Procjena dubine prostora iz 2D slika korištenjem konvolucijske neuronske mreže

Zeba, Ilija

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Ilija Zeba

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing. Ilija Zeba

Zagreb, 2023.

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

Zahvaljujem se mentoru dr. sc. Tomislavu Stipančiću na susretljivosti i pomoći prilikom izrade ovog rada

Ilija Zeba

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 / 23 – 6 / 1**	
Ur.broj: **15 - 1703 - 23 -**	

ZAVRŠNI ZADATAK

Student: **Ilija Zeba** JMBAG: **0035223573**

Naslov rada na hrvatskom jeziku: **Procjena dubine prostora iz 2D slika korištenjem konvolucijske neuronske mreže**

Naslov rada na engleskom jeziku: **Estimation of space depth from 2D images using a convolutional neural network**

Opis zadatka:

Vizijski sustavi s dvije kamere koriste algoritme za određivanje dubine prostora pomoću različitih metoda (npr. temeljenih na epipolarnoj geometriji). Procjena dubine presudan je korak prema zaključivanju geometrije scene iz 2D slika. Cilj monokularne procjene dubine je predvidjeti vrijednost dubine svakog piksela, uzimajući samo jednu RGB sliku kao ulaz.

U radu je potrebno:

- proučiti problematiku i metode procjena dubine prostora temeljem 2D ulaza,
- upoznati se s dostupnim bazama slika te odabrati i prilagoditi jednu koja će biti korištena u fazi izrade modela neuronske mreže,
- primijeniti odgovarajuće metode te razviti neuronsku mrežu za procjenu dubine prostora iz 2D ulaza.

Model je potrebno eksperimentalno evaluirati koristeći vizijski sustav u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava. U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:
30. 11. 2022.

Zadatak zadao:
Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
1. rok: 20. 2. 2023.
2. rok (izvanredni): 10. 7. 2023.
3. rok: 18. 9. 2023.

Predvideni datumi obrane:
1. rok: 27. 2. – 3. 3. 2023.
2. rok (izvanredni): 14. 7. 2023.
3. rok: 25. 9. – 29. 9. 2023.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

Oznaka	**Opis**
UI	Umjetna inteligencija
CNN	Konvolucijska neuronska mreža
3D	Trodimenzionalno
2D	Dvodimenzionalno
RGB	Red, green, blue (sustav boja)
BGR	Blue, green red (sustav boja)
RNN	Rekurentna neuronska mreža
GPU	Grafička jedinica
CPU	Procesorska jedinica
DL	Duboku učenje
DAG	Usmjereni aciklički graf

SAŽETAK

U radu se koristi konvolucijska neuronska mreža kako bi se procijenila dubina prostora iz 2D slike. Mreža prima 2D sliku kao ulaz i izlazi s odgovarajućom kartom dubine koja predstavlja procijenjenu dubinu za svaki piksel u slici. Da bi se mreža naučila predviđati dubinu, koristi se velika skupina podataka 2D slika i njihova odgovarajuća istinita karta dubine. Mreža uči predviđati dubinu smanjujući razliku između procijenjenih karata dubine i istinitih karata dubine.

Ključne riječi: umjetna inteligencija, računalni vid, programiranje

SUMMARY

In this paper convolutional neural network is used to estimate the depth of space from a 2D image as input and outputs a corresponding depth map that represents the estimated depth from each pixel in the image. To learn the network to predict depth, a large dataset od 2D images and their corresponding true depth map is used. The network can predict depth by reducing the difference between the estimated depth maps and the true depth maps.

Key words: artificial intelligence, computer vision, programming

1. UVOD

Umjetna inteligencija(UI) i tehnike dubokog učenja postaju sve popularnije u području računalne vizije, što uključuje procjenu dubine prostora. Osim toga, s napretkom tehnologije, postoji sve više aplikacija koje zahtijevaju točne informacije o udaljenosti objekata u prostoru od promatrača.

Postoji nekoliko različitih metoda za procjenu dubine iz 2D slike. Nekoliko primjera su:

- Stereo vid: ova metoda koristi par slika snimljenih iz različitih kutova kako bi se rekonstruirao 3D model scene. Dubina se procjenjuje analizom razlike u poziciji objekata na svakoj slici.
- Metode bazirane na strukturi scene: ove metode koriste geometrijske karakteristike scene, poput teksture i oblika objekata, kako bi se procijenila dubina. Na primjer, veličina i oblik sjene na slici može pružiti informacije o dubini.
- Metode bazirane na jednoj slici: ove metode koriste samo jednu sliku kako bi procijenile dubinu. Primjerice, dubina se može procijeniti na temelju toga koliko je objekt nejasan i koliko se brzo mijenja u odnosu na pozadinu.
- Konvolucijske neuronske mreže: ovo je vrlo popularna metoda koja koristi duboke neuronske mreže za naučiti mapiranje između 2D slika i dubinske mape. Mreža se obično trenira s velikim skupom podataka koji sadrže slike i odgovarajuće dubinske mape.

Konvolucijska neuronska mreža(CNN) predstavlja jedan od najučinkovitijih pristupa za rješavanje ovog problema. Ova vrsta dubokog modela učenja koristi se za učenje različitih reprezentacija slika, što omogućava preciznije predviđanje dubine.

Za implementaciju CNN se koristi OpenCV biblioteka. Predviđamo dubinu prostora tako da koristimo informacije koje naš CNN model stječe iz slike i koristimo ih za izračunavanje udaljenosti objekta u prostoru od promatrača.

Procjena dubine ima širok raspon primjena, uključujući navigaciju robota, razumijevanje prostora i 3D rekonstrukciju. CNN metoda može se koristiti u ovim i mnogim drugim područjima gdje je točna informacija o dubini kritična. S sposobnošću procjene dubine iz jedne 2D slike, CNN metoda ima potencijal korištenja u različitim stvarnim scenarijima, kao što su autonomna vozila, proširena stvarnost i virtualna stvarnost.

2. TEORIJSKA PODLOGA

Računalni model temelji se na određenim teorijskim podlogama. Da bi se uopće razvio, a onda i razumio, programski kod mora imati određenu strukturu. Kao glavni temelj ovdje se vidi umjetna inteligencija koja kao svoja područja ima računalni vid i strojno učenje. Ta znanja se primjenjuju u ovom radu.

2.1. Umjetna inteligencija

Umjetna inteligencija(UI) je dio računalne znanosti(informatike) koji se bavi razvojem računala da obavljaju zadaće za koje je potreban neki oblik inteligencije, tj. da se mogu snalaziti u novim prilikama, učiti nove koncepte, donositi zaključke, razumjeti prirodni jezik, raspoznavati prizore i dr. Naziv se također rabi za označivanje svojstva svakoga neživog sustava koji pokazuje inteligenciju, obično su to računalni sustavi, dok se izraz katkad neutemeljeno primjenjuje na robote, koji nisu nužno inteligentni. Inteligentnim sustavom smatra se svaki sustav koji pokazuje prilagodljivo ponašanje, uči na temelju iskustva, koristi velike količine znanja, pokazuje svojstva svjesnosti. Funkcije inteligentnog sustava jesu: prikupljanje i obrada informacija, interakcija s radnom okolinom, komunikacija s čovjekom ili s drugim inteligentim sustavima, prikupljanje i obrada znanja, zaključivanje, te planiranje. Dok se pod ljudskom inteligencijom smatraju čovjekove mogućnosti da istodobno pokazuje različite inteligentne odluke i obavlja takve funkcije, današnji su inteligentni sustavi ponajprije specijalizirani za pojedinu mogućnost.

Umjetna inteligencija naslijedila je mnoge zamisli, pristupe i tehnike iz drugih disciplina, a napose onih koje se bave istraživanjem načina ljudskog mišljenja: kognitivne znanosti, logike, psihologije, biologije, filozofije, lingvistike, matematike i dr. Neka od glavnih područja i primjena umjetne inteligencije jesu: računalne igre i simulacije, ekspertni sustavi, neuronske mreže, razumijevanje i obrada prirodnih jezika, računalni vid(raspoznavanje uzoraka ili predmeta, analiza scene), rješavanje problema, pretraživanje podataka, automatsko programiranje i dr.

Prema stupnju inteligencije, umjetna inteligencija se dijeli na tzv. *jaku* i *slabu*. *Jaka umjetna inteligencija* u tolikoj je mjeri razvijena da može razmišljati na istoj razini kao i čovjek.

Slaba umjetna inteligencija je pak ona kojoj se mogu pripisati određen, mali broj inteligentnih svojstva, npr. mogućnost prepoznavanja govora. [1]

2.2. Računalni vid

Računalni vid je područje umjetne inteligencije (UI) koje omogućuje računalima i sustavima da izvuku značajne informacije iz digitalnih slika, videa i drugih vizualnih inputa i poduzmu radnje ili daju preporuke na temelju tih informacija. Ako UI omogućuje računalima da razmišljaju, računalni vid im omogućuje da vide, promatraju i razumiju.

Računalni vid funkcionira gotovo isto kao i ljudski vid, osim što ljudi imaju prednost. Ljudski vid ima prednost doživotnog konteksta za treniranje kako razlikovati objekte, koliko su udaljeni, kreću li se i postoji li nešto pogrešno na slici.

Računalni vid trenira strojeve za obavljanje ovih funkcija, ali to mora učiniti u mnogo kraćem vremenu s kamerama, podacima i algoritmima umjesto mrežnice, optičkih živaca i vidnog korteksa. Budući da sustav osposobljen za pregled proizvoda ili promatranje proizvodnog sredstva može analizirati tisuće proizvoda ili procesa u minuti, uočavajući neprimjetne nedostatke ili probleme, može brzo nadmašiti ljudske sposobnosti.

Računalni vid koristi se u industrijama koje se kreću od energetike i komunalnih usluga do proizvodnje i automobilske industrije.

Računalni vid treba mnogo podataka. Iznova i iznova provodi analize podataka dok ne uoči razlike i konačno prepozna slike. Na primjer, da bi se računalo osposobilo za prepoznavanje automobilskih guma, potrebno mu je unijeti ogromne količine slika guma i predmeta povezanih s gumama kako bi naučilo razlike i prepoznalo gumu, posebno onu bez grešaka

Da bi se to postiglo koriste se dvije osnovne tehnologije: vrsta strojnog učenja koja se naziva duboko učenje i konvolucijska neuronska mreža (CNN).

Strojno učenje koristi algoritamske modele koji omogućuju računalu da uči o kontekstu vizualnih podataka. Ako se kroz model unese dovoljno podataka, računalo će "pogledati" podatke i naučiti razlikovati jednu sliku od druge. Algoritmi omogućuju stroju da uči sam, umjesto da ga netko programira da prepozna sliku.

CNN pomaže "gledati" modelu strojnog učenja ili dubokog učenja razlažući slike na piksele kojima se daju oznake ili oznake. Koristi oznake za izvođenje konvolucija (matematička operacija na dvjema funkcijama za proizvodnju treće funkcije) i daje predviđanja o tome što "vidi". Neuronska mreža pokreće konvolucije i provjerava točnost svojih predviđanja u nizu ponavljanja sve dok se predviđanja ne počnu ostvarivati. To je onda prepoznavanje ili gledanje slika na način sličan ljudima.

Slično kao što čovjek stvara sliku na daljinu, CNN prvo razaznaje oštre rubove i jednostavne oblike, a zatim popunjava informacije dok izvodi ponavljanja svojih predviđanja. CNN se

koristi za razumijevanje pojedinačnih slika. Rekurentna neuronska mreža (RNN) koristi se na sličan način za video aplikacije kako bi pomogla računalima razumjeti kako su slike u nizu okvira međusobno povezane.[2]

2.3. Strojno učenje

Strojno učenje grana je umjetne inteligencije (UI) i računalne znanosti koja se fokusira na upotrebu podataka i algoritama za oponašanje načina na koji ljudi uče, postupno poboljšavajući njegovu točnost.

Strojno učenje važna je komponenta rastućeg područja znanosti o podacima. Korištenjem statističkih metoda, algoritmi se obučavaju za izradu klasifikacija ili predviđanja te za otkrivanje ključnih uvida u projektima rudarenja podataka. Ovi uvidi potom pokreću donošenje odluka unutar aplikacija i poslovanja, idealno utječući na ključne pokazatelje rasta. Kako se veliki podaci nastavljaju širiti i rasti, potražnja na tržištu za podatkovnim znanstvenicima će se povećavati. Od njih će se tražiti da pomognu identificirati najrelevantnija poslovna pitanja i podatke za odgovore na njih.

Tijekom posljednjih nekoliko desetljeća, tehnološki napredak u pohrani i procesorskoj snazi omogućio je neke inovativne proizvode temeljene na strojnom učenju, kao što su Netflix-ov mehanizam za preporuke i autonomni automobili.

Algoritmi strojnog učenja obično se izrađuju pomoću okvira koji ubrzavaju razvoj rješenja, kao što su TensorFlow i PyTorch.[3]

Budući da se duboko učenje i strojno učenje koriste naizmjenično, vrijedno je uočiti nijanse između njih dvoje. Strojno učenje, duboko učenje i neuronske mreže sve su podpodručja umjetne inteligencije. Međutim, neuronske mreže zapravo su podpodručje strojnog učenja, a duboko učenje podpodručje neuronskih mreža.

2.3.1. Duboko učenje

Duboko učenje je tehnika strojnog učenja koja uči računala da rade ono što je ljudima prirodno: uče na primjeru. Duboko učenje ključna je tehnologija iza autonomnih vozila, koja im omogućuje prepoznavanje znaka stop ili razlikovanje pješaka od rasvjetnog stupa. Također je ključ za glasovnu kontrolu u potrošačkim uređajima kao što su telefoni, tableti, televizori i hands-free zvučnici. Duboko učenje u posljednje vrijeme dobiva puno pozornosti i to s dobrim razlogom. Postiže rezultate koji prije nisu bili mogući.

U dubokom učenju, računalni model uči obavljati zadatke klasifikacije izravno iz slika, teksta ili zvuka. Modeli dubokog učenja mogu postići najsuvremeniju točnost, koja ponekad

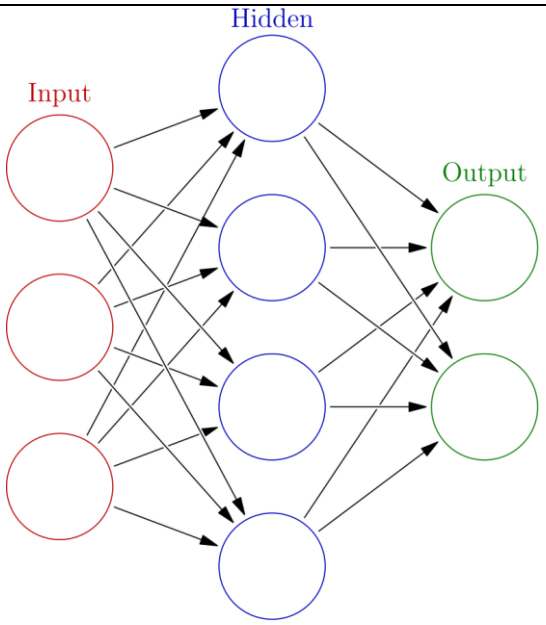
premašuje performanse na ljudskoj razini. Modeli se obučavaju pomoću velikog skupa označenih podataka i arhitektura neuronskih mreža koje sadrže mnogo slojeva.[4]

2.3.2. Neuronske mreže

Neuronska mreža računalni je sustav za učenje koji koristi mrežu funkcija za razumijevanje i prevođenje unosa podataka jednog oblika u željeni izlaz, obično u drugom obliku. Koncept umjetne neuronske mreže inspiriran je ljudskom biologijom i načinom na koji neuroni ljudskog mozga funkcioniraju zajedno kako bi razumjeli unose ljudskih osjetila.

Neuronske mreže samo su jedan od mnogih alata i pristupa koji se koriste u algoritmima strojnog učenja. Sama neuronska mreža može se koristiti kao dio u mnogim različitim algoritmima strojnog učenja za obradu složenih ulaznih podataka u prostor koji računala mogu razumjeti.

Neuronske mreže danas se primjenjuju na mnoge probleme iz stvarnog života, uključujući prepoznavanje govora i slika, filtriranje neželjene e-pošte, financije i medicinsku dijagnozu itd. Algoritmi strojnog učenja koji koriste neuronske mreže općenito ne moraju biti programirani posebnim pravilima koja definiraju što očekivati od ulaza. Algoritam za učenje neuronske mreže umjesto toga uči obradom mnogih označenih primjera (tj. podataka s "odgovorima") koji se daju tijekom obuke i pomoću ovog ključa odgovora kako bi naučio koje su karakteristike ulaza potrebne za konstruiranje ispravnog izlaza. Nakon što se obradi dovoljan broj primjera, neuronska mreža može početi obrađivati nove, neviđene ulaze i uspješno vraćati točne rezultate. Što više primjera i različitih unosa program vidi, to su rezultati obično točniji jer program uči s iskustvom.[5]



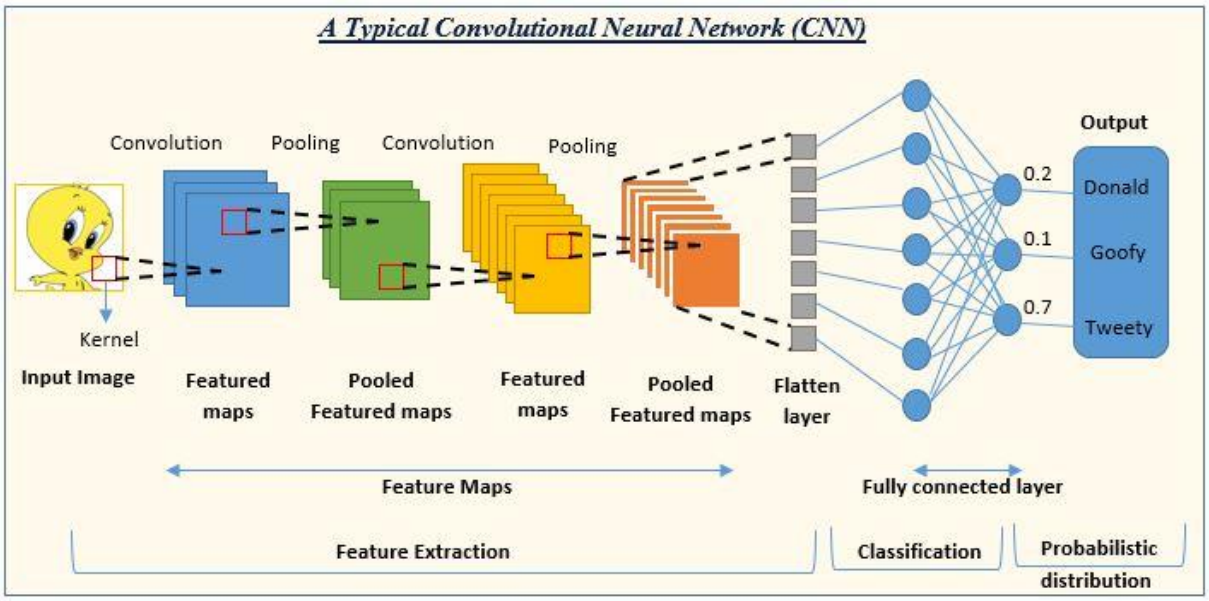
Slika 1. Neuronska mreža

Neuronske mreže mogu se primijeniti na širok raspon problema i mogu procijeniti mnoge različite vrste ulaza, uključujući slike, videozapise, datoteke, baze podataka i još mnogo toga. Oni također ne zahtijevaju eksplicitno programiranje za tumačenje sadržaja tih ulaza.

Zbog općeg pristupa rješavanju problema koji neuronske mreže nude, gotovo da nema ograničenja u područjima u kojima se ova tehnika može primijeniti. Neke uobičajene primjene neuronskih mreža danas uključuju prepoznavanje slika/uzoraka, predviđanje putanje samoupravnog vozila, prepoznavanje lica, rudarenje podataka, filtriranje neželjene e-pošte, medicinsku dijagnozu i istraživanje raka. Danas postoji mnogo više načina na koje se neuronske mreže koriste, a usvajanje se brzo povećava.[5]

2.3.2.1. Konvolucijske neuronske mreže

Postoje različite vrste neuronskih mreža koje se koriste za različite slučajeve upotrebe i vrste podataka. Na primjer, rekurentne neuronske mreže obično se koriste za obradu prirodnog jezika i prepoznavanje govora, dok se konvolucijske neuronske mreže (ConvNets ili CNN) češće koriste za zadatke klasifikacije i računalnog vida. Prije CNN-a, za identifikaciju objekata na slikama korištene su ručne, dugotrajne metode izdvajanja značajki. Međutim, konvolucijske neuronske mreže sada pružaju skalabilniji pristup klasifikaciji slika i zadacima prepoznavanja objekata, koristeći principe linearne algebre, posebno matričnog množenja, za prepoznavanje uzoraka unutar slike. Ipak, mogu biti računalno zahtjevni, zahtijevajući grafičke procesorske jedinice (GPU) za treniranje modela.[6]



Slika 2. Konvolucijska neuronska mreža

Konvolucijske neuronske mreže razlikuju se od ostalih neuronskih mreža svojim superiornim performansama s ulaznim signalima slike, govora ili zvuka. Imaju tri glavne vrste slojeva, a to su:

- Konvolucijski sloj(engl. Convolutional layer)
- Sloj udruživanja(engl. Pooling layer)
- Potpuno povezani (FC) sloj(engl. Fully connected layer)

Konvolucijski sloj je prvi sloj konvolucijske mreže. Dok konvolucijski slojevi mogu biti praćeni dodatnim konvolucijskim slojevima ili slojevima udruživanja, potpuno povezani sloj je završni sloj. Sa svakim slojem CNN postaje sve složeniji, identificirajući veće dijelove slike. Raniji slojevi fokusirani su na jednostavne značajke, kao što su boje i rubovi. Kako slikovni podaci napreduju kroz slojeve CNN-a, počinju prepoznavati veće elemente ili oblike objekta dok konačno ne identificira željeni objekt.

Većina izračuna događa se u konvolucijskom sloju, koji je temeljni blok CNN-a. Drugi konvolucijski sloj može slijediti početni konvolucijski sloj. Proces konvolucije uključuje jezgru ili filtar unutar ovog sloja koji se kreće preko receptivnih polja slike, provjeravajući je li neka značajka prisutna na slici.

Tijekom višestrukih iteracija, kernel prelazi preko cijele slike. Nakon svake iteracije izračunava se točkasti umnožak između ulaznih piksela i filtra. Konačni rezultat niza točaka poznat je kao mapa značajki ili konvolvirana značajka. U konačnici, slika se pretvara u numeričke vrijednosti u ovom sloju, što CNN-u omogućuje tumačenje slike i izdvajanje relevantnih uzoraka iz nje.

Poput konvolucijskog sloja, sloj udruživanja također prebacuje kernel ili filtar preko ulazne slike. Ali za razliku od konvolucijskog sloja, sloj udruživanja smanjuje broj parametara u ulazu i također rezultira gubitkom nekih informacija. S pozitivne strane, ovaj sloj smanjuje složenost i poboljšava učinkovitost CNN-a.

FC sloj je mjesto na kojem se vrši klasifikacija slika u CNN-u na temelju značajki ekstrahiranih u prethodnim slojevima. Ovdje *potpuno povezano* znači da su svi ulazi ili čvorovi iz jednog sloja povezani sa svakom aktivacijskom jedinicom ili čvorom sljedećeg sloja.

Svi slojevi u CNN-u nisu u potpunosti povezani jer bi to rezultiralo nepotrebno gustom mrežom. To bi također povećalo gubitke i utjecalo na kvalitetu izlaza, a bilo bi računalno skupo.[7]

3. KORIŠTENE DATOTEKE

3.1. Python

Python je interpretirani, objektno orijentirani programski jezik visoke razine s dinamičkom semantikom. Njegove visoke razine ugrađenih podatkovnih struktura, u kombinaciji s dinamičkim tipkanjem i dinamičkim vezanjem. Čine ga vrlo atraktivnim za brzi razvoj aplikacija, kao i za korištenje kao skriptni ili ljepljivi jezik za povezivanje postojećih komponenti. Pythonova jednostavna sintaksa koju je lako naučiti naglašava čitljivost i stoga smanjuje troškove održavanja programa. Python podržava module i pakete, što potiče modularnost programa i ponovnu upotrebu koda. Python tumač i opsežna standardna biblioteka dostupni su u izvornom i binarnom obliku bez naknade za sve glavne platforme i mogu se besplatno distribuirati.[8] Ovdje je korištena verzija za Windows, Python 3.7.3.

3.2. OpenCV

OpenCV (engl. Open Source Computer Vision Library) biblioteka je softvera za računalni vid i strojno učenje otvorenog koda. OpenCV je napravljen kako bi osigurao zajedničku infrastrukturu za aplikacije računalnog vida i ubrzao korištenje strojne percepcije u komercijalnim proizvodima. Budući da je proizvod s licencom za Apache 2, OpenCV tvrtkama olakšava korištenje i izmjenu koda.

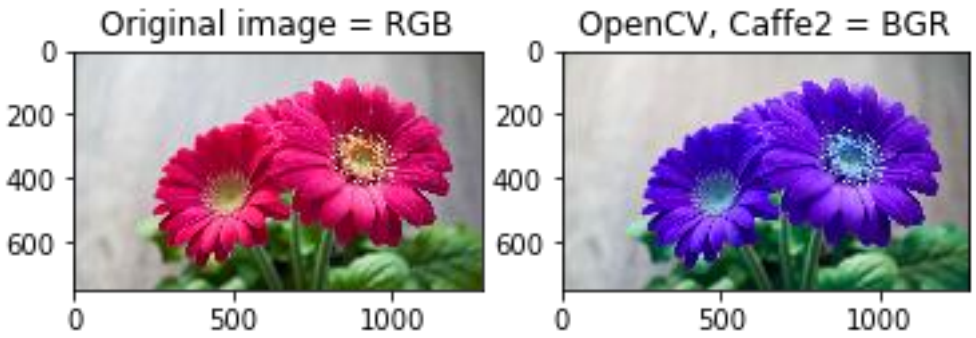
Knjižnica ima više od 2500 optimiziranih algoritama, što uključuje opsežan skup klasičnih i najsuvremenijih algoritama računalnog vida i strojnog učenja. Ovi se algoritmi mogu koristiti za otkrivanje i prepoznavanje lica, identificiranje objekata, klasificiranje ljudskih radnji u videozapisima, praćenje pokreta kamere, praćenje pokretnih objekata, izdvajanje 3D modela objekata, proizvodnju 3D oblaka točaka iz stereo kamera, spajanje slika kako bi se proizvela visoka rezolucija sliku cijelog prizora.

Ima C++, Python, Java i MATLAB sučelja i podržava Windows, Linux, Android i Mac OS. OpenCV se uglavnom oslanja na aplikacije za viziju u stvarnom vremenu.[9]

```
import cv2
```

Slika 3. Uvođenje biblioteke OpenCV

Iako je uobičajen prostor boja u računalima RGB (Red – Green – Blue), OpenCV ima format BGR (Blue – Green – Red) jer u vrijeme kada je napravljen taj je format bio češći. To znači da će vrijednosti crvene i plave boje zamijeniti mjesta pa je zapis drugačiji.



Slika 4. Razlike u RGB i BGR interpretaciji boja

Slika 4. prikazuje osnovnu razlika između dva formata čitanja boja. Prva slika je prikaz u RGB, a druga u BGR formatu gdje se vidi zamjena crvene i plave boje.

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Slika 5. Pretvaranje slike iz BGR u RGB format

Slika 5. prikazuje pretvaranje BGR formata u RGB format u Pythonu

3.3. NumPy

NumPy je temeljni paket za znanstveno računalstvo u Pythonu. To je biblioteka koja pruža višedimenzionalni objekt niza, razne izvedene objekte (kao što su maskirani nizovi i matrice) i asortiman rutina za brze operacije na nizovima, uključujući matematičke, logičke, manipulaciju oblikom, sortiranje, odabir, I/O, diskretne Fourierove transformacije, osnove linearne algebre, osnovne statističke operacije, slučajne simulacije i još mnogo toga. U središtu NumPy paketa nalazi se objekt *ndarray*. *Ndarray* enkapsulira n-dimenzionalne nizove homogenih tipova podataka, s mnogim operacijama koje se izvode u kompajliranom kodu za izvedbu. U Pythonu imamo liste koje služe kao nizovi, ali se sporo obrađuju. NumPy ima za cilj pružiti objekt niza koji je do 50x brži od tradicionalnih Python listi.[10]

NumPy nizovi pohranjeni su na jednom kontinuiranom mjestu u memoriji za razliku od popisa, tako da im procesi mogu pristupiti i manipulirati njima vrlo učinkovito. Ovo se ponašanje u informatici naziva lokalnost reference. Ovo je glavni razlog zašto je NumPy brži od lista. Također je optimiziran za rad s najnovijim CPU arhitekturama.[11]

Nizovi se vrlo često koriste u znanosti o podacima, gdje su brzina i resursi vrlo važni.

```
import numpy as np
```

Slika 6. Uvođenje biblioteke NumPy

3.4. PyTorch

PyTorch je potpuno opremljen okvir za izgradnju modela dubokog učenja. Napisan u Pythonu, relativno je jednostavan za učenje i korištenje za većinu programera strojnog učenja. PyTorch je prepoznatljiv po svojoj izvrsnoj podršci za GPU-ove i korištenje auto-diferencijacije obrnutog načina rada, što omogućuje izmjenu računskih grafikona u hodu. To ga čini popularnim izborom za brzo eksperimentiranje i izradu prototipova.

Okvir kombinira učinkovite i fleksibilne GPU-ubrzane pozadinske biblioteke iz Torcha s intuitivnim Python sučeljem koje se fokusira na brzu izradu prototipova, čitljiv kod i podršku za najširi mogući izbor modela dubokog učenja.

Ključne komponente PyTorcha su tenzori i grafovi.

Tenzori su temeljna vrsta podataka PyTorcha, slična višedimenzionalnom polju, koja se koristi za pohranu i manipuliranje ulazima i izlazima modela, kao i parametrima modela. Tenzori su slični NumPyjevima *ndarrayima*, osim što tenzori mogu raditi na GPU-u kako bi se ubrzalo računanje.

Grafovi su strukture podataka koje se sastoje od povezanih čvorova (koji se nazivaju vrhovi) i rubova. Svaki moderni radni okvir za dubinsko učenje temelji se na konceptu grafova, gdje su neuronske mreže predstavljene kao struktura grafa izračuna. PyTorch vodi evidenciju tenzora i izvršenih operacija u usmjerenom acikličkom grafu (DAG) koji se sastoji od *Function* objekata. U ovom DAG-u listovi su ulazni tenzori, korijeni su izlazni tenzori.

U mnogim popularnim okvirima, uključujući TensorFlow, računski graf je statički objekt. PyTorch se temelji na dinamičkim proračunskim grafovima, gdje se računski graf gradi i ponovno gradi tijekom izvođenja, s istim kodom koji izvodi proračune za prolaz naprijed također stvara strukturu podataka potrebnu za širenje unazad. PyTorch je prvi okvir za dubinsko učenje po principu definiranja koji odgovara mogućnostima i performansama okvira statičkih grafova kao što je TensorFlow, što ga čini dobrim za sve, od standardnih konvolucijskih mreža do rekurentnih neuronskih mreža.

Poznato je da je PyTorch okvir prikladan i fleksibilan, s primjerima koji pokrivaju učenje s potkrepljivanjem, klasifikaciju slika i obradu prirodnog jezika kao uobičajenije slučajeve upotrebe.[12]

```
import torch
```

Slika 7. Uvođenje biblioteke PyTorch

4. SKUPOVI PODATAKA

Procjena dubine zadatak je mjerenja udaljenosti svakog piksela u odnosu na kameru. Dubina se izvlači iz monokularnih (pojedinačnih) ili stereo (više prikaza scene) slika. Tradicionalne metode koriste geometriju s više pogleda za pronalaženje odnosa između slika. Novije metode mogu izravno procijeniti dubinu minimiziranjem regresijskog gubitka ili učenjem generiranja novog prikaza iz niza. Najpopularniji skupovi podataka su Stanford2D3D Panoramic, NYU-Depth V2, eBDtheque.[13]

4.1. Stanford2D3D Panoramic skup podataka

Skup podataka 2D-3D-S pruža niz međusobno registriranih modaliteta iz 2D, 2.5D i 3D domena, sa semantičkim i geometrijskim komentarima na razini instance. Pokriva više od 6000 m2 prostora prikupljenih u 6 velikih zatvorenih područja koja potječu iz 3 različite zgrade. Sadrži više od 70 000 RGB slika, zajedno s odgovarajućim dubinama, normalama površine, semantičkim komentarima, globalnim XYZ slikama (sve u obliku običnih i 360° jednakokutnih slika), kao i informacije o kameri. Također uključuje registrirane neobrađene i semantički označene 3D mreže i oblake točaka. Skup podataka omogućuje razvoj zajedničkih i kros-modalnih modela učenja i potencijalno nenadziranih pristupa koji koriste pravilnosti prisutne u velikim zatvorenim prostorima.[14]

Najbolji model, odnosno onaj koji ima najmanju apsolutnu relativnu grešku je *PanoFormer*

4.2. NYU-Depth V2 skup podataka

Skup podataka NYU-Depth V2 sastoji se od video sekvenci iz različitih scena u zatvorenom prostoru koje su snimile RGB i Depth kamere s Microsoft Kinecta. Sadrži:

- 1449 gusto označenih parova poravnatih RGB i dubinskih slika
- 464 nove scene snimljene iz 3 grada
- 407 024 novih neoznačenih okvira

Svaki objekt označen je klasom i brojem instance. Skup podataka ima nekoliko komponenti:

- Označeno: podskup video podataka popraćen gustim oznakama više klasa. Ovi su podaci također prethodno obrađeni kako bi se popunile nedostajuće oznake dubine.
- Neobrađeni podaci: neobrađeni RGB, dubina i podaci akcelerometra koje daje Kinect.
- Kutija s alatima: Korisne funkcije za manipuliranje podacima i naljepnicama.[15]

Model s najmanjim RMS-om je SwinV2-L 1K-MIM.

RMS vrijednost skupa vrijednosti (ili valnog oblika kontinuiranog vremena) kvadratni je korijen aritmetičke sredine kvadrata vrijednosti ili kvadrata funkcije koja definira kontinuirani valni oblik.

4.3. EBDtheque skup podataka

Baza podataka eBDtheque izbor je od sto stranica stripa iz Amerike, Japana (manga) i Europe.

4.3.1. MiDas model

Model MiDas je ima drugi najbolji apsolutni relativni rezultat , te je on odabrani model ovog rada.

Uspjeh monokularne procjene dubine oslanja se na velike i raznolike setove za vježbanje. Zbog izazova povezanih sa stjecanjem guste dubine temeljne istine u različitim okruženjima na skali, pojavio se niz skupova podataka s različitim karakteristikama i pristranostima. MiDas razvija alate koji omogućuju miješanje više skupova podataka tijekom obuke, čak i ako su njihove napomene nekompatibilne. Konkretno, predlažen je robustan cilj obuke koji je nepromjenjiv na promjene u dubinskom rasponu i mjerilu, zagovara se korištenje principijelnog učenja s više ciljeva za kombiniranje podataka iz različitih izvora i ističe se važnost prethodnog osposobljavanja kodera za pomoćne zadatke. MiDas eksperimentira s pet različitih skupova podataka za obuku, uključujući novi, masivni izvor podataka: 3D filmove. Kako bi pokazali snagu generalizacije pristupa, koriste *zero-shot cross-dataset transfer,* tj. procjenjuju skupove podataka koji nisu viđeni tijekom obuke. Eksperimenti potvrđuju da miješanje podataka iz komplementarnih izvora uvelike poboljšava procjenu monokularne dubine. MiDas pristup jasno nadmašuje konkurentske metode u različitim skupovima podataka, postavljajući nove standarde za monokularnu procjenu dubine. [16]

5. RAZVOJ RAČUNALNOG MODELA

Prvo se instaliraju potrebni programi – Python, te potrebne biblioteke za rad : OpenCV, NumPy, PyTorch.

5.1. Računalni model za procjenu dubine prostora iz 2D slike

5.1.1. Uvođenje biblioteka i odabir modela

Prvo se otvara novi Python projekt te se uvode potrebne biblioteke na sljedeći način:

```
import cv2
import torch
import numpy as np
```

Slika 8. Uvođenje biblioteka

```
# Load a MiDas model for depth estimation
model_type = "DPT_Large"     # MiDaS v3 - Large     (highest accuracy, slowest inference speed)
#model_type = "DPT_Hybrid"   # MiDaS v3 - Hybrid    (medium accuracy, medium inference speed)
#model_type = "MiDaS_small"  # MiDaS v2.1 - Small   (lowest accuracy, highest inference speed)

midas = torch.hub.load("intel-isl/MiDaS", model_type)
```

Slika 9. Odabir i uvođenje modela

Postoji više MiDas model : *DPT_Large*, *DPT_Hybrid* te *MiDas_small*

Za uvođenje modela koji se koristi za procjenu udaljenosti koristi se funkcija *torch.hub.load()*, prvi argument funkcije je repozitorij na GitHubu gdje se nalazi model kojeg učitavamo, dok je drugi element *string*, to jest ime modela kojeg želimo učitati.

Odabrani model u ovom slučaju je *DPT_Large* koji ima najveću preciznost, ali ima i najmanju brzinu interferencije.

5.1.2. Učitavanje slike

```
device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
midas.to(device)
midas.eval()

midas_transforms = torch.hub.load("intel-isl/MiDaS", "transforms")

if model_type == "DPT_Large" or model_type == "DPT_Hybrid":
    transform = midas_transforms.dpt_transform
else:
    transform = midas_transforms.small_transform
```

Slika 10. Učitavanje transformacija

Prvo se provjerava postoji li grafički procesor(GPU), ako nije dostupan, koristi se obični procesor(CPU).

midas.to(device) je naredba kojom se model premješta na GPU ili CPU.

midas.eval() je vrsta prekidača za neke specifične slojeve/dijelove modela koji se ponašaju drugačije tijekom vremena treninga i zaključivanja (ocjenjivanja). Moraju se isključiti tijekom evaluacije modela, a funkcija *.eval()* to čini.

Zatim se učitavaju *transoforms* sa githuba, koji se koriste za promjenu veličine i normaliziranje slike. *Transforms* se odabire ovisno o modelu kojeg učitavamo.

Transforms su uobičajene transformacije slike.

```
img = cv2.imread("C:\\Users\\Ilija\\Desktop\\faks\\zavrsni\\dog.jpg")

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

input_batch = transform(img).to(device)
```

Slika 11. Odabir slike

Funkcija *imread()* učitava sliku s adrese(path) koju joj postavimo za argument.

Također vraća oblik *numpy.ndarrey.*

Mijenja se i redoslijed boja iz BGR u RGB jer neke funkcije koriste BGR redoslijed, a neke RGB redoslijed, te se primjenjuju transformacije na sliku.

5.1.3. Stvaranje predikcije slike

```
with torch.no_grad():
    prediction = midas(input_batch)

    prediction = torch.nn.functional.interpolate(
        prediction.unsqueeze(1),
        size=img.shape[:2],
        mode="bicubic",
        align_corners=False,
    ).squeeze()

depth_map = prediction.cpu().numpy()

depth_map = cv2.normalize(depth_map, None, 0, 1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
```

Slika 12. Dio koda za stvaranje predikcije slike

Funkcija *with torch.no_grad()* „govori“ PyTorchu da ne izračunava gradijente, a program ga izričito koristi ovdje (kao kod većine neuronskih mreža) kako ne bi ažurirao gradijente jer bi to utjecalo na povratno širenje.

U 2. liniji koda se primjenjuju *Midas* model na argument *input_batch* odnosno na sliku koja je odabrana, output(izlazna vrijednost) slike se sprema kao varijabla *prediction.*

Funkcija torch.nn.functional.interpolate() vraća sliku u originalni oblik, jer je promjenjena veličina slika prilikom transformacija, također funkcija interpolira output slike, koristi se način interpolacije *bicubic.* Funkcija *.squeeze()* služi za stvaranje slike umjesto niza.

Pod varijablom *depth_map* se sprema dobivena slika, *.cpu()* kopira podatke(tenzor) na CPU, a ako cu već na CPU-u onda se ništa ne događa, dok *.numPy()* pretvara tenzor u NumPy niz.

U zadnjoj liniji koda se slika normalizira pomoću funkcije *cv2.normalize()*, parametri funkcije su:

- *depth_map*, izvor, tj. slika koja se normalizira
- 0 i 1, su donja i gornja granica raspona piksela
- *norm_type=cv2.NORM_MINMAX* postavlja minimalnu i makismalnu granicu
- *dtype=cv2.CV_64F* označava interpretaciju podataka kao 64-bitni floating-point

5.1.4. Dobivanje konačne procjene slike

```
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)


depth_map = (depth_map*255).astype(np.uint8)

depth_map = cv2.applyColorMap(depth_map , cv2.COLORMAP_MAGMA)

cv2.imwrite('opencv5'+'.png', depth_map)
```

Slika 13. Ispis slike

U liniji kod 1. se ponovno mijenja format boja iz RGB formata u BGR format

U 4. liniji kod množimo sa 255 kako bi dobili 8-bit *integere*, odnosno cijele brojeve bez predznaka, a za slike se uvijek koriste takvi *integeri*.

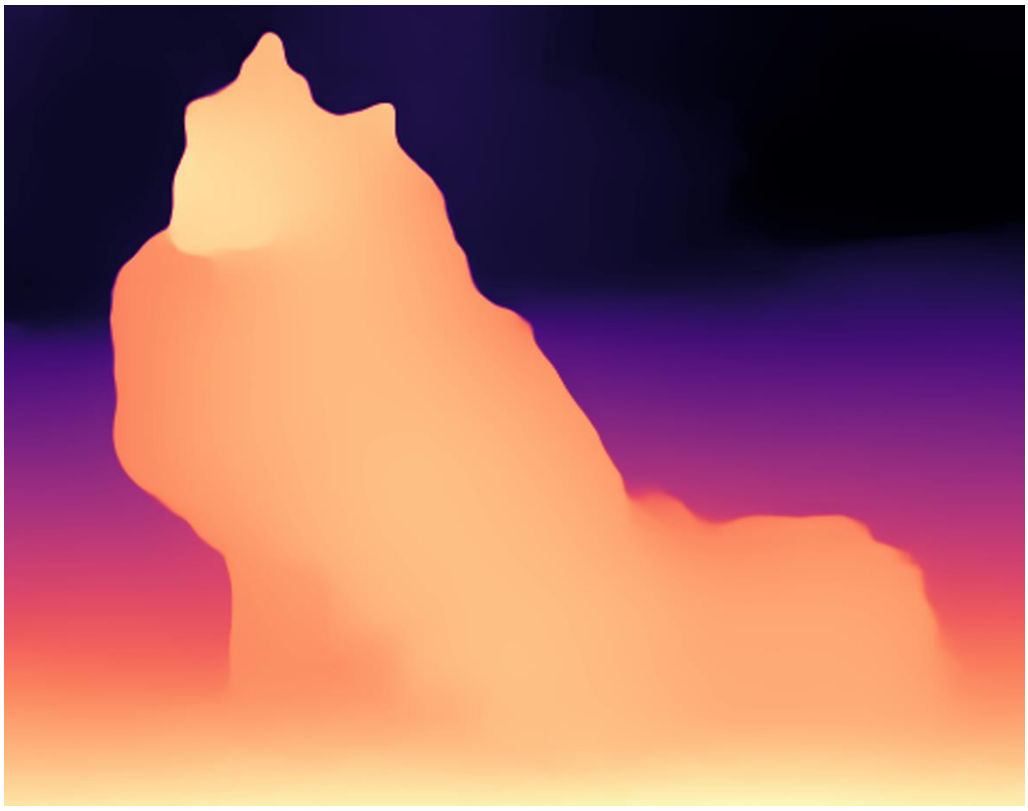
Pomoću funkcije *cv2.applyColorMap()* se pretvara slika koja je dosad bila u nijansama sive, u sliku u boji. U OpenCV-u postoji 21 različita mapa s bojama.

Na kraju su konačna procjena dubine prostora sprema u istu mapu u kojoj se nalazi kod pomoću funkcije *cv2.imwrite()*, sprema se u *png.* obliku te u ovom slučaju pod imenom „opencv5“

U nastavku se mogu vidjeti konačni rezultati na primjeru dvije slike.

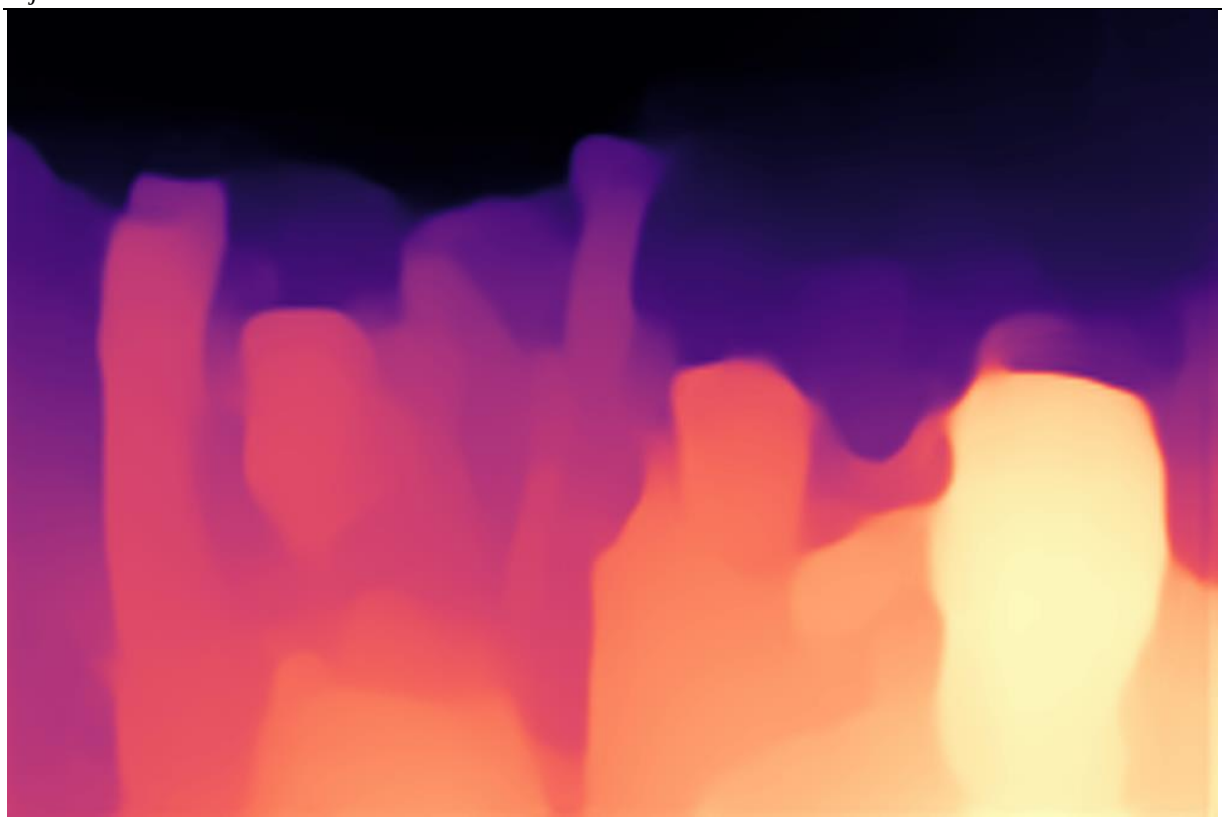
Slika 14. Slika psa u prirodi

Slika 15. Procjena dubine prostora pomoću modela MiDaS_small



Slika 16. Procjena dubine prostora pomoću modela DPT_Large

Slika 17. Slika skupine ljudi



Slika 18. Procjena dubine prostora pomoću modela MiDaS_small

Slika 19. Procjena dunine prostora pomoću modela DPT_Large

5.2. Računalni model za procjenu dubine prostora iz 2D videozapisa

Za procjenu dubine prostora iz 2D videozapisa potrebno je malo preinačiti kod koji se koristi kod procjene dubine prostora iz 2D slike. U ovom poglavlju će biti objašnjene samo te promjene, dok se cijeli kod nalazi u prilogu.

5.2.1. Učitavanje videozapisa

```
cap = cv2.VideoCapture("C:\\Users\\Ilija\\Desktop\\faks\\zavrsni\\video1.mp4")

frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

size = (frame_width, frame_height)

result = cv2.VideoWriter('output.avi',
                         cv2.VideoWriter_fourcc(*'MJPG'),
                         10, size)
```

Slika 20. Učitavanje videozapisa

Funkcija *cv2.Videocapure()* biblioteke OpenCV omogućuje dohvaćanje odnosno čitanje odabranog videozapisa, potrebno je unutar zagrada navesti točnu lokaciju videozapisa(path), da bi se on mogao pročitati.

U 3. i 4. liniji kod se pročita širina i visina videa, odnosno broj piksela videa, te se spremaju kao odabrane varijable.

Pod varijablom *size* se sprema rezolucija videozapisa. U 8. liniji koda funkcija *cv2.VideoWriter()* stvara novi video pod imenom *output.avi*, novi video se sprema u istu mapu kao i kod, a može se i odabrati željena lokacija, novi video ima istu rezoluciju kao i odabrani video te se sprema u odabranom formatu.

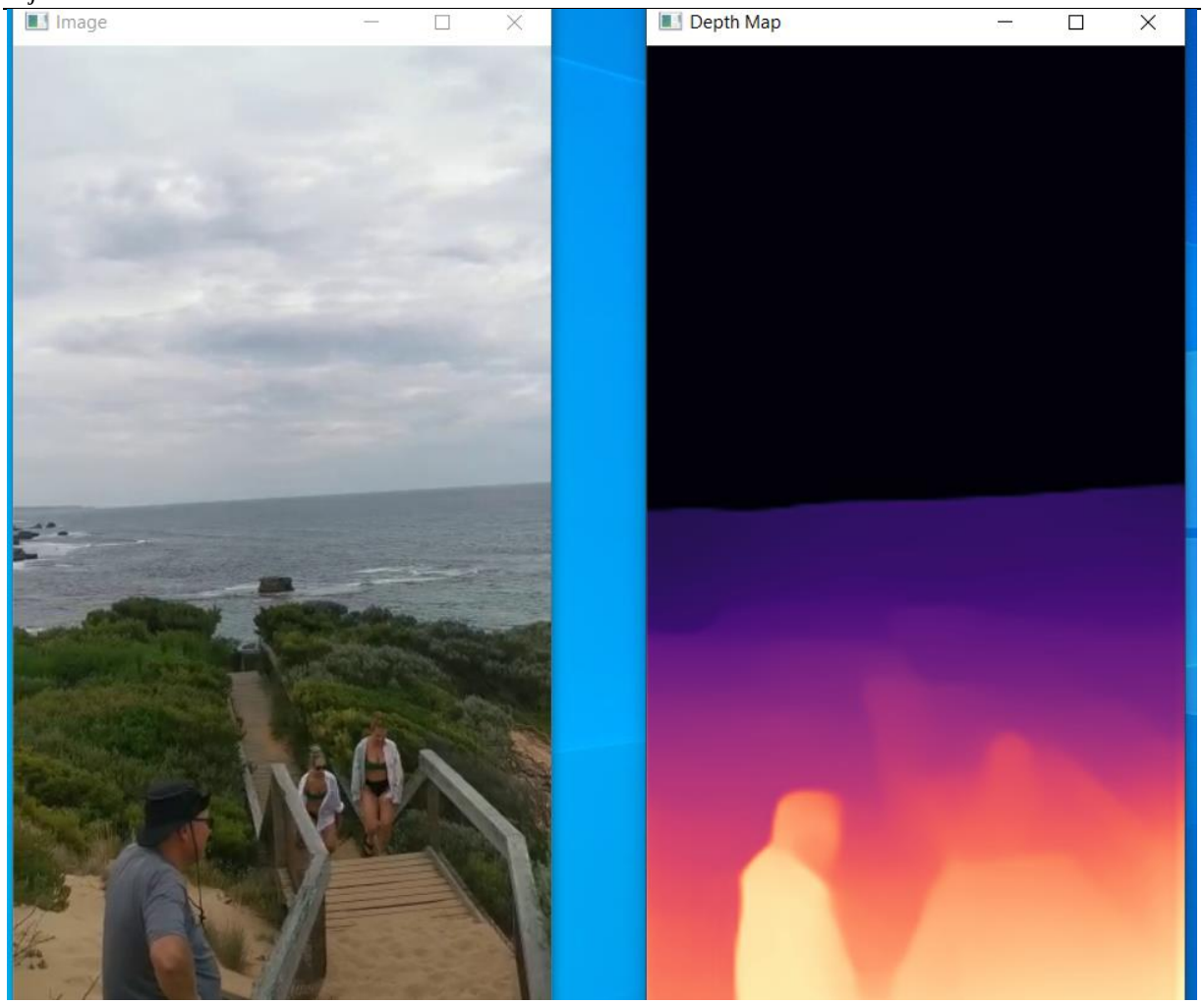
5.2.2. Obrada videozapisa

```
while cap.isOpened():

    success, img = cap.read()

    if(success == False):
        break

    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    input_batch = transform(img).to(device)

    with torch.no_grad():
        prediction = midas(input_batch)

        prediction = torch.nn.functional.interpolate(
            prediction.unsqueeze(1),
            size=img.shape[:2],
            mode="bicubic",
            align_corners=False,
        ).squeeze()

    depth_map = prediction.cpu().numpy()

    depth_map = cv2.normalize(depth_map, None, 0, 1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)

    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

    depth_map = (depth_map*255).astype(np.uint8)
    depth_map = cv2.applyColorMap(depth_map , cv2.COLORMAP_MAGMA)

    result.write(depth_map)

    cv2.imshow('Image', img)
    cv2.imshow('Depth Map', depth_map)

    if cv2.waitKey(5) & 0xFF == 27:
        break

cap.release()
```

Slika 21. Obrada videozapisa

Uvodi se petlja *while* koja se vrti dok traje video, naredbom *read* se učita rezolucija iz videozapisa, pod varijablom *success* se sprema *bool*(True ili False) koji označava je li rezolucija uspješno spremljena. Dok se pod varijablom *img* sprema zabilježena slika.

Ako nije uspješno pročitana rezolucija, program izlazi iz petlje. Ostatak koda je već objašnjen kod procjene dubine prostora iz 2D slike, osim zadnjeg dijela gdje se pomoću funkcije *cv2.imshow()* usporedno prikazuju originalni videozapis te videozapis sa procjenom dubine prostora. Pritiskom na tipku „Esc“ zaustavlja se videozapis, te se može ugasiti.



Slika 22. Procjena dubine prostora iz videozapisa

Korišteni model za Sliku 22. je *MiDaS_small*, koji ima najmanju točnost, ali i visoku brzinu interferencije. Kod korištenja *DPT_Large* modela dolazi do značajnog usporenja videozapisa, jer CPU ne može tako brzo obraditi sliku.

6. ZAKLJUČAK

Konvolucijske neuronske mreže su se pokazale vrlo učinkovitim alatima za procjenu dubine prostora iz 2D slika. Ovakav pristup omogućuje da se uz pomoć neuronske mreže nauči mapiranje iz 2D slike u dubinsku mapu, što se može koristiti za razne primjene u računalnom vidu, robotici i drugim područjima.

Postoje različiti pristupi za rješavanje ovog problema, kao što su metode bazirane na stereo parovima slika, metode bazirane na strukturi scene te metode bazirane na jednoj slici. U ovom radu se koristila metoda bazirana na jednoj slici koja se temelji na konvolucijskoj neuronskoj mreži. Ova metoda je pokazala vrlo dobre rezultate u usporedbi s drugim metodama i postaje sve popularnija u posljednje vrijeme.

Izvedene su brojne eksperimente s različitim arhitekturama konvolucijskih neuronskih mreža, kao i s različitim skupovima podataka za učenje i testiranje. Rezultati eksperimenata pokazuju da je moguće postići vrlo dobre rezultate u procjeni dubine, pri čemu se ovisno o metodi, arhitekturi i skupu podataka postižu različiti stupnjevi točnosti.

Ukratko, korištenje konvolucijskih neuronskih mreža za procjenu dubine prostora iz 2D slika predstavlja vrlo zanimljiv i obećavajući pristup koji će sigurno biti sve više istraživan i primjenjivan u budućnosti.

LITERATURA

[1] Enciklopedija: https://enciklopedija.hr/natuknica.aspx?ID=63150, pristupljeno 19.02.2023.

[2] IBM: https://www.ibm.com/topics/computer-vision, pristupljeno 19.02.2023.

[3] IBM: https://www.ibm.com/topics/machine-learning, pristupljeno 19.02.2023.

[4] Mathworks: https://www.mathworks.com/discovery/deep-learning.html, pristupljeno 20.02.2023.

[5] Deepai: https://deepai.org/machine-learning-glossary-and-terms/neural-network, pristupljeno 20.02.2023.

[6] IBM: https://www.ibm.com/topics/convolutional-neural-networks, pristupljeno 23.02.2023.

[7] Techtarget: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network, pristupljeno 23.02.2023.

[8] Python: https://www.python.org/doc/essays/blurb/, pristupljeno 20.02.2023.

[9] OpenCV: https://opencv.org/about/, pristupljeno 20.02.2023.

[10] NumPy: https://numpy.org/doc/stable/user/whatisnumpy.html, pristupljeno 20.02.2023.

[11] w3schools: https://www.w3schools.com/python/numpy/numpy_intro.asp, pristupljeno 21.02.2023.

[12] nvidia: https://www.nvidia.com/en-us/glossary/data-science/pytorch/, pristupljeno 21.02.2023.

[13] paperswithcode: https://paperswithcode.com/task/depth-estimation, pristupljeno 21.02.2023.

[14] paperswithcode: https://paperswithcode.com/dataset/2d-3d-s, pristupljeno 21.02.2023.

[15] paperswithcode: https://paperswithcode.com/dataset/nyuv2, pristupljeno 22.02.2023.

[16] paperswithcode: https://paperswithcode.com/paper/towards-robust-monocular-depth-estimation, pristupljeno 22.02.2023.

PRILOZI

I. Programski kod - slika

II. Programski kod - videozapis

I. Programski kod – slika

```
import cv2
import torch
import numpy as np

#model_type = "DPT_Large"     # MiDaS v3 - Large     (najveća preciznost, slowest inference speed)
#model_type = "DPT_Hybrid"   # MiDaS v3 - Hybrid    (srednja preciznost, medium inference speed)
model_type = "MiDaS_small"  # MiDaS v2.1 - Small   (najmanja preciznost, highest inference speed)

midas = torch.hub.load("intel-isl/MiDaS", model_type)

device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
midas.to(device)
midas.eval()

midas_transforms = torch.hub.load("intel-isl/MiDaS", "transforms")

if model_type == "DPT_Large" or model_type == "DPT_Hybrid":
        transform = midas_transforms.dpt_transform
else:
        transform = midas_transforms.small_transform

img = cv2.imread("C:\\Ilija\\dog.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
input_batch = transform(img).to(device)

with torch.no_grad():

        prediction = midas(input_batch)
        prediction = torch.nn.functional.interpolate(
                prediction.unsqueeze(1),
                size=img.shape[:2],
                mode="bicubic",
                        align_corners=False,
        ).squeeze()

depth_map = prediction.cpu().numpy()

depth_map = cv2.normalize(depth_map, None, 0, 1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
```

```
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
depth_map = (depth_map*255).astype(np.uint8)

depth_map = cv2.applyColorMap(depth_map , cv2.COLORMAP_MAGMA)

cv2.imwrite('opencv2223'+'.png', depth_map)
```

II. Programski kod – videozapis

```
import cv2
import torch
import numpy as np


#model_type = "DPT_Large"     # MiDaS v3 - Large     (highest accuracy, slowest
inference speed)
#model_type = "DPT_Hybrid"   # MiDaS v3 - Hybrid    (medium accuracy, medium
inference speed)
model_type = "MiDaS_small"  # MiDaS v2.1 - Small   (lowest accuracy, highest
inference speed)

midas = torch.hub.load("intel-isl/MiDaS", model_type)

device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
midas.to(device)
midas.eval()

midas_transforms = torch.hub.load("intel-isl/MiDaS", "transforms")

if model_type == "DPT_Large" or model_type == "DPT_Hybrid":
    transform = midas_transforms.dpt_transform
else:
    transform = midas_transforms.small_transform

cap = cv2.VideoCapture("C:\\Users\\Ilija\\Desktop\\faks\\zavrsni\\video1.mp4")
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
size = (frame_width, frame_height)

result = cv2.VideoWriter('output4.avi',  cv2.VideoWriter_fourcc(*'MJPG'), 10, size)
```

```
while cap.isOpened():
    success, img = cap.read()

    if(success == False):
        break
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    input_batch = transform(img).to(device)
    with torch.no_grad():
        prediction = midas(input_batch)
        prediction = torch.nn.functional.interpolate(
            prediction.unsqueeze(1),
            size=img.shape[:2],
            mode="bicubic",
            align_corners=False,
        ).squeeze()

    depth_map = prediction.cpu().numpy()
    depth_map = cv2.normalize(depth_map, None, 0, norm_type=cv2.NORM_MINMAX,
dtype=cv2.CV_64F)

    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

    depth_map = (depth_map*255).astype(np.uint8)
    depth_map = cv2.applyColorMap(depth_map , cv2.COLORMAP_MAGMA)
    result.write(depth_map)

    cv2.imshow('Image', img)
    cv2.imshow('Depth Map', depth_map)
    if cv2.waitKey(5) & 0xFF == 27:
        break
cap.release()
```