Prepoznavanje i segmentacija objekata u slikama korištenjem dubokog učenja

Lulić, Luko

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 338

PREPOZNAVANJE I SEGMENTACIJA OBJEKATA U SLIKAMA KORIŠTENJEM DUBOKOG UČENJA

Luko Lulić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 338

PREPOZNAVANJE I SEGMENTACIJA OBJEKATA U SLIKAMA KORIŠTENJEM DUBOKOG UČENJA

Luko Lulić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 338

Pristupnik:	**Luko Lulić (0036515205)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Damir Seršić

Zadatak: **Prepoznavanje i segmentacija objekata u slikama korištenjem dubokog učenja**

Opis zadatka:

Duboko učenje je vrlo uspješno u prepoznavanju objekata u slikama, osobito ako se učenje odvija na odgovarajućem skupu podataka prethodno obrađenih pomoću naprednih metoda obrade slike. Predprocesiranje se obično koristi za odvajanje korisnih informacija od šuma i smetnji, kao i za izdvajanje značajki od interesa. Učenje i primjena takvih mreža obično se provodi na grafičkim (GPU) ili tenzorskim procesnim jedinicama (TPU). U ovom diplomskom radu potrebno je prikupiti veliki skup podataka odabranih slika objekata i pravilno ih prethodno obraditi. Duboka mreža bit će dizajnirana i osposobljena da ispuni željeni zadatak: pouzdano otkrivanje i praćenje objekata u video sekvenci. Proces učenja može se provoditi na bilo kojoj dostupnoj platformi, kao što je računanje u oblaku, CPU ili GPU. Konačna implementacija temeljit će se na TPU akceleratorskom uređaju.

Rok za predaju rada: 28. lipnja 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 338

Prepoznavanje i segmentacija objekata u slikama korištenjem dubokog učenja

Luko Lulić

Zagreb, rujan 2024.

SADRŽAJ

1. Uvod

U suvremenom maloprodajnom sektoru zbog visoke konkurencije i malih stopa profita velika pozornost obraća se optimizaciji prodaje. Sadržaj polica jedna je od najutjecajnijih komponenti koji utječe na opći dojam i zadovoljstvo kupaca, te na njihove odluke pri odabiru proizvoda koji će staviti u svoju košaricu. Stoga prodavači moraju osigurati da su njihove police dobro popunjene, dok proizvođači provode redovite provjere kako bi osigurali da je njihov proizvod zastupljen na policama sukladno s postignutim ugovorom i predstavljen na zadovoljavajuć način. Tradicionalne metode popisa i verifikacije plasmana proizvoda su radno intenzivne i podložne ljudskim pogreškama, što dovodi do potrebe za automatiziranim rješenjima. Ovaj diplomski rad istražuje razvoj sustava računalnog vida za automatsku detekciju različitih proizvoda na policama trgovina mješovitom robom.

Računalni vid, polje na presjeku umjetne inteligencije i obrade slike, ima za cilj omogućiti strojevima tumačenje i razumijevanje vizualnih informacija iz svijeta. U kontekstu maloprodaje, sustavi računalnog vida mogu revolucionirati upravljanje zalihama, smanjiti troškove rada i poboljšati cjelokupno iskustvo kupnje osiguravajući da su proizvodi pravilno postavljeni i dostupni.

Danas najpopularniji pristup za rješavanje problema otkrivanja objekata na slikama su duboki modeli temeljeni na konvolucijskim neuronskim mrežama. Konvolucijski modeli detektiraju objekte tako da pronalaze uzorke na slikama usporedbom niza naučenih filtera s regijama na slici. Povezivanjem više konvolucijskih slojeva, model dobiva sposobnost prepoznavanja sve kompleksnijih uzoraka više razine, uz dobru otpornost na varijacije u izgledu objekta. Dok takvi modeli ostvaruju vrlo dobre rezultate, dolaze i s nekim nedostatcima. Kako bi trenirali takve modele potrebno je prikupljanje velikih skupova podataka, dok proces treniranja može biti resursno zahtijevan i dugotrajan. Sljedeći izazov postoji kod održavanja modela, kada želimo dodati novu klasu koju model treba moći prepoznati, što je čest slučaj za ovu primjenu kod izlaska novog proizvoda ili promjene pakiranja proizvoda. Tada je potrebno ili ponoviti treniranje zadnjih slojeva modela nad cijelim skupom podataka što utječe na

sve klase ili održavati zasebne klasifikacijske glave za svaku klasu što dovodi do veće kompleksnosti modela, može zahtijevati još veću količinu podataka i usporava vrijeme inferencije modela.

Ideje ovog rada je iskoristiti činjenicu da su sva pakiranja istog proizvoda identična, što omogućava korištenje algoritama za usklađivanje ključnih točki na slikama kako bi detektirali objekt. Ovi algoritmi proučavaju mikro regije slike, pronalaze one najzanimljivije i kreiraju njihove deskriptore. Ti deskriptori za otporni su na promjene u rotaciji, osvjetljenju i dimenziju objekta na slici. Usporedbom deskriptora moguće je upariti iste točke na objektu na dvije slike. Oslanjanjem na opisane sposobnosti povezivanja značajnih točki na pakiranjima proizvoda sa slike s onima u 3D modelima u bazi podataka u ovom radu cilj je ostvariti rezultate koji su ekvivalentni najuspješnijim današnjim metodama. Prednosti takvog sustava su što je za detekciju dovoljan 3D model objekta koji želimo moći detektirati, te ne zahtijeva treniranje.

U idućem poglavlju opisana su tri algoritma za usklađivanje ključnih točki rastuće složenosti korištena u ovom radu: SIFT, Superpoint i OmniGlue. Opisan je jedan od najpopularnijih današnjih modela za detekciju YOLO, koji je u ovom radu korišten za usporedbu rezultata.U trećem poglavlju opisano je postignuto rješenje, te su u zadnjem poglavlju predstavljeni postignuti rezultati.

2. Metode otkrivanje objekata

Otkrivanje objekata je ključni zadatak u području računalnog vida koji se odnosi na identifikaciju i lokalizaciju objekata različitih klasa unutar slike ili videa. Ovo područje ima široku primjenu u raznim industrijama, uključujući autonomna vozila, sigurnosne sustave, medicinsku dijagnostiku, industrijsku automatizaciju i maloprodaju. Otkrivanje objekata uključuje dva glavna zadatka: lokalizaciju, što predstavlja određivanje položaja objekta unutar slike i klasifikaciju objekta u jednu od predefiniranih klasa ili kategorija. Ovi zadaci se često izvode istovremeno, gdje sustav ne samo da prepoznaje što se nalazi na slici, nego i točno određuje gdje se objekt nalazi. Postoje mnoge metode koje se koriste za otkrivanje objekata, u ovom poglavlju ćemo detaljnije proučiti neke od algoritama za poklapanje ključnih točki i konvolucijske neuronske mreže.

2.1. Algoritmi za usklađivanje ključnih točaka

Algoritmi za opisivanje i usklađivanje ključnih točaka koriste se za identifikaciju i povezivanje značajnih točaka na slikama, omogućujući prepoznavanje objekata i scena. Opisivači ključnih točaka, kao što su SIFT, SURF i ORB, stvaraju vektore značajki za svaku ključnu točku, čime omogućuju usporedbu točaka između različitih slika. Algoritmi za usklađivanje, poput RANSAC (Random Sample Consensus), koriste ove opisnike za pouzdano uparivanje točaka, čak i u prisutnosti šuma i iznimaka. Ovi algoritmi su temeljni za mnoge aplikacije, uključujući 3D rekonstrukciju, robotsku navigaciju i prepoznavanje objekata.

2.1.1. Sift

Scale-Invariant Feature Transform (SIFT) (Lowe (2004)) je jedan od najpoznatijih i najšire korištenih algoritama za detekciju i opisivanje ključnih točaka u slikama. Razvijen od strane Davida Lowa 1999. godine, SIFT je postao standard za mnoge aplikacije u računalnom vidu zbog svoje robusnosti i točnosti. Do 2020. godine algoritam

je bio zaštićen patentom. U ovom dijelu, istražit ćemo kako SIFT funkcionira, njegove ključne komponente, prednosti i nedostatke te primjene u stvarnom svijetu.

SIFT algoritam identificira ključne točke koje su invarijantne na promjene skale i rotacije, te djelomično invarijantne na promjene osvjetljenja i perspektive. Proces se sastoji od četiri glavne faze: detekcija ključnih točaka (Keypoint Detection), pronalazak točne lokacije ključnih točaka (Keypoint Localization), pridruživanje orijentacije (Orientation Assignment) i kreiranje deskriptora ključnih točaka (Keypoint Descriptor).

Prvi korak u SIFT algoritmu je identifikacija potencijalnih ključnih točaka u slici. Ovaj korak koristi Difference of Gaussian (DoG) metodu za identifikaciju točaka koje su značajne u različitim skalama. Slika se filtrira Gaussovim zamućenjem pri različitim standardnim devijacijama za generiranje skale prostora, zatim se uspoređuje vrijednost piksela s njegovim susjedstvom i s odgovarajućim susjedstvom prethodnoj i sljedećoj skali. Ako se odredi da je taj piksel ekstrem on se smatra potencijalnom ključnom točkom.

Nakon inicijalne detekcije, točne lokacije ključnih točaka se određuju prilagodba kvadratne funkcije na podatke susjedstva. Ovo omogućava preciznije određivanje položaja ključnih točaka i eliminaciju onih koje su slabo definirane ili su na rubovima.

SIFT algoritam dodjeljuje orijentaciju svakoj ključnoj točki na temelju lokalne gradijentne orijentacije. Ovo omogućava da ključne točke budu invarijantne na rotaciju. Orijentacija se računa pomoću histograma orijentacija gradijenata unutar lokalnog područja oko svake ključne točke.

U zadnjoj fazi, za svaku ključnu točku kreira se opisnik koji sažima lokalne gradijente u susjedstvu ključne točke. Opisnik je obično vektor duljine 128, gdje se gradijenti dijele u 4x4 mrežu i za svaku ćeliju se stvara histogram s 8 orijentacija.

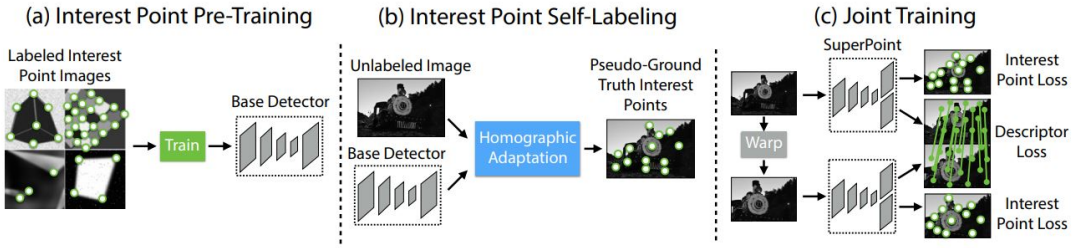
SIFT algoritam ima nekoliko prednosti, uključujući invarijantnost na skalu i rotaciju, robusnost na promjene osvjetljenja, i stvaranje diskriminativnih opisnika za precizno uparivanje točaka između različitih slika. Međutim, SIFT je računski složen i spor, te je bio patentiran što je ograničavalo njegovu slobodnu upotrebu. Primjene SIFT-a uključuju prepoznavanje objekata, spajanje slika u panorame, 3D rekonstrukciju i robotsku navigaciju. Unatoč svojim nedostacima, SIFT ostaje temeljni alat u računalnom vidu zbog svoje robusnosti i učinkovitosti.

2.1.2. Superpoint

SuperPoint je moderan algoritam za detekciju i opisivanje ključnih točaka u slikama, koji koristi pristup temeljen na dubokom učenju. Razvili su ga Daniel DeTone, Tomasz Malisiewicz i Andrew Rabinovich, a predstavljen je u radu "SuperPoint: Self-Supervised Interest Point Detection and Description" 2018. godine.

SuperPoint mreža se sastoji od tri glavne komponente: enkodera, detektora ključnih točaka i deskriptora. Enkoder koristi konvolucijsku neuronsku mrežu piramidalne strukture za ekstrakciju značajki u slici. Dekoder ključnih točaka dodjeljuje svakoj točki u izlaznoj slici vjerojatnost da je ključna točka. Ovaj dekoder koristi pristup bez parametara poznat kao "sub-pixel convolution". Deskriptor računa tenzor veličine H×W×D, gdje je H visina, W širina, a D dubina (broj dimenzija opisnika), koristeći polugustu mrežu opisnika za smanjenje memorijskih zahtjeva tijekom treniranja. Zatim se primjenjuje bikubična interpolacija i L2-normalizacija kako bi se aktivacije imale jediničnu duljinu. Ovi vektori opisnika omogućuju usporedbu i uparivanje ključnih točaka između različitih slika.

Treniranje SuperPoint algoritma započinje treniranjem baznog modela MagicPoint koji je zadužen za kreiranje pseudo istinitih oznaka koje se koriste pri treniranju detektora ključnih točaka i deskriptora. Treniranje MagicPoint odvija se u dva glavna koraka: inicijalno treniranje na sintetiziranim podacima, te fino podešavanje na stvarnim slikama koristeći homografsku adaptaciju. Sintetički skup podataka sastoji se od slika s jednostavnim geometrijskim oblicima poput kvadrata, trokuta i krugova, gdje su ključne točke unaprijed definirane. MagicPoint mreža se trenira na ovom skupu podataka, učeći prepoznati ključne točke na temelju unaprijed definiranih oznaka. Zatim se uz homografske adaptacije koristi MagicPoint na stvarnim slikama kako bi kreirali bogat skup pseudo istinitih točki interesa za treniranje glavnog SuperPoint modela.



Slika 2.1: Shema samonadziranog treniranja u algoritmu Superpoint DeTone et al. (2018)

SuperPoint se trenira korištenjem COCO skupa podataka, primjenjujući zajednički pristup treniranju na parovima slika koje su povezane nasumično generiranim homo-

grafijama. Ovaj proces omogućuje istovremenu optimizaciju detekcije ključnih točaka i računanja opisnika. Za razliku od tradicionalnih sustava koji odvojeno detektiraju ključne točke i računaju opisnike, SuperPoint dijeli većinu parametara mreže između oba zadatka, omogućujući zajedničku računalnu i reprezentacijsku učinkovitost.

Zbog efikasne strukture SuperPoint algoritma moguć je rad u stvarnom vremenu, što ga čini pogodnim za aplikacije poput robotske navigacije i proširene stvarnosti. Self-supervised treniranje i homografska adaptacija čine ga robusnim na različite promjene uvjeta slike, te algoritam postiže gušća i točnija uparivanja ključnih točki od algoritama koji nisu temeljeni na dubokom učenju poput SIFT, LIFT i ORB.

2.1.3. OmniGlue

OmniGlue je napredni algoritam za usklađivanje slika, koji kombinira različite tehnike ekstrakcije i analize značajki kako bi precizno identificirao odgovarajuće ključne točke između dviju slika. Objavljen je 2024. godine od strane Googleovih istraživača Hanwen Jiang, Arjun Karpur, Bingyi Cao, Qixing Huang i Andre Araujo. Glavni cilj modela je postići dobru sposobnost generalizacije u novim domenama za koje ne postoje veliki skupovi podataka. Kako bi ostvario ovaj cilj model je strukturiran kroz četiri glavne faze.

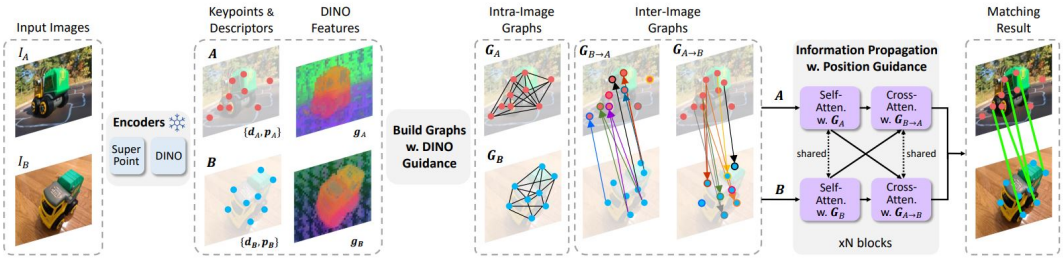
Prva faza OmniGluea uključuje ekstrakciju značajki iz slika korištenjem dvaju komplementarnih enkodera: SuperPoint i DINOv2. SuperPoint je dizajniran za precizno usklađivanje, fokusirajući se na finu razlučivost značajki. On identificira ključne točke i njihove lokalne deskriptore, omogućujući detaljno prepoznavanje strukture slike. S druge strane, DINOv2 je model temeljen na širokom vizualnom znanju koji kodira grube, ali sveobuhvatne informacije o slikama. Ova kombinacija omogućuje OmniGlueu da iskoristi prednosti oba pristupa – detaljnu analizu od SuperPointa i širu vizualnu kontekstualizaciju od DINOv2.

Nakon ekstrakcije značajki, OmniGlue gradi grafove asocijacije ključnih točaka, kako unutar jedne slike (intra-image) tako i između dviju slika (inter-image). Ovi grafovi omogućuju strukturiranje i povezivanje ključnih točaka na temelju njihovih značajki. Inovativno, graf između slika koristi vodstvo DINOv2 modela, koji pruža grubi signal sličnosti između ključnih točaka prepoznatih od strane SuperPointa. Ova metoda omogućuje OmniGlueu da zadrži relevantne veze između ključnih točaka iz različitih slika, istovremeno eliminirajući šum i irelevantne podatke.

Treća faza uključuje propagaciju informacija među ključnim točkama koristeći izgrađene grafove. Ovaj proces koristi slojeve samo-pažnje (self-attention) i među paž-

nje (cross-attention) za komunikaciju unutar jedne slike i između dviju slika. Ključna inovacija ovdje je razdvajanje pozicijskih i vizualnih signala. Za razliku od prethodnih modela koji miješaju ove signale, OmniGlue tretira prostorne informacije i značajke pojavljivanja odvojeno. Ovaj dizajn omogućuje rafiniranje značajki ključnih točaka na temelju njihovog prostornog rasporeda i sličnosti značajki, bez kontaminacije konačnih deskriptora pozicijskim informacijama. To značajno poboljšava sposobnost modela da generalizira na nove i nepoznate slike.

Nakon što su značajke ključnih točaka pročišćene, završna faza OmniGluea uključuje primjenu optimalnih slojeva za usklađivanje kako bi se stvorila mapa između ključnih točaka na dvjema slikama. Ovaj proces koristi napredne algoritme za usklađivanje koji osiguravaju da su ključne točke precizno povezane, omogućujući točno prepoznavanje i analizu sličnosti između slika.



Slika 2.2: Arhitektura modela Omniglue Jiang et al. (2024)

2.2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (CNN) su duboke neuronske mreže posebno dizajnirane za obradu slika i drugih strukturiranih mrežnih podataka. One koriste konvolucijske slojeve koji primjenjuju filtre na ulazne slike kako bi izdvojili značajke poput rubova i tekstura. Nakon konvolucijskih slojeva slijede funkcije aktivacije, kao što je ReLU, koje uvode nelinearnost i omogućuju mreži učenje složenih uzoraka. Slojevi za sažimanje smanjuju prostorne dimenzije karata značajki, smanjujući računalno opterećenje i sprječavajući prenaučenost. Potpuno povezani slojevi na kraju mreže integriraju ove značajke za konačne predikcije, poput klasifikacije slika. CNN-ovi za treniranje koriste velike količine podataka. CNN-ovi su primijenjeni na širok raspon zadataka, uključujući prepoznavanje objekata, analizu videozapisa i medicinsku dijagnostiku, te su postali ključni alat u umjetnoj inteligenciji zahvaljujući svojoj sposobnosti automatskog učenja hijerarhijskih reprezentacija iz sirovih podataka.

2.2.1. YOLO

YOLO (You Only Look Once) Redmon et al. (2015) je popularan model za detekciju objekata u računalnom vidu. Razvijen od strane Josepha Redmona i njegovog tima, YOLO se ističe po svojoj brzini i preciznosti. Za razliku od tradicionalnih metoda koje dijele sliku u više regija i analiziraju ih zasebno, YOLO pristupa problemu detekcije objekata kao jedinstvenom zadatku regresije. To znači da cijelu sliku obrađuje odjednom, predviđajući klase objekata i njihove koordinatne okvire u jednoj iteraciji. Ovaj pristup omogućuje vrlo brzo prepoznavanje objekata, što je idealno za aplikacije u stvarnom vremenu, poput video nadzora i autonomnih vozila.

Ulazna slika se dijeli na mrežu ćelija, pri čemu je svaka ćelija odgovorna za detekciju objekata čije središte pada unutar nje. Svaka ćelija predviđa određen broj okvira s koordinatama središta okvira, širinom, visinom i vrijednošću povjerenja koja izražava koliko model vjeruje da se u tom okviru nalazi objekt. Povjerenje se izračunava kao produkt vjerojatnosti postojanja objekta i indeksa preklapanja (IOU) između predviđenog i stvarnog okvira. Svaka ćelija također predviđa uvjetne vjerojatnosti klasa, a konačna procjena klase za svaki okvir dobiva se množenjem povjerenja i vjerojatnosti klase. Mreža koristi 24 konvolucijska sloja za izdvajanje značajki, a 2 potpuno povezana sloja za predviđanje koordinata i klasa. Ovakav dizajn omogućuje YOLO-u brzu obradu slika u stvarnom vremenu uz očuvanje visoke točnosti, dok model globalno razmatra cijelu sliku i sve objekte u njoj.

Proces treniranja YOLO modela započinje predtretiranjem konvolucijskih slojeva na ImageNet skupu podataka. Nakon toga, arhitektura se prilagođava za detekciju objekata dodavanjem dodatnih konvolucijskih i potpuno povezanih slojeva. Model predviđa koordinate bounding boxova i klase, optimizirajući s pomoću metode najmanjih kvadrata pogreške, uz dodatne mjere za preciznost okvira i povjerenje. Trening se provodi na PASCAL VOC skupu podataka uz tehnike poput dropouta i augmentacije podataka kako bi se spriječilo prenaučavanje.

YOLO predstavlja jedinstveni model za detekciju objekata koji omogućava jednostavnu konstrukciju i direktno treniranje na cijelim slikama. Njegova prednost leži u brzini i sposobnosti generalizacije na nove domene, što ga čini idealnim za primjene koje zahtijevaju brzu i pouzdanu detekciju objekata u stvarnom vremenu.

3. Postignuto rješenje

U ovom radu razvijen je sustav za detekciju objekata u slikama bez potrebe za treniranjem modela, koristeći skup 3D modela namirnica kao osnovu za prepoznavanje. Proces započinje prikupljanjem i obradom skupa 3D modela ciljnih objekata koje želimo detektirati. Na temelju ovih modela kreiran je sustav koji može automatski prepoznavati i lokalizirati objekte u slikama.

Sustav funkcionira tako da za svaki objekt iz baze podataka kreira projekciju 3D modela iz određene perspektive. Zatim se koristeći algoritme za usklađivanje ključnih točaka, projekcija 3D objekta uspoređuje sa slikom na kojoj želimo izvršiti detekciju. U slučaju da sličnost između projekcije i slike ne zadovoljava minimalni prag, sustav nastavlja s analizom iz druge perspektive, čime se omogućava detekcija objekta bez obzira na njegovu rotaciju na slici.

Kada sustav postigne zadovoljavajuću razinu sličnosti, započinje proces pronalaženja idealne perspektive objekta korištenjem metode gradijentnog spusta. Ovaj korak je ključan za optimizaciju detekcije i precizno postavljanje okvira oko detektiranog objekta.

Detekcija i lokalizacija objekata u ovom radu provedena je na dva načina: korištenjem direktnog preslikavanja okvira izračunom homografije na temelju detektiranih ključnih točaka te upotrebom konvolucijske neuronske mreže (CNN) istrenirane na općem skupu podataka.

U prvom pristupu, homografija se izračunava pomoću detektiranih ključnih točaka, što omogućuje precizno preslikavanje okvira objekta na sliku. Ova metoda omogućuje brzu i izravnu lokalizaciju, ali može biti osjetljiva na šum i pogrešna poklapanja ključnih točaka, što može rezultirati netočnom detekcijom.

Drugi pristup koristi CNN, koji je treniran za postavljanje okvira na temelju prepoznatih značajki na slici u kombinaciji s ključnim točkama dobivenim iz algoritama za poklapanje. Prednost ovog pristupa je u tome što CNN može učinkovitije filtrirati šum i zanemariti neispravna poklapanja ključnih točaka, što dovodi do robusnijeg i pouzdanijeg postavljanja okvira, posebno u složenijim scenama s mnogo potencijalnih

smetnji.

Nakon što je razvijeno rješenje uspješno implementirano, rezultati su uspoređeni s rezultatima postignutim korištenjem YOLO modela treniranog na istom skupu podataka. Ova usporedba omogućuje procjenu učinkovitosti i preciznosti razvijenog sustava u odnosu na jedan od najpoznatijih modela za detekciju objekata u računalnom vidu, te daje uvid u potencijalne prednosti i nedostatke korištene metode u različitim kontekstima primjene.

3.1. Prikupljeni skup podataka

Skup podataka korišten u ovom radu sastoji se od 20 trodimenzionalnih modela u .obj formatu, koji predstavljaju različite objekte od interesa za detekciju. Ovi modeli su kreirani pomoću LiDAR senzora ugrađenog u iPhone 15 Pro uređaj, koji omogućava visoku preciznost u prikupljanju dubinskih informacija o objektima.

Za generiranje 3D modela korištena je aplikacija KIRI Engine, koja koristi kombinaciju LiDAR podataka i fotografskih informacija za izradu modela. Proces kreiranja modela započinje snimanjem objekta iz svih kutova, pri čemu aplikacija kontinuirano prikuplja i analizira podatke s LiDAR-a i kamere kako bi rekonstruirala geometriju i teksturu objekta u trodimenzionalnom prostoru.

Iako ovaj pristup omogućava stvaranje detaljnih 3D modela, prisutne su određene neispravnosti koje mogu nastati tijekom procesa skeniranja i modeliranja. Ove neispravnosti mogu uključivati manje deformacije geometrije, neujednačenosti u teksturama ili gubitak detalja u dijelovima modela gdje su podaci bili nedovoljni ili neprecizni. Takvi problemi mogu utjecati na točnost detekcije u kasnijim fazama obrade, što je potrebno uzeti u obzir pri korištenju ovih modela za testiranje i evaluaciju razvijenog sustava. Unatoč tome, modeli generirani ovim procesom pružaju dobru osnovu za eksperimentiranje i razvoj naprednih algoritama za detekciju i lokalizaciju objekata u slikama.

3.2. Otkrivanje objekta usklađivanjem ključnih točaka

3.2.1. Kreiranje projekcije 3D objekta

U ovom radu, za dobivanje projekcija 3D modela korištena je biblioteka Open3D. Open3D je moćna i fleksibilna open-source biblioteka koja omogućuje jednostavnu i

brzu obradu trodimenzionalnih podataka. Ova biblioteka nudi širok spektar funkcionalnosti, uključujući obradu 3D struktura, vizualizaciju, rekonstrukciju scena i usklađivanje površina, kao i podršku za strojno učenje u 3D okruženju putem integracija s PyTorch i TensorFlow bibliotekama. Zahvaljujući svojoj optimiziranoj izvedbi i mogućnosti paralelizacije, Open3D omogućuje učinkovito procesiranje složenih 3D podataka, što je ključno za realizaciju ovakvih projekata.

Proces započinje učitavanjem 3D modela uz pripadajuću teksturu iz OBJ datoteke. Nakon što se model uspješno učita, koristi se vizualizator kako bi se kreirale projekcije modela iz različitih perspektiva. Vizualizator omogućuje rotaciju i postavljanje kamere kako bi se dobile projekcije koje simuliraju različite kutove gledanja.

Ove projekcije koriste se za usporedbu s ciljnim slikama, što je ključni korak u procesu detekcije objekata. Na ovaj način omogućuje se pouzdana identifikacija objekata neovisno o njihovoj orijentaciji na slici, što je od velike važnosti za točnost i robusnost razvijenog sustava.

Slika 3.1: Primjer kreiranih projekcija

3.2.2. Algoritam pronalaženja perspektive 3D objekta gradijentnim spustom

Algoritam pronalaženja perspektive 3D objekta gradijentnim spustom za sliku predanu kroz argument iterativno prolazi kroz sve 3D modele iz skupa podataka, te za svaki prolazi kroz skup preddefiniranih perspektiva, odabranih kako bi korišteni algoritam usklađivanja ključnih točki mogao postići minimalnu uspješnost potrebnu za nastavak gradijentnog spusta neovisno o početnoj rotaciji objekta na slici. U svakoj iteraciji, trenutni položaj kamere koristi se za stvaranje projekcije 3D modela. Ova projekcija predstavlja pogled na 3D objekt iz te specifične perspektive. Zatim se kreirana pro-

jekcija uspoređuje s izvornom slikom s pomoću odabranog algoritma za usklađivanje ključnih točki. Na temelju izračunatih poklapanja izračunava se kvaliteta podudaranja. Kvaliteta se računa na temelju broja pronađenih parova ključnih točki skaliranih s kvalitetom podudaranja. Ako izračunata kvaliteta podudaranja ne zadovoljava minimalni prag prekida se petlja i nastavlja s idućom pozicijom kamere. U slučaju da je za je zadovoljen uvjet za nastavak nastavlja se s procjenom gradijenta.

```
1  def detect_objects(self, img):
2      # Za svaki model
3      for model in self.models:
4          # Za svaku pocetnu poziciju kamere
5          for camera_pose in CAMERA_POSES:
6              current_camera_pose = camera_pose
7              max_iter = 20
8              while True:
9                  max_iter -= 1
10                 if max_iter == 0:
11                     break
12
13                 self.camera_history.append(list(
                       current_camera_pose))
14                 projection = create_projection(model,
                       deg_to_rad(current_camera_pose))
15
16                 # Izracunaj keypointe i podudaranja
17                 match, matches_img = self.match_keypoints(
                       projection, img, verbose=True)
18
19                 # Izracunaj fitness
20                 fitness = self.calculate_match_fitness(match)
21                 self.fitness_history.append(fitness)
22
23                 # Ako je fitness manji od minimalnog, prekini
24                 if fitness < self.min_fitness:
25                     break
26
27                 # Izracunaj gradijent
```

```
28                  grad = self.calculate_gradient_with_history(
                        current_camera_pose, fitness, prim, img)
29                  self.grad_history.append(grad)
30
31                  # Azuriraj poziciju kamere
32                  current_camera_pose = self.update_camera_pose(
                        current_camera_pose, grad, learning_rate
                        =110)
33
34                  # Ako je doslo do detekcije, prekini
35                  if self.detection_threshold_reached(fitness):
36                      self.detected = True
37                      self.plot_fitness()
38                      self.plot_camera_movement()
39                      return self.detected, current_camera_pose,
                            match
40
41      self.plot_fitness()
42      self.plot_camera_movement()
43      return self.detected, current_camera_pose, match
```

Listing 3.1: Metoda detect_objects

Funkcija za izračun gradijenta ima ključnu ulogu u procesu optimizacije položaja kamere kako bi se postigla što bolja sličnost između projekcije 3D objekta i ciljne slike. Ova funkcija koristi informacije o trenutnom položaju kamere, povijesti fitness vrijednosti, te prethodnim gradijentima kako bi izračunala optimalan smjer i veličinu pomaka kamere u sljedećoj iteraciji. Na početku, funkcija generira slučajni pomak kamere unutar 3D prostora. Ovaj pomak dodaje se trenutnom položaju kamere kako bi se dobio novi smjer, iz kojeg će se stvoriti nova projekcija 3D objekta. Zatim se izračunava nova projekcija 3D modela iz smjera određenog novim položajem kamere. Nakon stvaranja nove projekcije, ona se uspoređuje s ciljanom slikom s pomoću algoritma za podudaranje ključnih točaka, što omogućuje izračun nove fitness vrijednosti.

Na temelju razlike između nove i prethodne fitness vrijednosti izračunava se osnovni gradijent, koji pokazuje smjer u kojem bi se kamera trebala pomaknuti kako bi se poboljšala sličnost između projekcije i ciljne slike. Ako postoji povijest prethodnih položaja kamere, funkcija uzima u obzir ranije izračunate gradijente. Zbrajanjem pro-

sječnih vrijednosti prethodnih gradijenata s trenutnim gradijentom dobiva se konačni gradijent, koji je stabilniji i precizniji. Ovaj pristup omogućuje izbjegavanje lokalnih minimuma i poboljšava proces optimizacije. Odabrana je metoda procjene gradijenta na temelju samo jednog slučajnog pomaka jer je stvaranje projekcije i usporedba ključnih točaka računalno najzahtjevniji dio algoritma.

```
1
2  def calculate_gradient_with_history(self, camera_pose, fitness
       , model, img, n=3):
3      # Pomakni kameru u random smjeru
4      cam_shift = np.random.rand(3)
5      direction = camera_pose + cam_shift
6
7      # Kreiraj projekciju 3D modela
8      projection, time_taken = create_projection(model,
           deg_to_rad(direction))
9
10     self.avg_projection_time += time_taken
11     self.num_projections += 1
12
13     # Usporedi projekciju s slikom
14     matches, _ = self.match_keypoints(projection, img)
15
16     # Izracunaj fitness izmedu dviju slika
17     new_fitness = self.calculate_match_fitness(matches)
18
19     # Izracunaj gradijent
20     grad = np.array([0,0,0]).astype(float)
21     grad += cam_shift * (new_fitness - fitness)
22
23     if len(self.camera_history) == 0:
24         return grad
25
26     # Uracunaj inerciju gradijenta
27     grad_moment = np.array([0,0,0]).astype(float)
28     for i in range(1, n):
29         if i >= len(self.grad_history):
```

```
30                 break
31             grad_moment += self.grad_history[-i]
32         grad_moment /= n
33         grad += grad_moment
34         return  np.array(grad)
```

Listing 3.2: Metoda calculate_gradient_with_history

Nakon procjene gradijenta, ažurira se trenutna pozicija kamere. Pomak u perspektivi izračunava se kao produkt gradijenta i stope učenja, pri čemu se osigurava kontrola nad veličinom pomaka kako bi se spriječili preveliki skokovi i osigurao stabilan napredak prema optimalnoj poziciji kamere.

```
1
2  def update_camera_pose(self, camera_pose, grad, learning_rate
       =1, max_delta=20):
3      delta = learning_rate * grad
4      delta = np.clip(delta, -max_delta, max_delta)
5      camera_pose += delta
6      camera_pose = camera_pose % 360
7      return camera_pose
```

Listing 3.3: Metoda update_camera_pose

Ako je u ovoj iteraciji postignut zadovoljavajući fitness, objekt se smatra detektiranim, a funkcija pohranjuje posljednju projekciju i pronađene parove ključnih točaka za daljnju upotrebu u postavljanju okvira oko detektiranog objekta. U slučaju da fitness prag nije dostignut unutar zadanog broja iteracija, petlja se prekida, a algoritam prelazi na sljedeću unaprijed definiranu poziciju kamere.

3.2.3. Postavljanje graničnog okvira nad detektiranim objektom i detekcija više instanci

Nakon izvršenog posljednjeg koraka gdje smo gradijentnim spustom pronašli optimalnu rotaciju 3D objekta iz našeg skupa podataka za najveću sličnost s objektom na

izvornoj slici još uvijek nije završen proces detekcije. Poznato je da određeni objekt postoji na izvornoj slici, ali nije poznato gdje točna lokacija tog objekta, niti je poznato koliko je instanci tog objekta na slici. U tu svrhu razvijena su dva rješenja. Prvo koristi direktan pristup u kojem se okvir objekta postavlja direktnim izračunom homografije između parova ključnih točki projekcije i izvorne slike. Zatim se postavljeni okvir izrezuje sa slike te se ponavlja proces iz prethodnog koraka kako bi provjerili postoji li još instanci tog objekta. Ovaj postupak primjenjiv je kada postoje vrlo kvalitetni parovi ključnih točki jer u suprotnom postoji visoka razina šuma koja dovodi do neprecizno postavljenih okvira oko objekta na slici. Kako bi napravili robusnije rješenje razvijen je drugi pristup pomoću dubokog učenja.

Direktan izračun

Prvi korak u postavljanju okvira oko detektiranog objekta je definiranje okvira na generiranoj projekciji objekta. Budući da projekcija sadrži samo model objekta na bijeloj pozadini, ovaj zadatak postaje jednostavan. Potrebno je identificirati rubne piksele koji nisu bijele boje – krajnje lijevi krajnje gornji i krajnje desni krajnje donji piksel. Ti pikseli definiraju granice objekta, a prema njima se određuju rubne točke okvira koji obuhvaća cijeli model.

Ovaj postupak započinje inicijalizacijom FLANN (Fast Library for Approximate Nearest Neighbors) pronalazača podudaranja. Nakon toga, podudarnici se pronalaze pomoću metode knnMatch, koja uspoređuje deskriptore i za svaku ključnu točku iz izvorne slike pronalazi dva najbolja podudaranja u ciljnoj slici. Rezultati se vraćaju kao parovi, a zatim se primjenjuje omjer testiranja za filtriranje dobrih podudarnika , odabiru se samo oni koji zadovoljavaju uvjet da je udaljenost prvog podudarnika manja od 0.75 puta udaljenost drugog podudarnika. Ovaj proces osigurava da se zadrže samo najpouzdaniji parovi ključnih točaka, koji se kasnije koriste za izračun matrice transformacije, tj. homografije, koja se primjenjuje na prethodno definirani okvir.

Primjenom homografije, okvir koji je postavljen na projekciji prenosi se na ciljnu sliku, no zbog različitih perspektiva taj okvir može biti rotiran ili iskrivljen. Stoga je potrebno dodatno ga prilagoditi kako bi se osigurala točna i precizna detekcija objekta u slici, uz minimalna odstupanja u obliku i poziciji okvira.

Kako bi se detektirale sve instance objekta, već prepoznate instance "maskiraju" se, a proces se ponavlja sve dok se ne pronađu svi objekti unutar zadanih parametara. Na kraju, svi detektirani okviri bit će prikazani na izvornoj slici, čime se osigurava potpuna detekcija svih prisutnih instanci objekta.

Slijedna detekcija jedne po jedne instance objekta na slici pomoću algoritama kao što su SIFT ili ORB moguća je zbog načina na koji ovi algoritmi funkcioniraju u pronalaženju i uparivanju ključnih točaka između slike i projekcije.

SIFT i ORB detektiraju ključne točke koje predstavljaju značajne značajke u slici, poput rubova, kutova, ili karakterističnih tekstura. Kada se ove značajke detektiraju na slici i projekciji, algoritmi uspoređuju te točke kako bi pronašli poklapanja. Ova poklapanja temelje se na sličnosti lokalnih značajki, kao što su orijentacija, ljestvica i tekstura.

Kada postoji više instanci istog objekta na slici, najčešće su sva poklapanja između ključnih točaka slike i projekcije koncentrirani oko jedne od tih instanci. To je zato što poklapanja točaka u SIFT-u ili ORB-u rezultiraju sličnim rasporedom točaka na slici kao i na projekciji. Budući da se svaka instanca objekta na slici pojavljuje s različitim transformacijama (npr. rotacija, translacija), poklapanja za jednu instancu će obično biti dovoljno različita od poklapanja za drugu instancu.

To znači da kada algoritam pronađe prvi set poklapanja, sva poklapanja će gotovo uvijek biti koncentrirani oko jedne specifične instance objekta, ignorirajući ostale instance na slici. Jednom kada se ova instanca detektira i "maskira", algoritam može nastaviti s pretraživanjem ostatka slike za sljedeću instancu, čime se omogućava slijedna detekcija.

Drugim riječima, zbog lokalnog karaktera detekcije značajki i njihovog uspoređivanja, SIFT i ORB će, u svakoj iteraciji, prirodno koncentrirati svoje poklapanje na samo jednu instancu objekta, omogućujući algoritmu da postupno identificira sve prisutne instance na slici.

Konvolucijski model za detekciju

Zbog problema s nekonzistentnošću koji su nastajali uslijed pogrešnog uparivanja ključnih točaka, pristup direktnog izračuna točne lokacije objekta na slici pokazao se nepouzdanim. Da bi se ovaj problem riješio, odlučeno je iskoristiti snagu dubokog učenja. Razvijen je model koji koristi značajke poput rubova i kutova, dobivene konvolucijskom neuronskom mrežom, kao dodatne informacije uz poklapanja ključnih točaka iz prethodnih koraka. Ovaj kombinirani pristup omogućava znatno precizniju detekciju objekata na slici.

Model koristi samonadziran pristup za generiranje skupa podataka koji simulira njegov zadatak koristeći opći skup podataka. Ovaj pristup omogućava modelu da automatski uči i prilagođava se specifičnostima zadatka detekcije, bez potrebe za ruč-

nim označavanjem ili prilagođavanjem podataka. Ključna prednost ovog rješenja je da se zadržava važno svojstvo koje omogućava detekciju novih objekata bez potrebe za ponovnim treniranjem modela. Drugim riječima, jedino što je potrebno za detekciju novog objekta jest posjedovanje njegovog 3D modela. Model je dizajniran da generalizira na različite oblike i strukture objekata, što ga čini vrlo fleksibilnim i prilagodljivim različitim scenarijima detekcije.

Ova inovativna kombinacija klasičnih metoda računalnog vida s naprednim tehnikama dubokog učenja omogućava pouzdaniju i robusniju detekciju, posebno u slučajevima kada dolazi do velikih varijacija u izgledu objekata na slici ili kada postoje problemi s točnošću uparivanja ključnih točaka. Na taj način, model uspješno balansira između potrebe za preciznošću i fleksibilnosti, omogućujući detekciju objekata u stvarnom svijetu uz minimalne zahtjeve za ručnom intervencijom.

DataGenerator je ključni element u pripremi podataka za treniranje modela za detekciju objekata. Proces započinje odabirom slika iz skupa podataka, uz učitavanje pripadajućih anotacija koje precizno opisuju položaj i kategoriju svakog objekta na slici.

Kako bi se modelu omogućilo da nauči prepoznavati objekte u raznim uvjetima, slike se podvrgavaju raznim transformacijama poput rotacije, zamućenja, dodavanja šuma i flipanja. Ove transformacije služe kao simulacija nesavršenosti koje se mogu pojaviti prilikom stvaranja 3D modela ili u stvarnom svijetu, stvarajući složenije uvjete za detekciju.

Transformirane slike se zatim koriste za izrezivanje okvira koji sadrže objekte, a ti okviri se uspoređuju s originalnim slikama pomoću algoritama za poklapanje ključnih točaka. Ovaj postupak simulira projekciju 3D modela iz optimalnog kuta gledanja, koji je pronađen putem gradijentnog spusta. Time se dobivaju ključne točke s ugrađenim šumom, koji odražava stvarne uvjete u primjeni, što dodatno povećava robusnost modela.

Model na ulazu prima ključne točke i originalnu sliku, dok se od njega na izlazu traži precizno određivanje okvira oko svih instanci objekta na slici.

Na kraju, svi pripremljeni podaci se grupiraju u skupove koji se koriste tijekom treniranja modela. Nakon svakog prolaza kroz podatke, redoslijed slika se nasumično mijenja kako bi se spriječilo da model nauči nepravilne obrasce temeljem redoslijeda podataka. Ovaj pristup osigurava dinamičnu i učinkovitu pripremu raznovrsnih podataka, što je ključno za razvoj preciznog i pouzdanog modela za detekciju objekata.

Model za detekciju objekata koristi hibridnu arhitekturu koja kombinira konvolucijsku neuronsku mrežu (CNN) za obradu slike s potpuno povezanim slojevima za

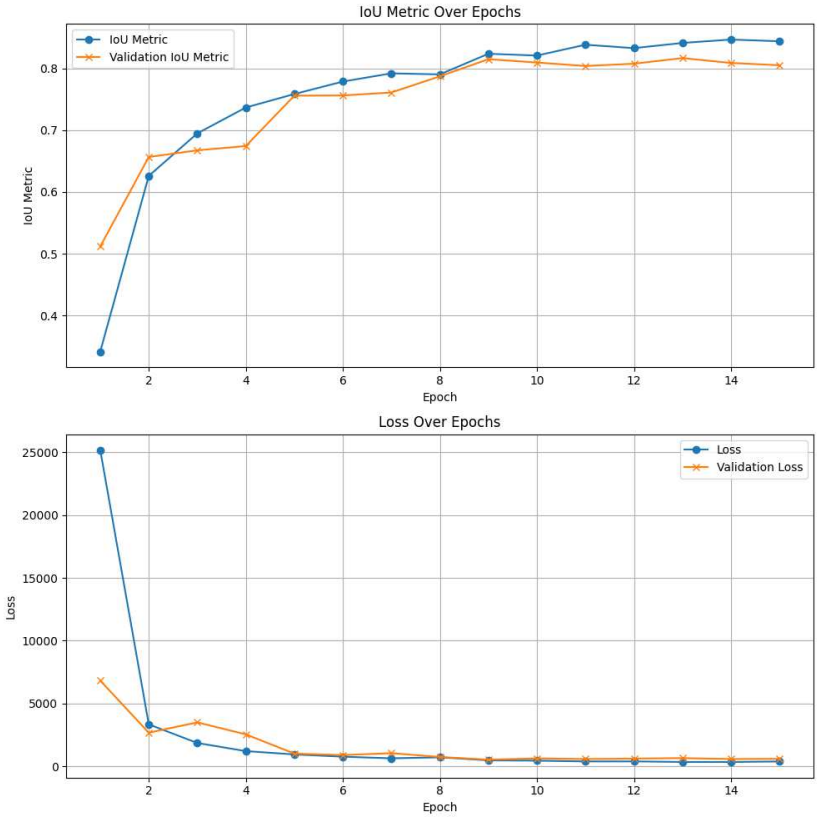
obradu ključnih točaka. Ulazi modela su slika dimenzija (640, 640, 3) i niz ključnih točaka koji sadrži x, y koordinate, odziv, udaljenost podudaranja i kut između točke na projekciji i ciljnoj slici, s maksimalno 128 ključnih točaka. Slika prolazi kroz četiri konvolucijska sloja s ReLU aktivacijom i praćena slojevima za normalizaciju i smanjenje dimenzionalnosti (BatchNormalization i MaxPooling). Ovaj dio modela izdvaja prostorne značajke slike. Paralelno, ključne točke prolaze kroz dva potpuno povezana slojeva kako bi se izlučile značajke specifične za prostorni raspored i dodatne atribute poput kuta i udaljenosti podudaranja. Izlazi iz konvolucijskih i ključnih točaka spajaju se i prolaze kroz dodatna tri potpuno povezana sloja kako bi se integrirale sve informacije. Konačni izlaz modela predviđa rubne okvire (bounding boxes) za maksimalno 8 objekata, sa središnjim koordinatama, širinom i visinom svakog okvira.

```
1
2  def create_detection_model(input_shape=(640, 640, 3),
       max_keypoints=128, max_objects=8):
3      # Ulaz slike
4      image_input = tf.keras.Input(shape=input_shape, name="
           image_input")
5
6      # Ulaz kljucnih tocki
7      keypoints_input = tf.keras.Input(shape=(max_keypoints, 5),
           name="keypoints_input")  # (x, y, response, match
           distance, angle)
8
9      # CNN za ekstrakciju znacajki sa slike
10     x = layers.Conv2D(16, (3, 3), activation='relu')(
           image_input)
11     x = layers.BatchNormalization()(x)
12     x = layers.MaxPooling2D((2, 2))(x)
13     x = layers.Conv2D(32, (3, 3), activation='relu')(x)
14     x = layers.BatchNormalization()(x)
15     x = layers.MaxPooling2D((2, 2))(x)
16     x = layers.Conv2D(64, (3, 3), activation='relu')(x)
17     x = layers.BatchNormalization()(x)
18     x = layers.MaxPooling2D((2, 2))(x)
19     x = layers.Conv2D(256, (3, 3), activation='relu')(x)
20     x = layers.BatchNormalization()(x)
```

```
21      x = layers.MaxPooling2D((2, 2))(x)
22      x = layers.Flatten()(x)
23
24      # Kljucne tocke
25      kp = layers.Dense(64, activation='relu')(keypoints_input)
26      kp = layers.Dense(32, activation='relu')(kp)
27
28      # Povezivanje znacajki slike i kljucnih tocki
29      combined = layers.concatenate([x, layers.
            GlobalAveragePooling1D()(kp)])
30      combined = layers.Dense(64, activation='relu')(combined)
31      combined = layers.Dense(32, activation='relu')(combined)
32      combined = layers.Dense(16, activation='relu')(combined)
33
34      # Izlazni sloj
35      output_bboxes = layers.Dense(max_objects * 4, activation=
            None, name='bboxes_output')(combined)
36      output_bboxes = layers.Reshape((max_objects, 4))(
            output_bboxes)
37
38      # Kreiranje modela
39      model = models.Model(inputs=[image_input, keypoints_input
            ], outputs=[output_bboxes])
40
41      return model
```

Listing 3.4: Konvolucijski model_camera_pose

Zbog sporog rada generatora podataka, model je treniran na unaprijed pripremljenim skupovima podataka. Generirano je ukupno 300 grupa, svaka s 16 slika i pripadajućim ključnim točkama, koristeći prethodno opisani proces. Trening modela proveden je u 15 epoha, uz praćenje ključnih metrike: omjer presjeka i unije (IoU) između predviđenih okvira i stvarnih oznaka na validacijskom skupu, te srednja apsolutna pogreška (MAE) predviđenih okvira. Najbolje performanse model je ostvario u trinaestoj epohi s IoU vrijednošću od 0.8166 i MAE od 18.1063 na validacijskom skupu.



Slika 3.2: Graf treniranja modela

3.3. YOLO v8

Za usporedbu učinkovitosti standardnih metoda dubokog učenja u odnosu na naš prilagođeni pristup, trenirali smo YOLOv8 Jocher et al. (2023) model koristeći isti skup 3D modela. Proces treniranja započeo je generiranjem sintetičkog skupa slika, stvaranjem projekcija iz različitih smjerova i kutova, kako bi model bio izložen raznim mogućim izgledima objekata. Dodatno, slike su podvrgnute raznim obradama, kako bi se simulirali različiti uvjeti iz stvarnog svijeta i time poboljšala generalizacija modela. Nakon faze treniranja, YOLO model je testiran na istom skupu podataka koji smo koristili za testiranje našeg specifičnog algoritma. Također treba istaknuti značajan nedostatak ove metode: dodavanje novih klasa objekata zahtijeva ponovno treniranje cijelog modela. To znači da, za svaku novu klasu koju želimo detektirati, moramo generirati novi skup podataka i ponovno proći kroz cijeli proces treniranja, što je vremenski i računalno zahtjevno.

3.3.1. Obrada podataka

Za svaki 3D model kreiran je opsežan niz projekcija, pri čemu su slike generirane rotiranjem modela horizontalno za 10 stupnjeva i vertikalno za 30 stupnjeva, što je rezultiralo ukupno s 432 slike po modelu. Varirana je pozicija i veličina modela unutar slike. Kako bi se simulirali uvjeti iz stvarnog svijeta, na ove slike dodana je umjetna pozadina s različitim uzorcima šuma. Potom su na generirane slike primijenjene razne augmentacije kako bi se dodatno povećala raznolikost skupa podataka i poboljšala sposobnost modela za generalizaciju. Augmentacije su uključivale rotacije, dodavanje slučajnog šuma, zamućivanje, te varijacije nijanse i saturacije boja, čime su simulirani razni uvjeti osvjetljenja i kvalitete slike.

Dobiveni skup podataka podijeljen je na skup za treniranje i validaciju, čime je osigurano da model ima dovoljno raznolikih primjera za učenje, dok je za testiranje korišten zaseban skup slika iz stvarnog svijeta.

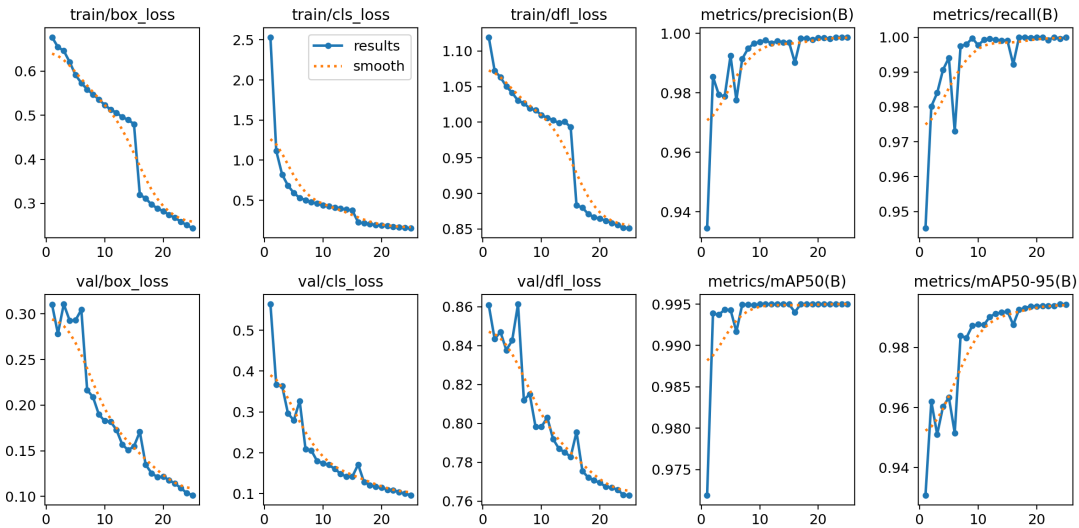
Slika 3.3: Primjer obrađenih projekcija za treniranje YOLO modela

3.3.2. Treniranje modela

Za treniranje modela YOLOv8s korišten je Google Colab, čime je osigurano dovoljno računalne snage za brzu i učinkovitu obuku. Skup podataka pripremljen prema ranije opisanim koracima, uključujući generiranje projekcija 3D modela, dodavanje umjetne pozadine i primjenu augmentacija, prilagođen je za unos u YOLOv8s model. Dimenzije slika su postavljene na 640x640 piksela kako bi se uskladile s očekivanim ulaznim dimenzijama modela.

Treniranje je provedeno u 25 epoha, s veličinom grupe (eng. batch size) od 16, što je omogućilo modelu da se efikasno uči na raznolikom skupu podataka bez preopterećenja memorije dostupne na Colab GPU instancama. Korišten je standardni YOLOv8s model, koji je prilagođen da detektira objekte na slikama generiranim iz 3D modela. Proces treniranja uključivao je automatsku validaciju nakon svake epohe, čime se osigurala kontrola nad performansama modela i spriječilo pretreniranje.

Tokom treniranja, model je optimiziran korištenjem Adam optimizer-a, dok je kao funkcija gubitka korištena kombinacija gubitka koordinate, gubitka razlike između predviđenih i stvarnih klasa, te gubitka za konfidenčne vrijednosti detektiranih objekata. Na kraju procesa, model je evaluiran na testnom skupu slika iz stvarnog svijeta, što je omogućilo procjenu njegove sposobnosti da generalizira naučene obrasce na nove, neviđene primjere.



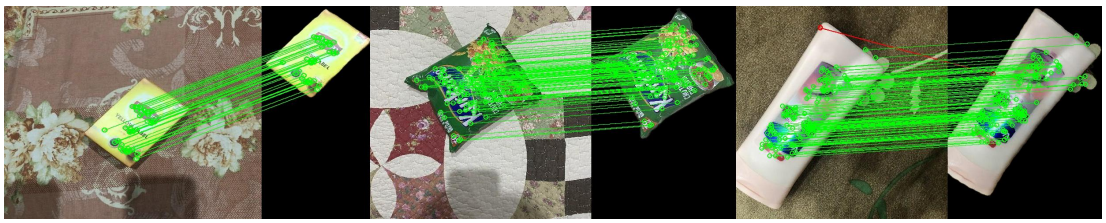
Slika 3.4: Grafovi treniranja YOLO modela

4. Rezultati

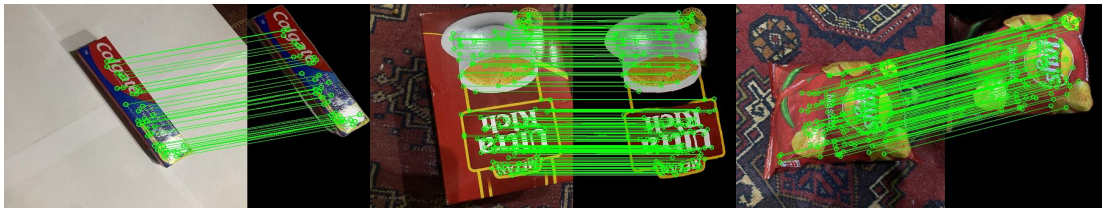
4.1. Usporedba algoritama za usklađivanje ključnih točki

U ovom odjeljku uspoređujemo kvalitetu različitih algoritama za poklapanje ključnih točaka, pri čemu su analizirani SIFT, SuperPoint i Omniglu algoritmi. Ispitivanje je provedeno u simuliranom okruženju koje oponaša zadatke poklapanja ključnih točaka, kao što je opisano u prethodnim dijelovima ovog rada. Algoritmi su testirani na skupu slika namijenjenom evaluaciji u sličnim uvjetima kao tijekom treniranja konvolucijskog modela.

Za testiranje je korišten postupak u kojem su nad slikama primijenjene različite transformacije koje mijenjaju njihov lokalni izgled, čime se otežava zadatak algoritama za poklapanje. Nakon primjene transformacija, objekt sa slike je izrezan, a pozadina uklonjena kako bi se dodatno naglasila relevantnost ključnih točaka na samom objektu. Cilj je bio usporediti transformirani objekt s originalnom slikom i validirati uspješnost poklapanja, analizirajući pokazuju li uparene ključne točke iste lokacije na obje slike, uz dozvoljenu pogrešku do 5 piksela. Na sljedećim slikama prikazana su pronađena podudaranja za algoritmima SIFT, Superpoint i Omniglue. Točna podudaranja prikazana su zelenom bojom, dok su netočna prikazana crvenom bojom.



Slika 4.1: Pronađena podudaranja ključnih točki algoritmom SIFT



Slika 4.2: Pronađena podudaranja ključnih točki algoritmom SuperPoint

Slika 4.3: Pronađena podudaranja ključnih točki algoritmom Omniglue

Primijenjene su različite transformacije, uključujući promjene svjetline, elastične deformacije, zamućenje pokreta te dodavanje šuma. Testiranje je provedeno kroz tri razine intenziteta transformacija kako bi se procijenila otpornost algoritama na povećane izazove. Na taj način dobivena je jasna slika o robusnosti svakog algoritma u uvjetima realnih distorzija.

U sljedećim tablicama prikazani su rezultati mjerenja za tri algoritma: SIFT, SuperPoint i Omniglue. Sva mjerenja su provedena na skupu od 50 slika. Prikazani parametri uključuju prosječan broj poklapanja ključnih točaka, prosječan broj netočnih poklapanja, prosječno vrijeme poklapanja, postotak netočnih poklapanja, te broj slika s manje od 5 točnih poklapanja. Ovi podaci omogućuju usporedbu učinkovitosti i preciznosti algoritama u različitim uvjetima testiranja.

Tablica 4.1: Rezultati mjerenja performansi algoritama za poklapanje ključnih točaka, niska razina distorzija

Algoritam	Prosječan broj poklapanja	Prosječan broj netočnih poklapanja	Postotak netočnih poklapanja (%)	Postotak slika s manje od 5 točnih poklapanja (%)	Prosječno vrijeme poklapanja (s)
SIFT	76	1.55	2.04	2	0.0739
SuperPoint	36.67	0.82	2.25	32	1.2785
Omniglue	29.57	1.33	4.51	0.00	43.50

Tablica 4.2: Rezultati mjerenja performansi algoritama za poklapanje ključnih točaka, srednja razina distorzija

Algoritam	Prosječan broj poklapanja	Prosječan broj netočnih poklapanja	Postotak netočnih poklapanja (%)	Postotak slika s manje od 5 točnih poklapanja (%)	Prosječno vrijeme poklapanja (s)
SIFT	12.55	1.75	13.91	46	0.0654
SuperPoint	7.49	0.53	7.07	70	1.3617
Omniglue	25.52	7.19	28.17	9.05	43.2033

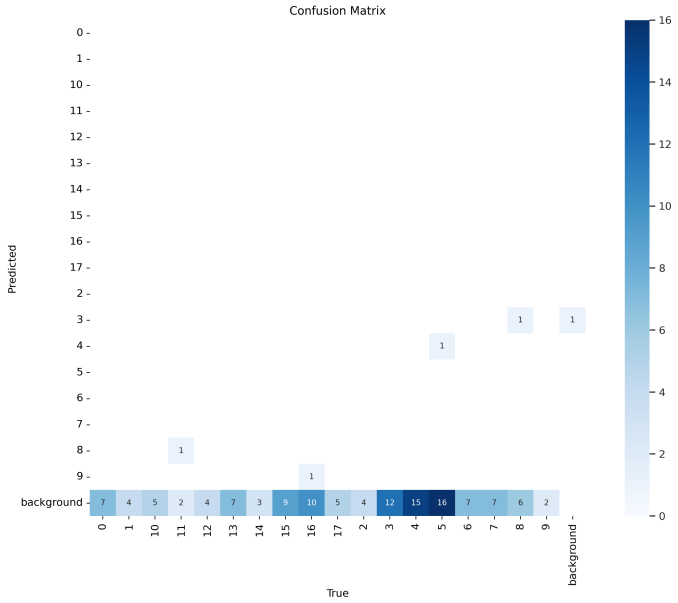
Tablica 4.3: Rezultati mjerenja performansi algoritama za poklapanje ključnih točaka, visoka razina distorzija

Algoritam	Prosječan broj poklapanja	Prosječan broj netočnih poklapanja	Postotak netočnih poklapanja (%)	Postotak slika s manje od 5 točnih poklapanja (%)	Prosječno vrijeme poklapanja (s)
SIFT	4.45	0.96	21.59	70.59	0.0673
SuperPoint	2.06	0.35	17.14	86.27	1.3082
Omniglue	21.43	10.0	46.67	38.10	42.69

4.2. YOLO

U ovom odjeljku analizirana je uspješnost YOLOv8 modela treniranog na sintetičkom skupu podataka kreiranom projekcijama 3D modela. U prethodnom poglavlju prikazano je kako je model postizao iznimno dobre rezultate na validacijskom dijelu sintetičkog skupa, no prilikom testiranja na stvarnim slikama, model nije pokazao sposobnost generalizacije. Na slici 4.3 vidljivo je kako model često pogrešno klasificira objekte ili, čak i kada ispravno prepozna klasu, značajno griješi u pozicioniranju okvira (bounding boxes). Osim toga, na slici 4.4 prikazana je konfuzijska matrica za testni skup. Iz matrice je jasno da su gotovo svi primjeri klasificirani kao pozadina. To se događa jer, čak i kad model ispravno prepozna klasu, zbog loše postavljenih okvira objekti bivaju označeni kao pozadina, što drastično smanjuje točnost modela na stvarnim podacima.

Slika 4.4: Inferencija YOLOv8 modelom na testnom skupu



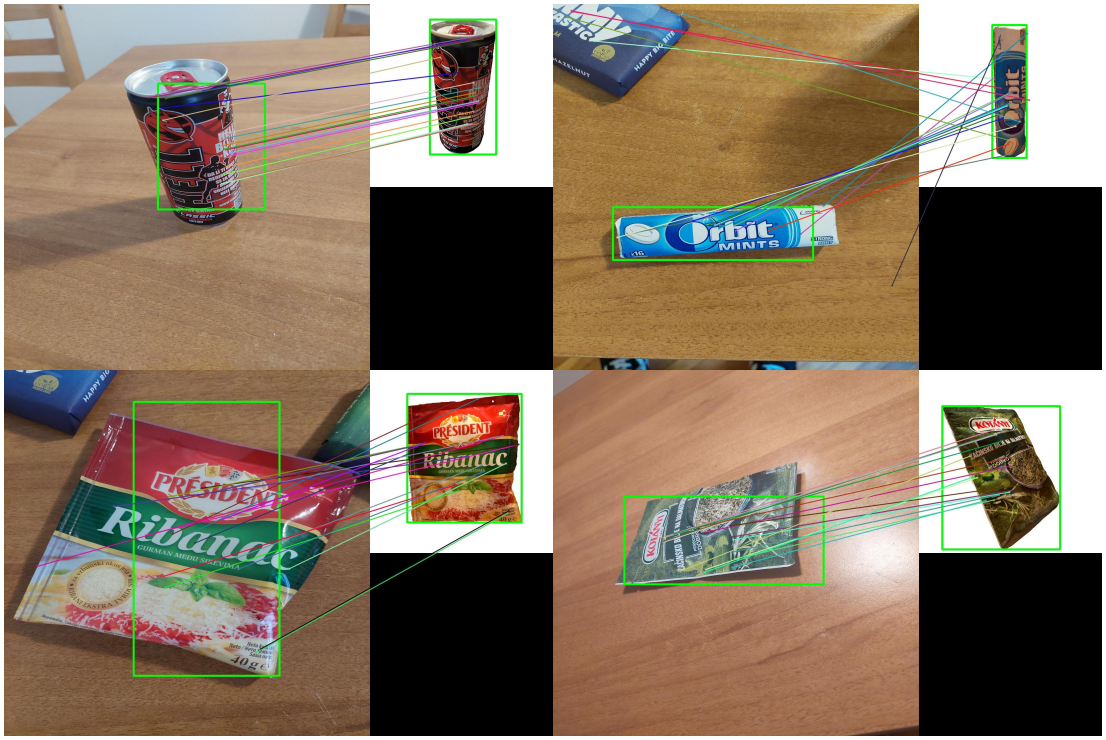
Slika 4.5: Konfuzijska matrica YOLOv8 modela na testnom skupu

4.3. Razvijeno rješenje

4.3.1. Direktan izračun

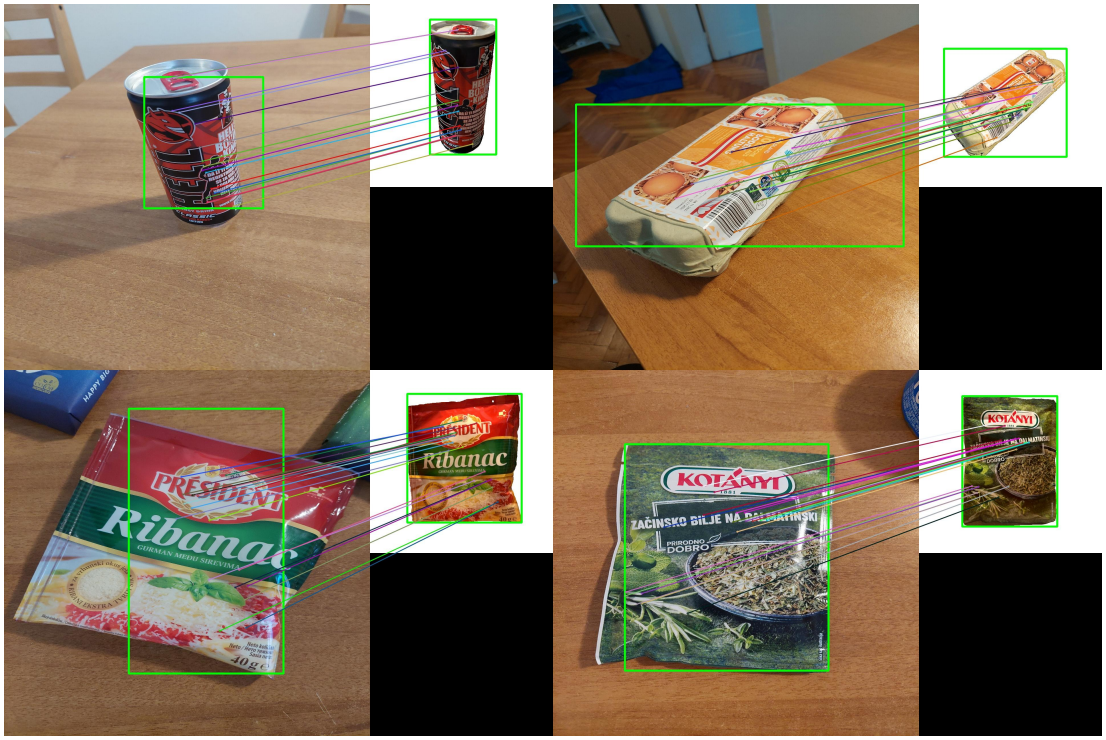
U ovom odjeljku prikazani su rezultati eksperimentalnog postavljanja okvira oko objekata na slikama, gdje se okvir s pronađene projekcije preslikava na ciljnu sliku putem izračuna homografije, koristeći parove ključnih točaka. Metodologija korištenja homografije putem ključnih točaka detaljno je opisana u prethodnom poglavlju, a ovdje su prikazani konkretni rezultati za svaki od testiranih algoritama: SIFT, SuperPoint i OmniGlue. Na slikama su vidljivi primjeri detekcije objekata, te se uspoređuju rezultati u smislu točnosti pozicioniranja okvira za svaki algoritam.

SIFT algoritam je za sve ispitivane slike uspješno pronašao dovoljan broj poklapanja ključnih točaka, omogućujući izračun homografije i preslikavanje okvira. Iako je SIFT pokazao sposobnost pronalaska dovoljno poklapanja, glavni izazov bio je razlikovanje ispravnih od neispravnih podudaranja, što je otežavalo točnu detekciju. Unatoč tome, postavljeni okviri u većini slučajeva približno odgovaraju stvarnoj lokaciji objekta, iako ne uvijek s visokom preciznošću.



Slika 4.6: Primjeri postavljenih okvira uz korištenje SIFT algoritma

SuperPoint algoritam se pokazao manje uspješnim, budući da za čak 14 od 20 testiranih primjera nije uspio pronaći dovoljan broj poklapanja ključnih točaka za izračun homografije. Kod onih primjera gdje je algoritam uspio detektirati dovoljan broj poklapanja, rezultati su bili slični onima koje je pružio SIFT. Međutim, ovaj manjak uspješnih podudaranja značajno ograničava praktičnost SuperPoint algoritma u zadatku postavljanja okvira.



Slika 4.7: Primjeri postavljenih okvira uz korištenje Superpoint algoritma

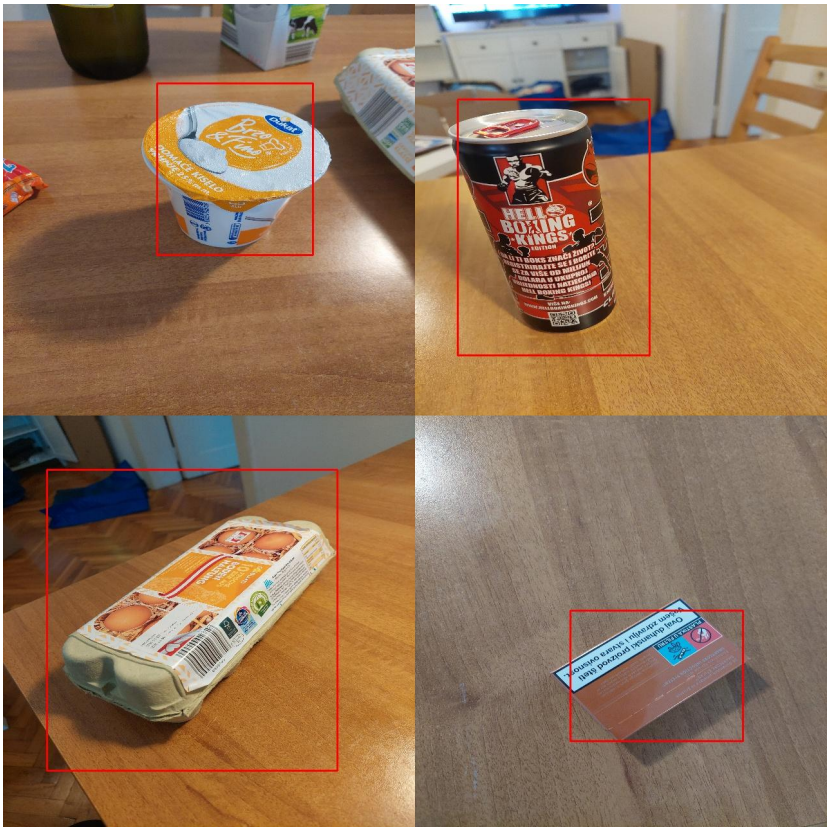
OmniGlue algoritam postigao je najbolje rezultate među testiranim algoritmima. Vizualnom inspekcijom jasno se vidi da su ključne točke bolje raspoređene i da postavljeni okviri preciznije prate stvarne oblike i pozicije objekata. Ipak, i kod OmniGlue algoritma primijećen je značajan broj netočnih uparivanja ključnih točaka, što može dovesti do pogrešnih okvira u određenim slučajevima. Unatoč tome, OmniGlue se pokazao najučinkovitijim među ispitivanim algoritmima, pružajući najsigurnije i najtočnije rezultate detekcije.

Slika 4.8: Primjeri postavljenih okvira uz korištenje Omniglue algoritma

Zaključno, iako svi algoritmi pokazuju određene nedostatke, posebno u vezi s razlikovanjem ispravnih od neispravnih podudaranja, OmniGlue je pružio najsolidnije rezultate. SuperPoint se suočava s ograničenjima u pronalasku dovoljnog broja ključnih točaka, dok SIFT pokazuje veću robusnost, ali ima problema s preciznošću detekcije. Daljnje unapređenje bilo kojeg od ovih algoritama moglo bi uključivati bolju filtraciju netočnih podudaranja kako bi se poboljšala točnost postavljenih okvira.

4.3.2. Konvolucijski model za detekciju

U ovom odjeljku prikazani su rezultati treniranog modela za detekciju, koji koristi konvolucijske slojeve za prepoznavanje značajki u kombinaciji s podudaranjima ključnih točaka. Prikazani su rezultati temeljeni na SIFT algoritmu za podudaranje, koji su se pokazali kvalitetnijima u usporedbi s metodom direktnog izračuna. Iako postoji određeni stupanj pogreške, postavljeni okviri su u prosjeku precizniji od onih dobivenih izravnim računanjem homografije. Osim toga, model uspješno detektira objekte u svim slučajevima, bez potpunih promašaja. Rezultati na stvarnom skupu podataka, koji sadrži do tada neviđene kategorije objekata, konzistentni su s rezultatima na validacijskom skupu, što ukazuje na dobru sposobnost generalizacije modela. To sugerira da bi implementacija složenijeg modela mogla donijeti još bolje rezultate na zahtjevnijim problemima.



Slika 4.9: Rezultati konvolucijskog modela za predikciju

5. Zaključak

U ovom radu predstavljeno je rješenje za detekciju proizvoda na slikama, na temelju algoritama za podudaranje ključnih točaka, koji služe kao osnova za lociranje objekata. Glavni cilj rada bio je razvoj sustava za detekciju objekata bez potrebe za prikupljanjem velikih skupova fotografija svake prepoznate klase. Umjesto toga, sustav se oslanja na 3D modele kao primarni izvor podataka, što omogućuje jednostavno dodavanje novih klasa u model. Za ovu svrhu razvijen je algoritam pronalaska optimalnog gledišta 3D objekta koristeći metodu gradijentnog spusta.

Tijekom istraživanja ispitana su tri algoritma različite složenosti: SIFT, SuperPoint i OmniGlue, te dvije metode za određivanje egzaktne lokacije objekata na slikama, bazirane na ključnim točkama. Prva metoda koristi izravni izračun homografije između projekcije 3D modela i ciljne slike pomoću pronađenih ključnih točaka. Druga metoda, temeljena na samonadziranom učenju, koristi duboku konvolucijsku neuronsku mrežu za ekstrakciju značajki, koje se potom kombiniraju s ključnim točkama radi preciznijeg lociranja objekata.

Rezultati ovog rada su mješoviti. Sustav uspješno detektira jednostavnije objekte, dok svi ispitani algoritmi za podudaranje ključnih točaka pokazuju poteškoće u dosljednom pronalasku ispravnih parova ključnih točaka za složenije zadatke. Jedan od mogućih razloga za te poteškoće su nepravilnosti u procesu izrade 3D modela. Iako modeli vizualno nalikuju stvarnim objektima, postoji visoka razina lokalnih odstupanja koja negativno utječu na deskriptore ključnih točaka, što rezultira slabijim podudaranjima. Zbog toga bi u budućem radu vrijedilo razmotriti korištenje stvarnih fotografija objekata iz različitih kutova, svakako prikupljenih tijekom izrade 3D modela, kao izvora podataka.

Uz to, u radu je također ispitana izvedba trenutno popularnog YOLOv8 modela za detekciju objekata, koji je treniran isključivo na sintetičkom skupu podataka generiranom iz 3D modela. Ovaj model nije pokazao sposobnost generalizacije na stvarnim slikama. Istraživanje u ovom radu otvara mogućnosti za daljnja unaprjeđenja, uključujući poboljšanje kvalitete 3D modela i korištenje sofisticiranijih modela dubokog

učenja poput prilagodbe jednog od popularnih modela za detekciju prikazom pronađenih ključnih točki kao dodatnih kanala na slici.

LITERATURA

Daniel DeTone, Tomasz Malisiewicz, i Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2018. URL https://arxiv.org/abs/1712.07629.

Hanwen Jiang, Arjun Karpur, Bingyi Cao, Qixing Huang, i Andre Araujo. Omni-glue: Generalizable feature matching with foundation model guidance, 2024. URL https://arxiv.org/abs/2405.12979.

Glenn Jocher, Ayush Chaurasia, i Jing Qiu. Ultralytics YOLO, Siječanj 2023. URL https://github.com/ultralytics/ultralytics.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, i Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, stranice 779–788, 2015. URL https://api.semanticscholar.org/CorpusID:206594738.

Prepoznavanje i segmentacija objekata u slikama korištenjem dubokog učenja

Sažetak

U suvremenom maloprodajnom sektoru, visoka konkurencija potiču optimizaciju prodaje, pri čemu sadržaj polica igra ključnu ulogu u zadovoljstvu kupaca. Tradicionalne metode popisa proizvoda su radno intenzivne i sklone pogreškama, pa se javlja potreba za automatiziranim rješenjima. Ovaj rad istražuje razvoj sustava računalnog vida za automatsku detekciju proizvoda, koristeći algoritme za usklađivanje značajki na slikama i 3D modele objekata. Uspoređuje se učinkovitost ovih metoda s konvolucijskim neuronskim mrežama, poput YOLO modela, u kontekstu maloprodaje.

Ključne riječi: računalni vid, otkrivanje objekata, podudaranje značajki, yolo, konvolucijske neuronske mreže

Recognition and segmentation of objects in images using deep learning

Abstract

In the modern retail sector, high competition drives the optimization of sales, where shelf content plays a crucial role in customer satisfaction. Traditional methods of product inventory are labor-intensive and prone to errors, creating a need for automated solutions. This paper explores the development of a computer vision system for automatic product detection, utilizing feature matching algorithms and 3D object models. The effectiveness of these methods is compared with convolutional neural networks, such as the YOLO model, in the retail context.

Keywords: computer vision, object detection, feature matching, yolo, convolutional neural netowrks