relu'),
    Dense(3)  # Output layer with 3 neurons for 3D points
])

# Compile the model
model.compile(optimizer='Adam', loss='MAE')

# Training the model
epochs = 1000
batch_size = 32

model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1)

# Evaluate the model on the testing set
loss = model.evaluate(X_test, y_test, batch_size=batch_size)
print(f'Test Loss: {loss}')
```

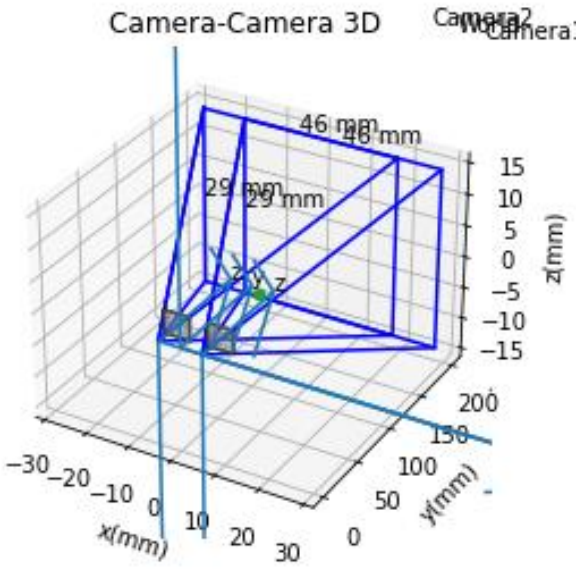
Slika 24. Model konvolucijske neuronske mreže

6. REZULTATI

U ovom poglavlju dana je usporedba rezultata srednje geške (MAE) neuronskih mreža dobivene iz testnih podataka za tri konfiguracije stereo kamere. Za svaku konfiguraciju uzeta su tri skupa podataka, svaki sa drugačijom mjernom i lokalizacijskom greškom (odabrane greške su 0.1 mm, 0.2 mm i 0.5 mm za obje greške). Svi setovi podataka sadrže ukupno 3375 mjerenja, što omogućuje treniranje neuronskih mreža da efikasno pronađu optimalne težine.

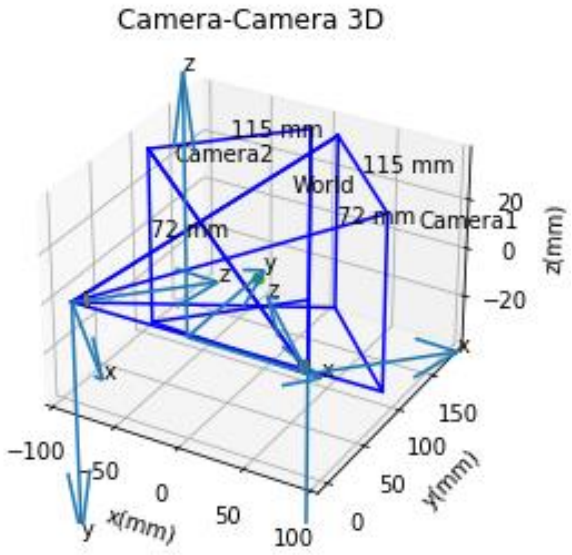
Uzeti su podaci za navedene konfiguracije stereo kamera:

1. Paralelna konfiguracija – ravnine slika nalaze se u istoj ravnini na razmaku od 10 mm po y osi



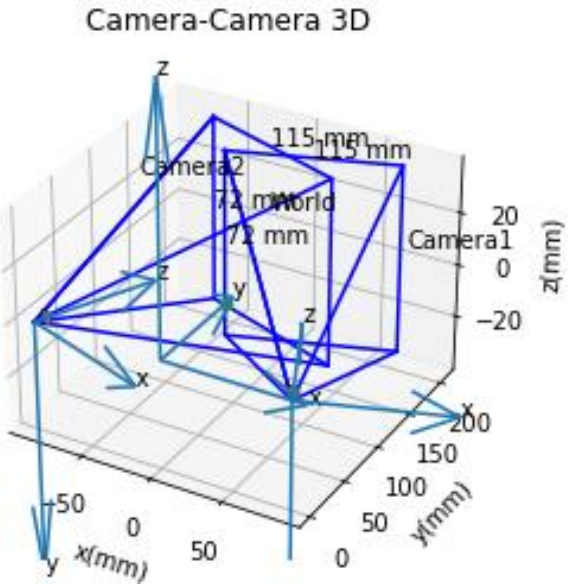
Slika 25. Paralelna konfiguracija kamera

2. Okomita konfiguracija – kut između optičkih osi kamera je 90°, a razmak između središta kamera po y osi iznosi 180 mm.



Slika 26. Okomita konfiguracija kamera

3. Kamere pod kutem od 44° - kut između optičkih osi kamera je 44°, a razmak između središta kamera po y osi iznosi 180 mm.



Slika 27. Konfiguracija s osima kamera pod 45°

6.1. Višeslojni perceptron

MLP mreža pokazuje dobre rezultate. Kod konfiguracije kamere od 0°, MAE iznosi 0.0268 pri mjernoj i lokalizacijskoj pogrešci od 0.1, dok se smanjuje na 0.0187 pri pogrešci od 0.5. Pri kutu od 44°, MLP mreža pokazuje značajno niže vrijednosti MAE, u rasponu od 0.0028 do 0.0041, s minimalnom pogreškom pri mjernoj i lokalizacijskoj grešci od 0.2. Konačno, pri

konfiguraciji kamere od 90°, MLP mreža postiže svoje najniže vrijednosti MAE, s rasponom od 0.0028 do 0.0032, što ukazuje na stabilnost mreže pri svim razinama pogreške.

Tablica 1. Srednja greška testnih podataka MLP-a

Konfiguracija kamera	0°			44°			90°		
Lokalizacijska i mjerna greška	0.1	0.2	0.5	0.1	0.2	0.5	0.1	0.2	0.5
MAE	0.0268	0.0068	0.187	0.0041	0.0028	0.0039	0.0032	0.0028	0.0029

6.2. Višeslojni perceptron treniran Levenberg-Marquardt algoritmom

Za MLP treniran s LM algoritmom, pri konfiguraciji kamere od 0°, MAE doseže 0.0535 za mjernu i lokalizacijsku grešku od 0.1 i ostaje relativno visoka, na 0.0458, pri pogrešci od 0.5. Kod konfiguracije kamere pod kutom od 44°, MAE se značajno smanjuje, s najnižom vrijednosti od 0.00244 pri lokalizacijskoj i mjernoj grešci od 0.2. Pri kutu od 90°, vrijednosti MAE su u rasponu od 0.00295 do 0.0034 pa su pri toj konfiguraciji najstabilniji rezultati s povećanjem lokalizacijske i mjerne greške. Ovi rezultati pokazuju da MLP treniran s LM algoritmom postaje efikasniji povećanjem mjerne i lokalizacijske greške te povećanjem kuta optičkih osi kamera.

Tablica 2. Srednja greška testnih podataka MLP treniran LM

Konfiguracija kamera	0°			44°			90°		
Lokalizacijska i mjerna greška	0.1	0.2	0.5	0.1	0.2	0.5	0.1	0.2	0.5
MAE	0.0535	0.0177	0.0458	0.00420	0.00244	0.00601	0.00295	0.00295	0.0034

6.3. Konvolucijska neuronska mreža

CNN mreža pokazuje najniže vrijednosti MAE u gotovo svim postavkama. Za konfiguraciju kamere od 0°, MAE se smanjuje s povećanjem mjerne i lokalizacijske greške, od 0.0256 pri pogrešci 0.1 do 0.0199 pri pogrešci 0.5. Kod kamere orijentirane pod kutem od 44°, MAE dodatno opada, pri čemu su vrijednosti u rasponu od 0.0049 do 0.0070. Najniže vrijednosti MAE zabilježene su pri konfiguraciji kamere od 90°, gdje je MAE u rasponu od 0.0034 do

0.0053. Ovi rezultati ukazuju na to da CNN mreža najpreciznije predviđa pri većim kutovima kamere, s minimalnim MAE čak i pri višim razinama mjerne i lokalizacijske greške.

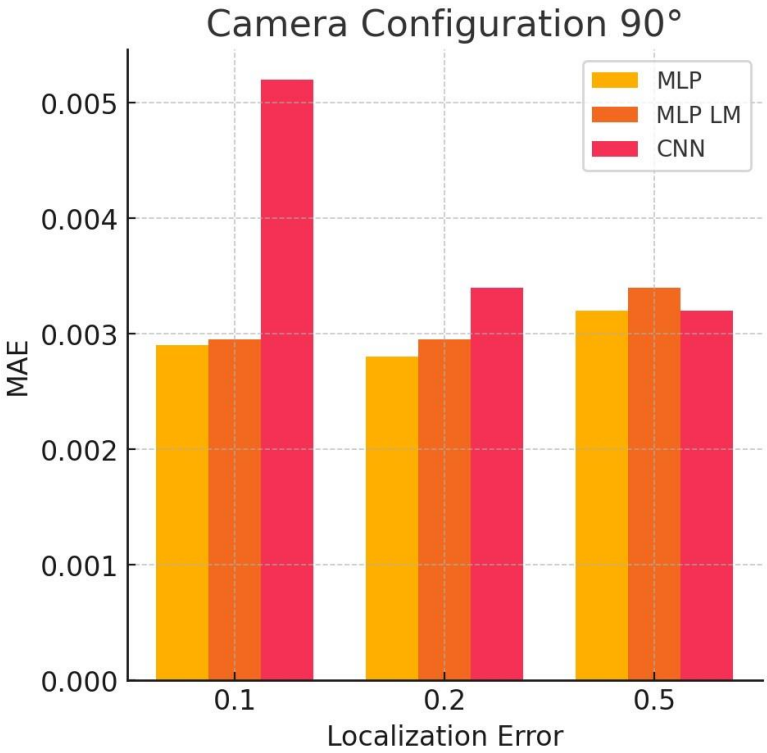
Tablica 3. Srednja greška testnih podataka CNN

Konfiguracija kamera	0°			44°			90°		
Lokalizacijska i mjerna greška	0.1	0.2	0.5	0.1	0.2	0.5	0.1	0.2	0.5
MAE	0.0256	0.0184	0.0199	0.0053	0.0049	0.0070	0.0034	0.0053	0.0052

6.4. Usporedba rezultata neuronskih mreža

Rezultati pokazuju da se MAE za različite konfiguracije neuronskih mreža (MLP, MLP-LM i CNN) značajno razlikuje ovisno o orijentaciji kamere i mjernoj i lokalizacijskoj pogrešci. Zato su rezlutati vizualizirani stupičastim grafovima na Slikama 28.-30. za svaku mjernu i lokalizacijsku grešku i za svaku neuronsku mrežu u ovisnosti o konfiguraciji kamera.

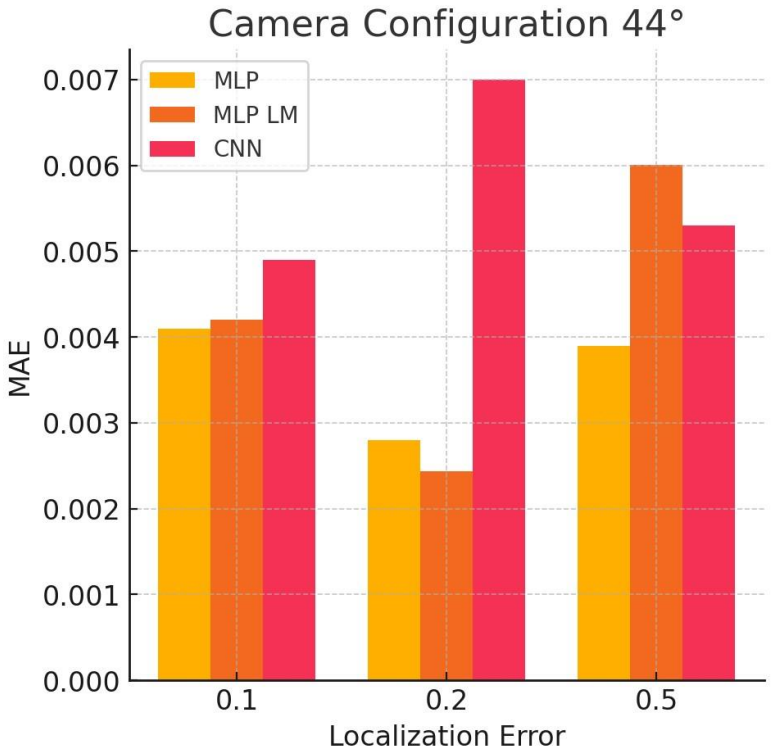
Za konfiguraciju u kojoj su optičke osi kamera pod 90°, sve tri neuronske mreže pokazuju niske vrijednosti MAE, ali s određenim razlikama ovisno o pogreškama mjerenja i lokalizacije. MLP treniran algoritmom LM postiže najnižu MAE za mjernu i lokalizacijsku grešku od 0.1, dok CNN konfiguracija ima najveću MAE. Za pogreške lokalizacije od 0.2 i 0.5, razlike među mrežama su manje izražene, MLP i MLP treniran s LM algoritmom pokazuju slične vrijednosti MAE, dok CNN konfiguracija ima nešto višu MAE. CNN konfiguracija, iako pokazuje veću MAE pri manjoj pogrešci lokalizacije (0.1), zadržava stabilnost pri višim pragovima te je pogodna za zadatke s višim mjernim i lokalizacijskim greškama. (Slika 28.)



Slika 28. Usporedba NN pri okomitoj konfiguraciji kamera

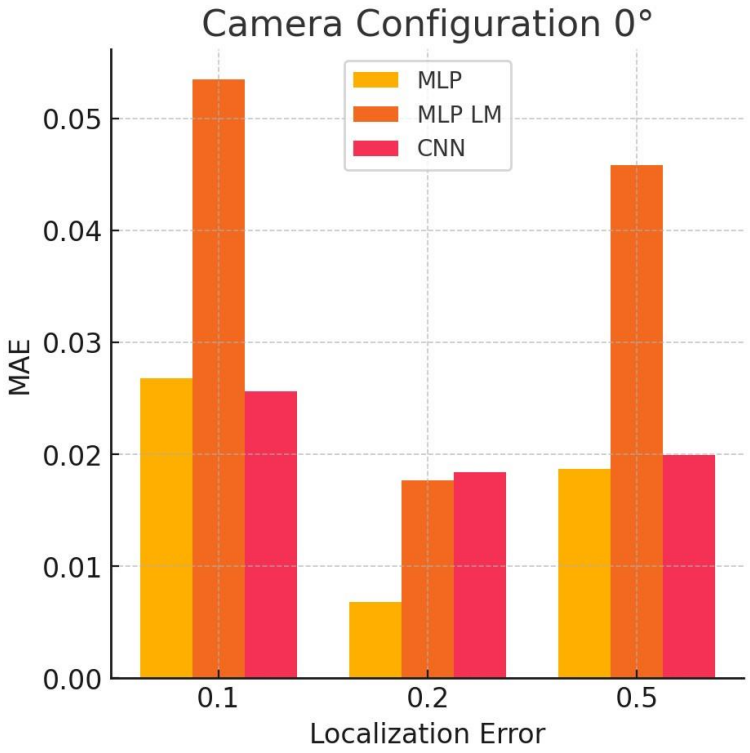
Za konfiguraciju u kojoj su optičke osi kamera pod 44°, sve neuronske mreže pokazuju različite vrijednosti MAE ovisno o pogreškama lokalizacije. Za lokalizacijsku grešku od 0.1, MLP i MLP-LM imaju slične vrijednosti MAE, dok CNN konfiguracija ima nešto višu vrijednost. Pri mjernoj i lokalizacijskoj grešci od 0.2, MLP-LM mreža ima najnižu MAE, dok MLP ima nešto višu vrijednost, a CNN postiže najvišu vrijednost.

Za mjernu i lokalizacijsku grešku od 0.5, MLP-LM i CNN pokazuju nešto veću MAE, dok MLP mreža ima najnižu vrijednost MAE. Ovi rezultati ukazuju na to da pri ovoj kongifuraciji kamera MLP i MLP-LM mreže postižu bolje rezultate pri manjim pogreškama lokalizacije, a MLP mreža se pokazuje najstabilnijom pri različitim razinama lokalizacijskih grešaka. (Slika 29).



Slika 29. Usporedba NN pri konfiguraciji kamera pod 44°

Za konfiguraciju u kojoj su optičke osi kamere pod 0°, sve tri neuronske mreže pokazuju visoke vrijednosti MAE za sve vrijednosti mjerne i lokalizacijske greške (0.1, 0.2, 0.5). Pri mjernoj i lokalizacijskoj grešci 0.1 MLP-LM ima najvišu MAE, dok CNN i MLP pokazuju nižu, ali i dalje relativno visoku MAE. Pri lokalizacijskoj i mjernoj grešci od 0.2 MLP dovoljno precizne rezultate za razliku od MLP-LM i CNN,a pri 0.5, MLP i CNN imaju MAE vrijednosti koje su znatno niže nego za MLP-LM. Ovi rezultati ukazuju na to da je konfiguracija kamere od 0° najmanje učinkovita, s najvišim vrijednostima MAE u odnosu na druge konfiguracije. (Slika 30.)



Slika 30. Usporedba NN pri paralelnoj konfiguraciji kamera

7. ZAKLJUČAK

U ovom radu istražena je primjena različitih neuronskih mreža za ekstrinzičnu kalibraciju stereo kamera i analiza kvalitete njihovih rezultata u odnosu na različite konfiguracije kamera, mjerne i lokalizacijske greške. Rezultati pokazuju da CNN mreža dosljedno postiže najniže vrijednosti srednje apsolutne pogreške u svim konfiguracijama, što ukazuje na njezinu sposobnost učenja iz podataka s visokim razinama lokalizacijske i mjerne greške.

Višeslojni perceptron treniran Levenberg-Marquardt algoritmom i višeslojni perceptron također pokazuju dobre rezultate, posebno pri konfiguracijama s većim kutovima između kamera (44° i 90°), gdje postižu niže vrijednosti MAE.

Iako sve neuronske mreže pokazuju određenu stabilnost i pouzdanost, CNN mreža se pokazala najrobustnijom za različite kutove kamere. Ovi rezultati sugeriraju da je CNN optimalan izbor za primjene gdje je potrebna velika preciznost u uvjetima visokih lokalizacijskih grešaka, dok MLP i MLP treniran LM algoritmom mogu pružiti zadovoljavajuću točnost pri različitim konfiguracijama i manjim greškama.

Ovi rezultati pridonose razumijevanju primjene neuronskih mreža u stvarnim sustavima. Umjesto podataka simuliranih modelom, moguće je napraviti skup podataka mjerenih sustavom stereo kamere koja je upravljana robotskom rukom tako da se pri svakom pomaku spreme podaci koordinata iste prostorne točke na obje kamere.

LITERATURA

[1] Hata K, Savarese S. CS231A Course Notes 1: Camera Models.

[2] Simple Camera Models with NumPy and Matplotlib | by Mario Namtao Shianti Larcher | Analytics Vidhya | Medium https://medium.com/analytics-vidhya/simple-camera-models-with-numpy-and-matplotlib-92281f15f9b2 (28.7.2024.).

[3] https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html (25.8.2024.)

[4] Special Orthogonal Group SO(3), Euler Angles, Angle-axis, Rodriguez Vector and Unit-Quaternion: Overview, Mapping and Challenges

[5] Stereo vision camera | https://www.e-consystems.com/blog/camera/technology/what-is-a-stereo-vision-camera-2/ (20.8.2024.)

[6] Šijak I..: Model prostornog odnosa i interakcije industrijske kamere i linijskog lasera

[7] https://udruga-penkala.hr/stereo-kamera-kao-senzor-u-robotici/2021/ (1.9.2024.)

[8] Lokalizacija objekata primjenom stereoskopske rekonstrukcije, Petra Marče

[9] https://www.cs.auckland.ac.nz/courses/compsci773s1t/lectures/773-GG/lectA-773.htm (2.9.2024.)

[10] Pronalaženje prohodnog tla stereoskopskim racunalnim vidom, Slavko Grahovac

[11] Predavanja Umjetne neuronske mreže, FER, B. D. Bašić, M. Čupić, J. Šnajder

[12] Korištenje WEB tehnologija za izradu i prikaz višeslojnih neuronskih mreža, Vedran Karačić

[13] https://scikit-learn.org/stable/modules/neural_networks_supervised.html (3.9.2024)

[14] https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/ (4.9.2024.)

[15] Camera calibration using neural networks, Márcio Mendonça