Sinteza novih pogleda na predmet iz dvodimenzionalnih slika korištenjem neuronskih mreža

Hrastnik, Iva

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 49

SINTEZA NOVIH POGLEDA NA PREDMET IZ DVODIMENZIONALNIH SLIKA KORIŠTENJEM NEURONSKIH MREŽA

Iva Hrastnik

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 49

SINTEZA NOVIH POGLEDA NA PREDMET IZ DVODIMENZIONALNIH SLIKA KORIŠTENJEM NEURONSKIH MREŽA

Iva Hrastnik

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 2. listopada 2023.

DIPLOMSKI ZADATAK br. 49

Pristupnica:	**Iva Hrastnik (0036515749)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	izv. prof. dr. sc. Matko Orsag

Zadatak: **Sinteza novih pogleda na predmet iz dvodimenzionalnih slika korištenjem neuronskih mreža**

Opis zadatka:

Potrebno je istražiti mogućnosti i postojeća rješenja za sintezu novih pogleda na predmet zasnovanih na Neural Radiance Fields metodi (NeRF). Posebnu je pažnju potrebno posvetiti rješenjima koja mogu stvoriti novi pogled uz različito osvjetljenje. Odabranom metodom treba stvoriti skup podataka za učenje korištenjem nekoliko slika predmeta, te stvoreni skup podataka iskoristiti za učenje neuronske mreže za detekciju upravo tog predmeta. U sklopu rada potrebno je analizirati točnost neuronske mreže za detekciju u ovisnosti o broju slika korištenih za sintezu novih pogleda odnosno stvaranje podataka za učenje.

Rok za predaju rada: 9. veljače 2024.

Želim zahvaliti svom mentoru izv. prof. dr. sc. Matku Orsagu na svoj pomoći, savjetima i prilikama još od preddiplomskog studija pa sve do danas. Posebne zahvale zaslužuje dr. sc. Frano Petric za kontinuirani trud i pomoć pri izradi Diplomskog rada tijekom cijelog semestra. Zahvaljujem svojoj obitelji i prijateljima na neizmjernoj podršci i razumijevanju svih ovih godina.

Sadržaj

1. Uvod

Velik napredak u područjima računalnog vida i umjetne inteligencije posljednjih nekoliko godina znatno je utjecao na percepciju grafičkih podataka kao i na rad s njima. Posebno zanimljiv izazov unutar ovog područja je sinteza novih pogleda na predmet iz dvodimenzionalnih slika, zadatak sa širokom primjenom u tehnologijama kao što su virtualna i proširena stvarnost te računalna grafika.

Generiranje novih neviđenih pogleda na predmet veoma je bitan element računalnog vida zato što računalu pruža uvid u vizualni svijet na način sličan ljudskom oku. Tradicionalne metode često se bore sa složenošću ovog zadatka, pogotovo kod prizora s različitim uvjetima osvjetljenja, preklapanjem predmeta ili složenim geometrijama. Ubrzanim razvojem umjetnih neuronskih mreža, a posebno pojavom *Neural Radiance Fields* metode (NeRF) dolazi do značajnog napretka i iznenađujuće dobrih rješenja ovog dugogodišnjeg problema [1].

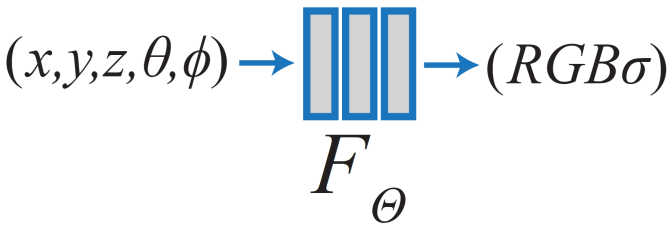
U navedenom radu Mildenhall i suradnici predstavljaju metodu čiji rezultati nadmašuju prethodna istraživanja, pružajući realistične sinteze pogleda na predmet posebno u scenama s kompliciranom geometrijom. Koristeći potpuno povezane (nekonvolucijske) duboke neuronske mreže za prepoznavanje složenih struktura svjetla unutar scene, metoda omogućuje generiranje novih pogleda na predmet kao nijedna prije. Okvir NeRF pokazao je izvanredan uspjeh u generiranju visokokvalitetnih slika podižući ljestvicu najsuvremenijih pristupa razumijevanju 3D prikaza.

Ovaj rad nastoji istražiti postojeća rješenja za sintezu novih pogleda na predmet zasnovanu na NeRF metodi. U narednim poglavljima potrebno je zadubiti se u teorijske temelje NeRF-a, komentirati postojeća slična rješenja i pokušati generirati što bolje 3D modele objekata učenjem neuronske mreže na skupu podataka od svega nekoliko slika.

2. Postojeća rješenja

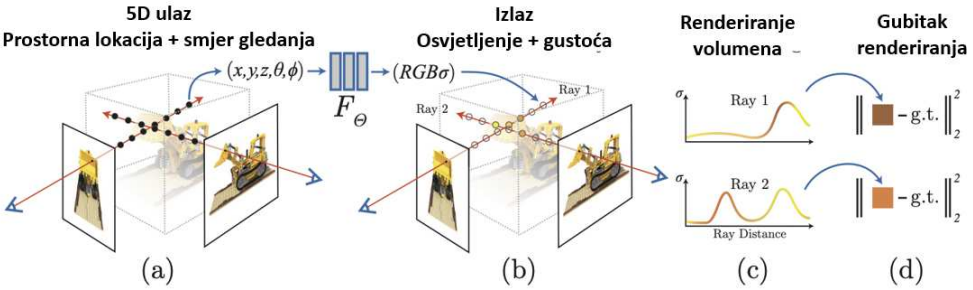
2.1. NeRF

Kroz navedeno istraživanje naziva *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis* [1] uvedena je NeRF metoda koja omogućuje generiranje novih visokokvalitetnih pogleda na predmet usprkos zamršenoj geometriji slike i raznim složenim strukturama svjetla. To je izvedeno razvijanjem algoritma koji opisuje 3D scenu koristeći se potpuno povezanom (nekonvolucijskom) dubokom mrežom. Za ulaz algoritma zadana je jedna kontinuirana 5D poza koja sadrži prostornu lokaciju (x, y, z) i smjer gledanja (θ, ϕ), a izlaz algoritma gustoća je volumena i emitirano osvjetljenje koje ovisi o smjeru gledanja na određenoj točki gledišta u prostoru 2.1.



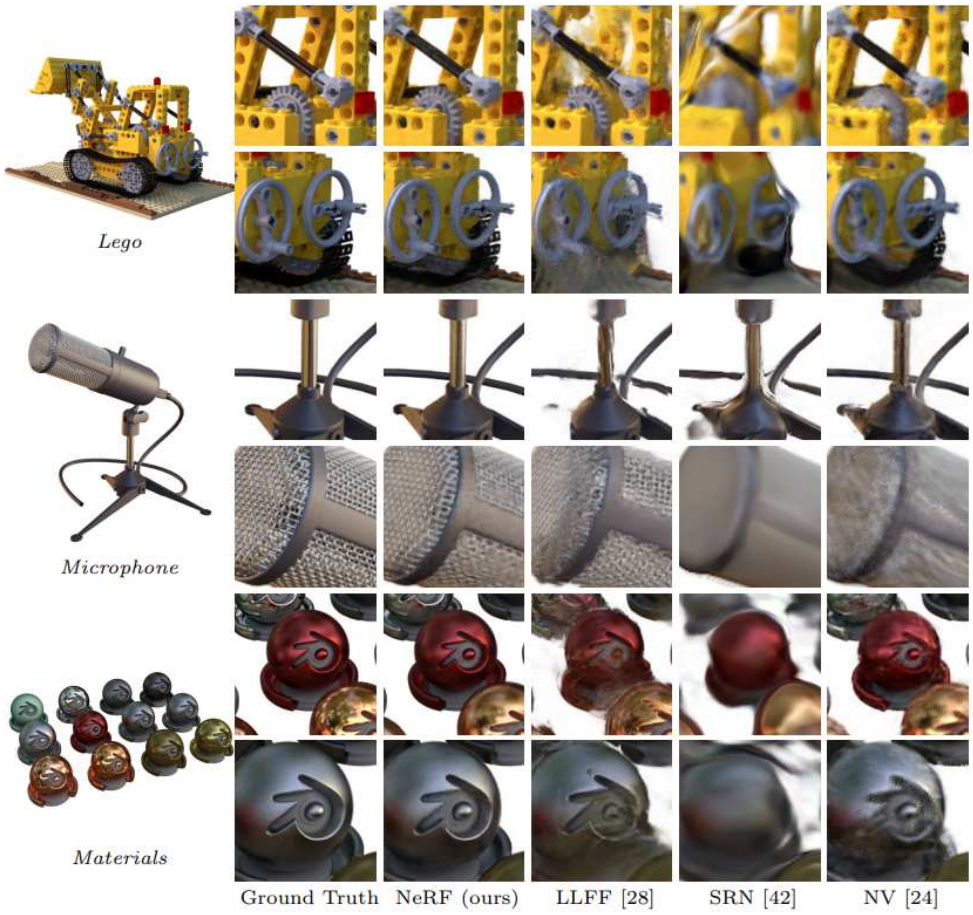
Slika 2.1. Prikaz ulaza i izlaza razvijenog algoritma [2]

Kreiranje novih pogleda na predmet u ovom radu ostvareno je pretraživanjem 5D poza duž zraka kamere i upotrebom klasičnih tehnika renderiranja volumena. Renderiranje volumena po prirodi je diferencijabilno što omogućuje izračun gradijenata bitnih za algoritme optimizacije kako bi iterativno ažurirali parametre modela i minimizirali razliku između generiranih pogleda i referentnih slika. Jedini potrebni ulaz algoritma za optimizaciju prikaza su slike zajedno s pripadajućim pozicijama kamere 2.2.



Slika 2.2. Vizualni prikaz NeRF metode s ulazima i izlazima [3]

Autori također opisuju kako učinkovito optimizirati NeRF za prikaz slika sa što kompliciranijom geometrijom. Na slici 2.3. prikazan je dio rezultata dobivenih na sintetičkom skupu podataka gdje se jasno vidi usporedba NeRF-a s prijašnjim metodama. Više o rezultatima kao i preostalim metodama može se pronaći u službenom tekstu rada [1].



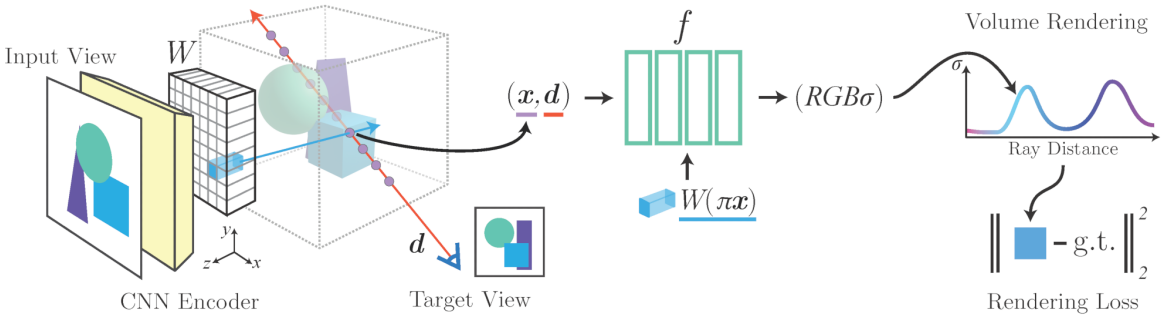
Slika 2.3. Usporedba rezultata NeRF metode s prijašnjim metodama na sintetičkom skupu podataka [1]

Kako je sintezu rezultata najbolje vidjeti iz video sadržaja, predlaže se pregled rezultata na službenoj web-adresi kako bi se istaknuo nevjerojatan uspjeh ove metode [4].

2.2. pixelNeRF

Ipak, NeRF zahtijeva mnogo ulaznih slika i optimizaciju prikaza svake scene zasebno. To uključuje izradu mnogo kalibriranih pogleda i značajno vrijeme računanja. S obzirom na to da je zadatak ovog rada istražiti mogućnosti sinteze novih pogleda na predmet koristeći samo nekoliko ulaznih slika, istraživanje mogućnosti produbljuje se nailaženjem na *pixelNeRF* [5], okvir za učenje koji predviđa sintezu kontinuiranog neuronskog prikaza scene uvjetovanu jednom ili nekoliko ulaznih slika.

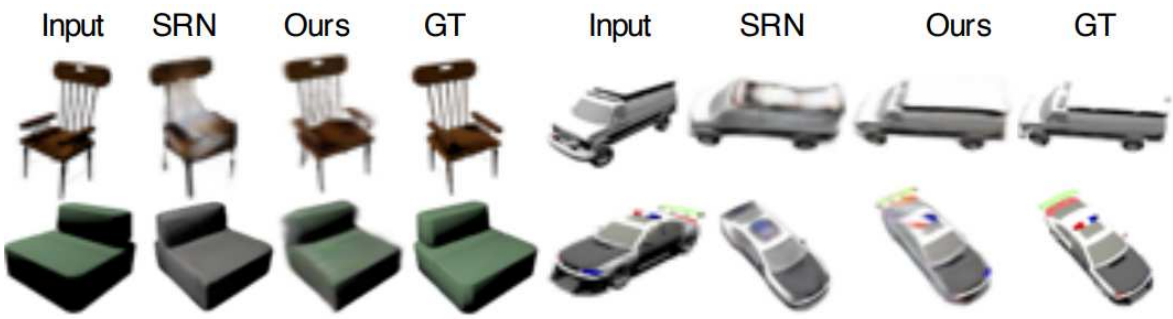
S pixelNeRF okvirom, kreće se korak dalje rješavanjem prethodno navedenih nedostataka. Za razliku od originalne NeRF mreže, koja ne koristi nikakve značajke slike, pixelNeRF za ulaz koristi prostorne značajke slike poravnate sa svakim pikselom. Ovaj okvir trenira se na slikama s više pogleda kako bi naučio prethodnu scenu i učinkovito izveo sintezu iz nekoliko ulaznih pogleda. Predstavlja se arhitektura čiji potpuno konvolucijski enkoder slike izvlači značajke poravnate s pikselima, a NeRF mreža služi za predviđanje boja i gustoće uvjetovanih prostornim značajkama slike. Iskorištavajući renderiranje volumena NeRF-a, model se može trenirati direktno iz slika bez korištenja 3D nadzora. Slika 2.4. predstavlja grafički prikaz ove metode.



Slika 2.4. Vizualni prikaz pixelNeRF metode s pripadajućim ulazima i izlazima [6]

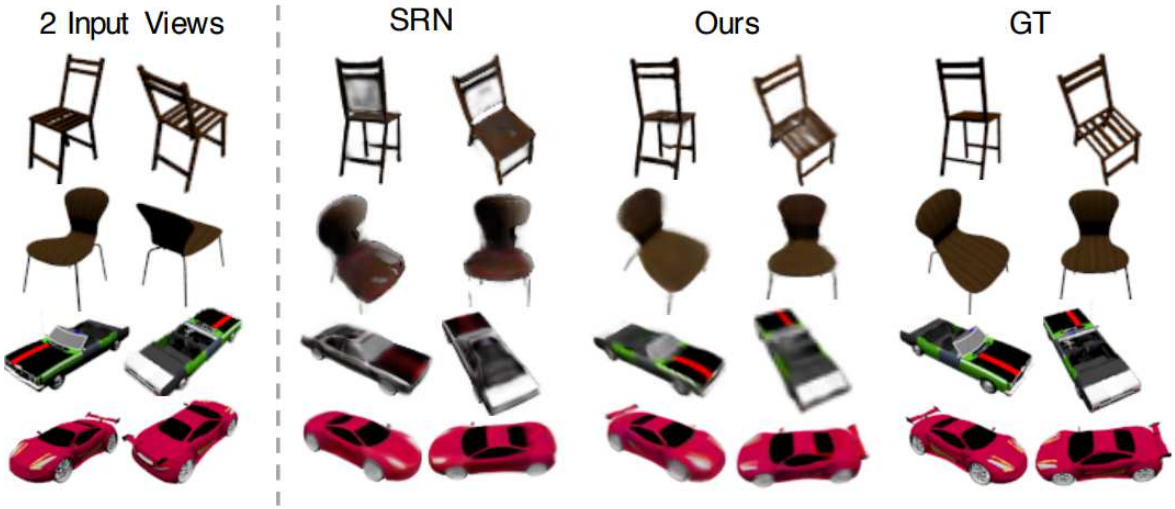
Opsežna testiranja pixelNeRF-a provedena su na ShapeNet skupu podataka [7] za sintezu novih pogleda na predmet koristeći samo jednu ulaznu sliku na cijelim neviđenim kategorijama slika. Dodatno je naglašena fleksibilnost samog pixelNeRF okvira demonstrirajući ga na ShapeNet scenama s više objekata i stvarnim scenama iz DTU skupa podataka. U svim slučajevima, pixelNeRF nadmašuje sva dotadašnja rješenja sinteze novih pogleda na predmet iz jedne slike.

Isto kao i kod NeRF metode, sve rezultate preporuča se pogledati na službenoj web-adresi pixelNeRF istraživanja [6] zato što ih se uvjerljivije može prikazati u video formatu. Ovdje su izloženi samo neki od tih rezultata. Na donjim slikama prikazana je usporedba ulazne slike s rezultatom Scene Representation Networks (SRN) [8] metode, izlaznom slikom pixelNeRF metode i *ground-truth* (GT) objektivno ispravnom referencom za rezultat.



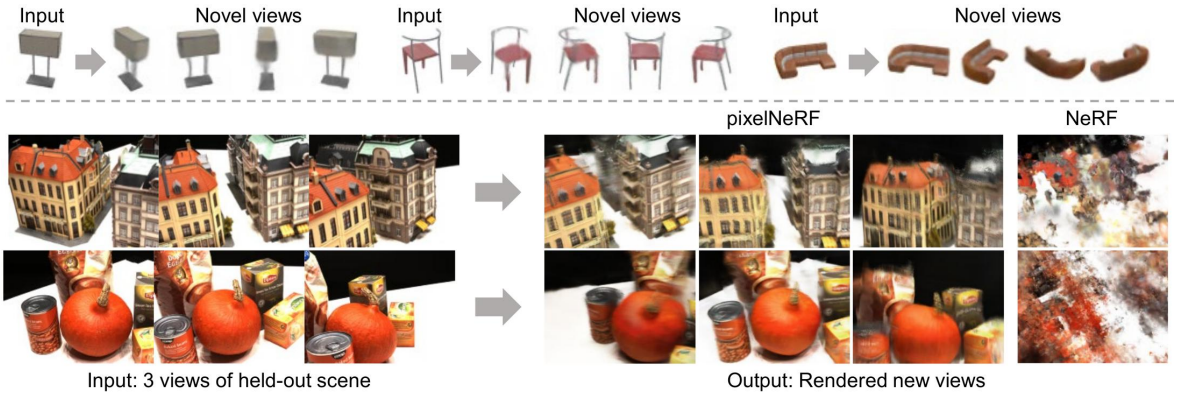
Slika 2.5. Usporedba rezultata pixelNeRF i SRN s jednom ulaznom slikom[5]

Na Slici 2.5. korištena je jedna slika kao ulaz, a na Slici 2.6. dvije ulazne slike.



Slika 2.6. Usporedba rezultata pixelNeRF i SRN metode s dvije ulazne slike[5]

Slika 2.7. pokazuje rad pixelNeRF metode naspram prethodno razvijene NeRF metode. S obzirom na to da se radi o ulaznom skupu od samo tri slike, poprilično je jasno koliko bolje pixelNeRF rekonstruira 3D scenu od NeRF-a u ovakvom slučaju.



Slika 2.7. Usporedba rezultata pixelNeRF i NeRF metode nad ulazom od tri slike [9]

Pokazan je veliki uspjeh u svijetu 3D rekonstrukcije objekata uzevši u obzir ovako dobre rezultate, a tako mali broj ulaznih slika. Upravo iz tog razloga, donesen je zaključak da je pixelNeRF metoda i više nego dostojna daljnjeg istraživanja u svrhu provođenja eksperimentalnog dijela ovog rada. U sljedećim poglavljima planirano je zadubiti se u razne mogućnosti koje pixelNeRF nudi, upute za pokretanje cjelokupnog modela i testiranje sinteze novih pogleda na predmet koristeći sintetičke podatke kao i fotografije predmeta.

3. Pokretanje pixelNeRF

Na službenom github repozitoriju [9] unutar datoteke *README.md* nalazi se detaljna uputa cijelog postupka potrebnog za pokretanje i korištenje pixelNeRF okvira. Za potrebe ovog istraživanja odabrani su dijelovi upute prikladni zadanom zadatku. Kako bi cijeli proces bio znatno olakšan, korištena je platforma *Google Colaboratory* [10], proizvod *Google Research-a*.

3.1. Google Colaboratory

Skraćenog naziva *Google Colab*, ova platforma omogućuje pisanje i izvršavanje python kôda putem preglednika, a posebno je pogodna za strojno učenje, analizu podataka i upotrebu u obrazovanju. S tehničke strane, *Colab* je zapravo *Jupyter* bilježnica [11] koja ne zahtijeva nikakav setup za korištenje, a pruža besplatan pristup računalnim resursima uključujući GPU-ove. Dodatna prednost je mogućnost povezivanja s *Google Drive* osobnom platformom za *cloud* pohranu i dijeljenje podataka. Naime, spajanjem na *Google Colab*, potrebno je svaki puta iznova učitati sve podatke potrebne za rad što može biti iscrpljujuć proces. Ovdje *Google Drive* postaje posebno koristan za direktno učitavanje ulaznih datoteka u *Colab* kao i spremanje izlaznih rezultata na *Drive*. Kreiranje ovakve bilježnice stvara pregledan format cijelog procesa istraživanja počevši od instalacije potrebnih alata do spremanja završnih rezultata na disk. Kôd se grupira u odlomke koji se mogu samostalno pokretati neovisno o ostatku kôda, a moguće je koristiti mnoštvo naredbi kako bi se obavljali različiti zadaci i upravljalo okolinom. Osim Python-a, često će biti korištene i *shell* naredbe ispred kojih se stavlja znak uskličnika (!*ls*) te naredbe za upravljanje datotekama ispred kojih je potrebno staviti znak postotka (%*cd*). Naredbe za upravljanje paketima !*pip install* i !*apt-get install* instaliraju Python pakete.

Prvi korak pri stvaranju *Colab* bilježnice je povezivanje osobnog *Google Diska* s bilježnicom. Za korištenje obje usluge potrebno je imati otvoren osobni *Google* račun i u prvi blok bilježnice zapisati sljedeće naredbe te ga pokrenuti.

Isječak kôda 3.1.1: Povezivanje Colab-a s Google Diskom

```
from google.colab import drive
drive.mount('/content/drive')
```

3.2. Aktivacija *pixelnerf* okruženja

Potrebno je pristati na povezivanje s osobnim Google Diskom i u sljedećem bloku bilježnice pozicionirati se u */drive/MyDrive* repozitorij. Na ovu lokaciju potrebno je dohvatiti i klonirati službeni github repozitorij *pixelNeRF* okvira [9], a zatim se i pozicionirati u *pixel-nerf* repozitorij. Ovo kloniranje potrebno je napraviti samo jedan put s obzirom na to da se repozitorij *pixel-nerf* sprema na Google Disk. Dakle, srednju liniju kôda potrebno je preskočiti svaki sljedeći put.

Isječak kôda 3.2.1: dohvaćanje pixelNeRF okvira u */drive/MyDrive* repozitorij

```
%cd drive/MyDrive/
!git clone https://github.com/sxyu/pixel-nerf.git
%cd pixel-nerf/
```

Sljedeći korak instalacija je *condacolab* [12] alata kreiranog specifično za korištenje *Conda* [13] alata u Google Colab-u. *Conda* je sustav koji pomaže u instalaciji, upravljanju i distribuciji softverskih paketa, a *condacolab* stoga omogućuje korištenje naredbe !*conda* u Colab-u.

Isječak kôda 3.2.2: instalacija *condacolab* alata

```
!pip install -q condacolab
import condacolab
condacolab.install()
```

Unutar dohvaćenog *pixel-nerf* github repozitorija nalazi se *environment.yml* datoteka [9] koja sadrži specifikacije potrebne za kreiranje novog *Conda* okruženja. Uz instalaciju *Python* verzije 3.8 ili više, potrebno je dohvatiti dodatne biblioteke *Scipy* [14], *Numpy* [15], *Pytorch* [16] i ostale navedene u sljedećem prikazu sadržaja datoteke. Koristeći *Pip* [17] obavlja se instalacija navedenih Python paketa.

Isječak kôda 3.2.3: sadržaj datoteke *environment.yml*

```
name: pixelnerf
channels:
  - pytorch
  - defaults
dependencies:
  - python>=3.8
  - pip
  - pip:
      - pyhocon
      - opencv-python
      - dotmap
      - tensorboard
      - imageio
      - imageio-ffmpeg
      - ipdb
      - pretrainedmodels
      - lpips
  - scipy
  - numpy
  - matplotlib
  - pytorch==1.6.0
  - torchvision==0.7.0
  - scikit-image==0.17.2
  - tqdm
```

Iz uputa unutar datoteke *README.md* [9] preuzet je sljedeći blok kôda. Nakon pozicioniranja u *drive/MyDrive/pixel-nerf/* repozitorij prvom naredbom, potrebno je kreirati novo *Conda* okruženje imena *pixelnerf* korištenjem zadane *environment.yml* datoteke, pa zatim napokon aktivirati pixelNeRF.

Isječak kôda 3.2.4: kreiranje okruženja i aktivacija pixelNeRF-a

```
%cd drive/MyDrive/pixel-nerf/
!conda env create -f environment.yml
!conda activate pixelnerf
```

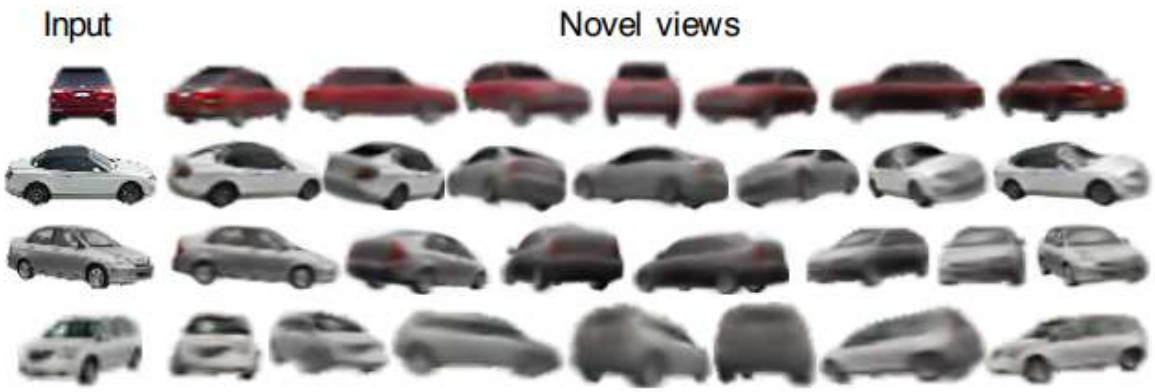
Također, korisno je provjeriti sadržaj datoteke *requirements.txt* ispisane u sljedećem isječku kôda. Alate koji nedostaju moguće je instalirati pokretanjem naredbe !*pip install {ime alata}*. Ili jednostavnije, pokrenuti naredbu !*pip -r requirements.txt*.

Isječak kôda 3.2.5: sadržaj datoteke *requirements.txt*

```
torch
torchvision
pretrainedmodels
pyhocon
imageio
opencv-python
imageio-ffmpeg
tensorboard
dotmap
numpy
scipy
scikit-image
ipdb
matplotlib
tqdm
lpips
```

3.3. Pokretanje pixelNeRF modela na fotografijama automobila

Za potrebe ovog istraživanja odabrano je pixelNeRF poglavlje *Real Car Images*. Ukratko, pixelNeRF model automobila direktno je primijenjen na stvarnim slikama automobila iz baze *Stanford cars dataset* [18]. Njihove rezultate istraživanja performansi modela na tom skupu podataka prikazuje Slika 3.1.



Slika 3.1. Rezultati pixelNeRF performansi na slikama stvarnih automobila [5]

Prema uputi [9] za pokretanje već treniranog pixelNeRF modela i generiranje video prikaza 3D modela automobila potrebno je preuzeti prethodno trenirane težine s poveznice [19]. Težine je potrebno raspakirati u repozitorij *pixelnerf/checkpoints/* tako da *pixelnerf/checkpoints/srn_car/pixel_nerf_latest* datoteka postoji.

Kako bi stvarne slike automobila bile spremne za korištenje, potrebno ih je dodatno obraditi uklanjanjem pozadine i normalizacijom. Na slike se primjenjuje modul za segmentaciju slike *PointRend* [20] koji kreira masku i okvir oko objekta na slici. Nakon uklanjanja pozadine korištenjem dobivene maske, slika se skalira tako da je centar graničnog okvira automobila poravnat s centrom slike i kraća stranica okvira iznosi 1/4 duljine bočne stranice slike.

Za pokretanje obrade slike, potrebno je prvo u direktorij *pixelnerf/input* staviti bilo koju fotografiju automobila. Zatim, s obzirom na to da obrada zahtijeva *PointRend* iz *detectron2* [21] biblioteke, potrebno ga je instalirati.

3.4. Instalacija *detectron2*

U *detectron2* github repozitoriju [22] nalaze se upute za lokalnu instalaciju, kao i za primjenu i instalaciju u *Google Colabu*. Iz upute za *Colab* [23] preuzeti su sljedeći isječci kôda i dodani u bilježnicu projekta.

Isječak kôda 3.4.1: preuzimanje i instalacija *detectron2*

```
!python -m pip install pyyaml==5.1
import sys, os, distutils.core

!git clone 'https://github.com/facebookresearch/detectron2'
dist = distutils.core.run_setup("./detectron2/setup.py")
!python -m pip install {
    ' '.join([f"'{x}'" for x in dist.install_requires])}
sys.path.insert(0, os.path.abspath('./detectron2'))
```

Pokretanjem isječka kôda 3.4.1 klonira se *detectron2* github repozitorij [22] te pomoću *pip* [17] alata instalira *detectron2* koji se zatim dodaje u *path*.

Isječak kôda 3.4.2: provjera instalacije *torch cuda* i *detectron2*

```
import torch, detectron2
!nvcc --version
TORCH_VERSION = ".".join(torch.__version__.split(".")[:2])
CUDA_VERSION = torch.__version__.split("+")[-1]
print("torch: ", TORCH_VERSION, "; cuda: ", CUDA_VERSION)
print("detectron2:", detectron2.__version__)
```

Isječak 3.4.2 služi za provjeru verzija *torch*, *cuda* i *detectron2*, a ujedno i njihovih uspješnih instalacija.

3.5. Obrada fotografije

S uspješnom instalacijom svih navedenih alata, aktivacijom pixelnerf okruženja, postavljenom fotografijom automobila u *pixelnerf/input* repozitorij i preuzetim težinama u *pixelnerf/checkpoints/srn_car* repozitorij može započeti obrada fotografije.

Slika 3.2. primjer je fotografije plavog automobila preuzete s web-adrese [24], pohranjene u *pixelnerf/input* repozitorij.

Slika 3.2. Ulazna fotografija plavog automobila [24]

Sva obrada događa se unutar *pixelnerf/scripts/preproc.py* python skripte koju je potrebno pokrenuti u sljedećem isječku kôda. Rezultat obrade normalizirana je slika u *PNG* formatu također pohranjena u *pixelnerf/input* repozitorij.

Isječak kôda 3.5.1: pokretanje obrade fotografije

```
!python scripts/preproc.py
```

Ukratko, ova skripta prima ulaznu fotografiju, koristeći *PointRend* provodi segmentaciju objekta i kreira izlaznu masku u *PNG* formatu. Slika se dodatno normalizira i sprema pod imenom <*ime_obrađene_slike*>*_normalize* kao prikaz objekta s bijelom pozadinom.

U skiptu je moguće uvesti proizvoljan broj slika, sve dok se one nalaze u *pixelnerf/input* repozitoriju. Slika 3.3. prikazuje izlazni rezultat skripte *preproc.py*, obrađenu verziju fotografije plavog automobila 3.2.

Slika 3.3. Obrađena slika plavog automobila

3.6. Provođenje evaluacije na stvarnoj fotografiji koristeći trenirani pixelNeRF model

PixelNeRF skripta *pixelnerf/eval/eval_real.py* koristi se za evaluaciju *NeRF* modela na stvarnim slikama. Korišten je prethodno trenirani *NeRF* model za sintezu novih prikaza ulazne slike iz različitih pozicija kamere.

Isječak kôda 3.6.1: pokretanje *eval_real.py* skripte

```
!python eval/eval_real.py
```

Generirani kadrovi pohranjuju se kao slike u repozitorij *pixelnerf/output/<ime_obrađene_slike>_normalize_frames*, a zatim se spajaju u videozapis pohranjen također u repozitorij *pixelnerf/output* pod imenom *<ime_obrađene_slike>_normalize_vid.mp4*.

Skripta prima proizvoljan broj slika iz *pixelnerf/input* repozitorija te za svaku zasebno provodi evaluaciju, generira kadrove i spaja ih u videozapis.

Slika 3.4. prikazuje svaki drugi generirani kadar nakon izvršavanja *eval_real.py* skripte. Prikazani kadrovi upravo predstavljaju rezultat sinteze novih pogleda na predmet dobiven korištenjem samo jedne ulazne slike.

Slika 3.4. Generirani novi pogledi na plavi automobil

4. Evaluacija pixelNeRF modela na nekoliko ulaznih slika različite prirode

Kako bi se ispitala kvaliteta modela, *eval_real.py* pokreće se na nekoliko različitih ulaznih slika. Slijedi stvarna fotografija automobila malo drugačijeg oblika 4.1.a i fotografija dječje igračke automobila 4.1.b

(a) Ulazna fotografija automobila [25]

(b) Ulazna fotografija dječje igračke automobila [26]

Slika 4.1.

Fotografije 4.1.a i 4.1.b obrađene su skriptom *preproc.py* i prikazane Slikama 4.2.a i 4.2.b Slika 4.2.c prikazuje generirane nove poglede na plavi automobil, dok Slika 4.2.d prikazuje generirane nove poglede na automobil igračku.

Nakon ovih modela slijede 4.3.a, 4.3.b i 4.3.c, sintetički generirane slike tri različita modela automobila. Njihove pripadajuće obrađene slike su 4.4.a, 4.4.b i 4.4.c Ovi modeli automobila generirani su u *Blenderu* [27], besplatnom *open-source* alatu za 3D računalnu grafiku. Slike iz Blendera imaju prednost nad stvarnim fotografijama zbog mogućnosti izrade slike iz bilo koje pozicije, udaljenosti i kuta postavljene zamišljene kamere.

(a) Obrađena slika automobila

(b) Obrađena slika igračke automobila

(c) Generirani novi pogledi na plavi automobil

(d) Generirani novi pogledi na automobil igračku

Slika 4.2.

(a) Ulazna slika crvenog automobila

(b) Ulazna slika bijelog automobila

(c) Ulazna slika žutog automobila

Slika 4.3.

(a) Obrađena slika crvenog automobila

(b) Obrađena slika bijelog automobila

(c) Obrađena slika žutog automobila

Slika 4.4.

(a) Generirani novi pogledi na crveni automobil

(b) Generirani novi pogledi na bijeli automobil

(c) Generirani novi pogledi na žuti automobil

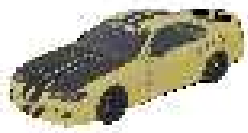
Slika 4.5.

5. Nadogradnja pixelNeRF metode

Prethodni pogledi na objekt automobila dobiveni *PixelNeRF* evaluacijom zadovoljavajući su, pogotovo uz činjenicu da su sintetizirani korištenjem samo jedne fotografije. Međutim, cilj rada je istražiti i proširiti mogućnosti ove metode što se izvodi provođenjem nekih preinaka.
Dobar primjer za daljnji rad slika je žutog sportskog automobila s mnogo detalja 4.3.c S obzirom na to da je slika generirana u *Blenderu* [27], njena obrada s pomoću *preproc.py* skripte nije nužna. To je od koristi jer osim što navedena skripta segmentira samo objekt na slici, ona ju normalizira i izreže što rezultira na oko vidljivom lošom rezolucijom obrađene slike. Prilikom kreiranja skupa slika modela žutog automobila u *Blenderu* postavljena je bijela pozadina i odgovarajuća dimenzija slike te daljnja obrada više nije potrebna, kao ni *pixelNeRF* skripta *preproc.py*.

Slika 5.1.a prikazuje sliku automobila izrađenu i obrađenu u *Blenderu* u usporedbi s istom slikom obrađenom pomoću skripte 5.1.b

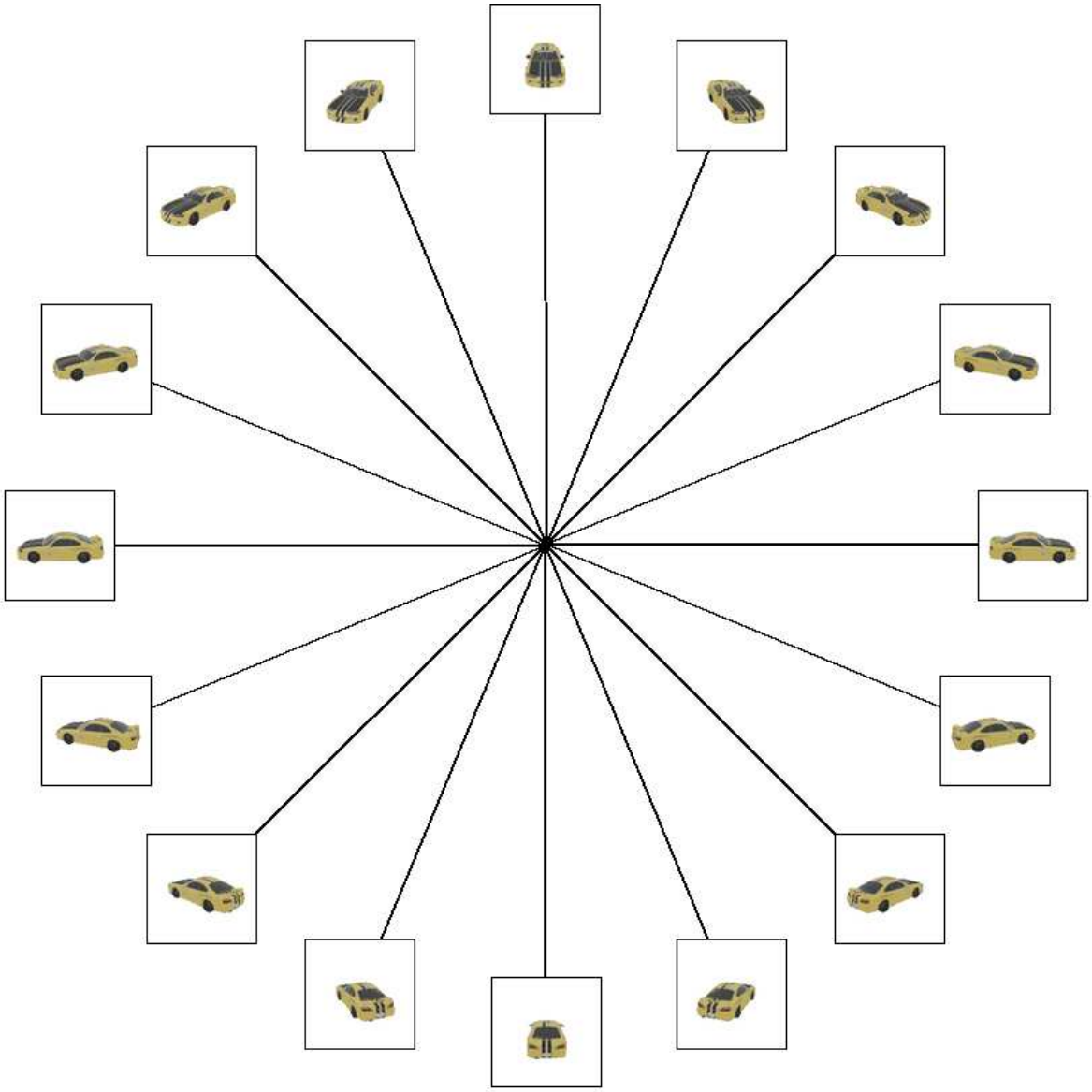
(a) Obrađena u *Blenderu*



(b) Obrađena u *preproc.py* skripti

Slika 5.1. Slika žutog automobila

Skup svih 16 ovako izrađenih ulaznih slika, svake pomaknute za 22.5 stupnjeva, prikazan je na Slici 5.2. Ove slike spremne su za korištenje u sljedećem koraku istraživanja, a to je pitanje kako *pixelNeRF* sintetizira nove poglede koristeći jednu, dvije, četiri, osam ili šesnaest ulaznih slika.



Slika 5.2. Ulazni skup slika žutog automobila

5.1. Izvođenje *eval_real.py* nad slikom poboljšane kvalitete

Usporedivši slike 5.1.a i 5.1.b uviđena je velika razlika u kvaliteti slike koja upućuje da bi model trebao ispasti bolje koristeći slike obrađene samo u Blenderu bez korištenja *preproc.py* skripte. Ova hipoteza potvrđena je izvođenjem kôda *eval_real.py* nad obje slike 5.3.a i 5.3.b Slike 5.4.a i 5.4.b prikazuju rezultate.

(a) Obrađena u *Blenderu*

(b) Obrađena u *preproc.py* skripti

Slika 5.3. Slika žutog automobila

(a) Generirani novi pogledi na automobil obrađen u *Blenderu* 5.3.a

(b) Generirani novi pogledi na automobil obrađen u *preproc.py* skripti 5.3.b

Slika 5.4.

5.2. Izmjena *eval_real.py* skripte za ulaz više slika

Trenutna skripta može primiti više ulaznih slika, za svaku sliku zasebno provodeći cijeli proces izrade novih pogleda na automobil. Takva skripta za N slika napravit će N modela baziranih na jednoj slici što trenutno nije željeno ponašanje.

Ideja je iskoristiti postojeću *pixelnerf/eval/eval_real.py* skriptu i preraditi je tako da uzima u obzir uneseni broj slika snimljenih iz različitih kutova postavljene kamere. Različite poglede iz kamere potrebno je poslati u obradu za isti model kako bi, očekivano, rezultat sinteze novih pogleda bio što jasniji i čišći.

Pretpostavka je da bi sa što većim brojem slika rezultat trebao biti sve bolji. Vrlo važan parametar sad postaje i poza kamere na svakoj slici kako bi model uspješno smjestio slike u prostoru i njihovu orijentaciju protumačio na ispravan način.

Zbog jednostavnosti izvedbe, svakoj slici u imenu već je dodijeljen kut pozicije kamere u obliku *angle<kutna vrijednost>*.

Nova dorađena skripta za svaku sliku treba izračunati *cam_pose* koristeći sljedeći isječak kôda. Za početak, u skripti su parametri udaljenosti od predmeta *radius* i povišenja kamere *elevation* postavljeni na 1. Varijablu kuta kamere, kao što je navedeno, skripta izvlači iz imena slike.

Postojeća *for* petlja iterira kroz ulazne slike, otvara svaku sliku koristeći *Python* biblioteku *PIL* [28], pretvara ju u RGB format, postavlja joj veličinu na zadanu *in_sz* i na kraju pretvara dobivenu sliku u tenzor.

Svaka slika zatim se dodaje u listu *images*, a poza kamere računa se iz parametara *angle*, *radius* i *elevation* pa također dodaje u listu *cam_poses*. Time su sve slike i pripadajuće im poze spremljene u liste po kojima se nadalje iterira. Tako se slaže kombinirana slika i kombinirana poza kamere, dva parametra koja je zapravo i bio glavni cilj preraditi prije nego ih se preda mreži. Ovime je postignuto spajanje proizvoljnog broja slika i pripadajućih im poza kamere u jedan model automobila.

Isječak kôda 5.2.1: isječak dijela skripte *eval_real.py* kojeg je potrebno izmijeniti

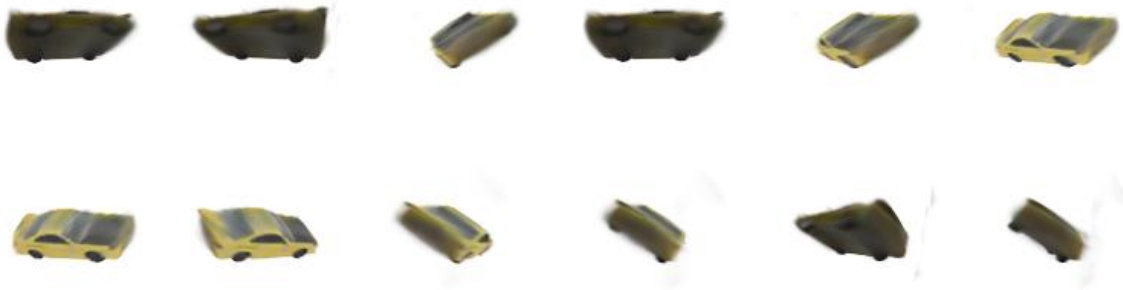
```
elevation = 1.0
radius = 1.0

_coord_from_blender = util.coord_from_blender()
image_to_tensor = util.get_image_to_tensor_balanced()

images = []
cam_poses = []

with torch.no_grad():
    for i, image_path in enumerate(inputs):

        image = Image.open(image_path).convert("RGB")
        image = T.Resize(in_sz)(image)
        image = image_to_tensor(image).to(device=device)

        images.append(image)
        angle = -<angle_from_image>
        cam_poses.append(_coord_from_blender @
                    util.pose_spherical(angle, elevation, radius))

    combined_image = torch.cat(
                [img.unsqueeze(0) for img in images], dim=0)
    merged_poses = torch.cat(
                [pose.unsqueeze(0) for pose in cam_poses], dim=0)

    net.encode(
        combined_image.unsqueeze(0),
        merged_poses.unsqueeze(0).to(device=device),
        focal.to(device=device),
    )
```

5.3. Rezultati nadograđene *eval_real.py* skripte za jednu ulaznu sliku

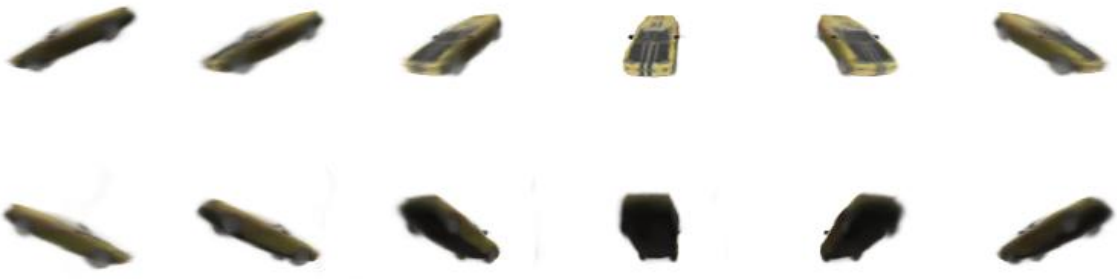
(a) Bočna slika

(b) prednja slika

Slika 5.5. Ulazne slike za dva različita pogleda



(a) Generirani novi pogledi na automobil koristeći jednu bočnu ulaznu sliku 5.5.a



(b) Generirani novi pogledi na automobil koristeći jednu prednju ulaznu sliku 5.5.b

Slika 5.6.

5.4. Rezultati nadograđene *eval_real.py* skripte za dvije ulazne slike

(a) **(b)**

Slika 5.7. Ulazne slike žutog automobila s povišenim kutom kamere, prvi slučaj

Slika 5.8. Novi pogledi na žuti automobil koristeći ulazne slike 5.7.

(a) **(b)**

Slika 5.9. Ulazne slike žutog automobila s povišenim kutom kamere, drugi slučaj

Slika 5.10. Novi pogledi na žuti automobil koristeći ulazne slike 5.9.

(a) **(b)**

Slika 5.11. Ulazne slike žutog automobila s povišenim kutom kamere, treći slučaj

Slika 5.12. Novi pogledi na žuti automobil koristeći ulazne slike 5.11.

(a) **(b)**

Slika 5.13. Ulazne slike žutog automobila s horizontalnim kutom kamere, prvi slučaj

Slika 5.14. Novi pogledi na žuti automobil koristeći ulazne slike 5.13.

(a) **(b)**

Slika 5.15. Ulazne slike žutog automobila s horizontalnim kutom kamere, drugi slučaj

Slika 5.16. Novi pogledi na žuti automobil koristeći ulazne slike 5.15.

(a) **(b)**

Slika 5.17. Ulazne slike žutog automobila s horizontalnim kutom kamere, treći slučaj

Slika 5.18. Novi pogledi na žuti automobil koristeći ulazne slike 5.17.

(a) **(b)**

Slika 5.19. Ulazne slike plavog automobila s horizontalnim kutom kamere, prvi slučaj

Slika 5.20. Novi pogledi na plavi automobil koristeći ulazne slike 5.19.

(a) **(b)**

Slika 5.21. Ulazne slike plavog automobila s horizontalnim kutom kamere, drugi slučaj

Slika 5.22. Novi pogledi na plavi automobil koristeći ulazne slike 5.21.

5.5. Rezultati dobiveni koristeći četiri ulazne slike

(a) Lijeva bočna slika **(b)** Desna bočna slika **(c)** Prednja slika **(d)** Stražnja slika

Slika 5.23. Četiri ulazne slike

Slika 5.24. Generirani novi pogledi na žuti automobil koristeći četiri ulazne slike 5.23.

(a) Lijeva bočna slika **(b)** Desna bočna slika **(c)** Prednja slika **(d)** Stražnja slika

Slika 5.25. Četiri ulazne slike

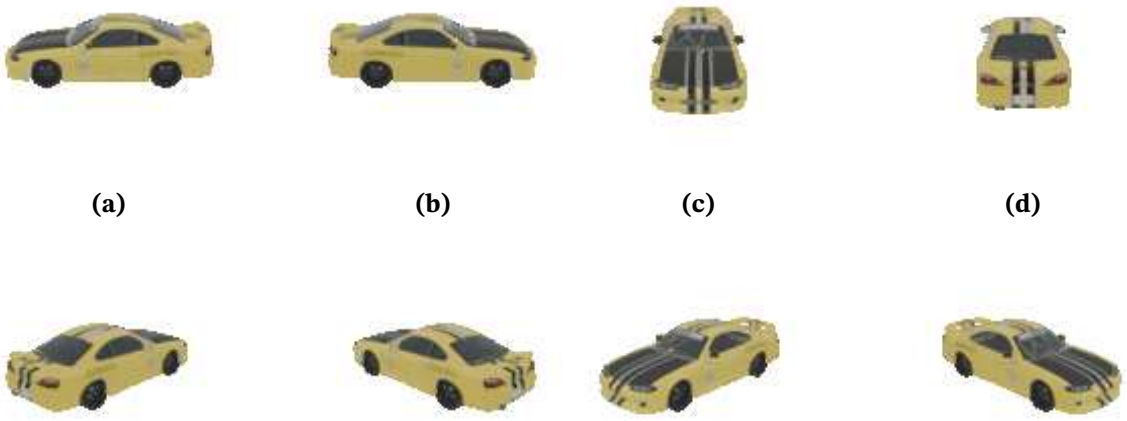
Slika 5.26. Generirani novi pogledi na žuti automobil iz nižeg kuta koristeći ulazne slike 5.25.

(a) Lijeva bočna slika **(b)** Desna bočna slika **(c)** Prednja slika **(d)** Stražnja slika

Slika 5.27. Četiri ulazne slike

Slika 5.28. Generirani novi pogledi na plavi automobil koristeći ulazne slike 5.27.

5.6. Rezultati dobiveni koristeći osam ulaznih slika

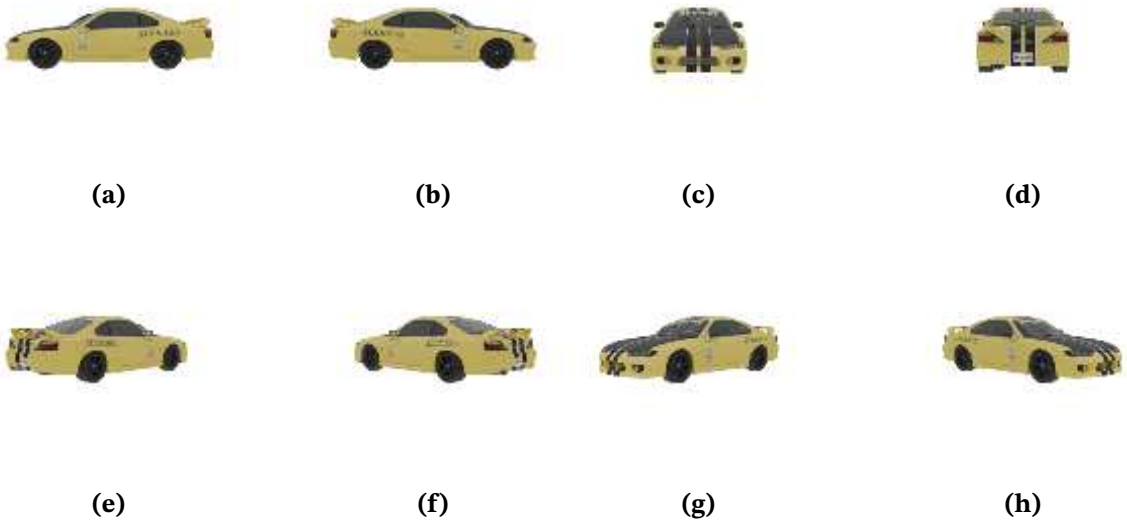


(a) **(b)** **(c)** **(d)**

(e) **(f)** **(g)** **(h)**

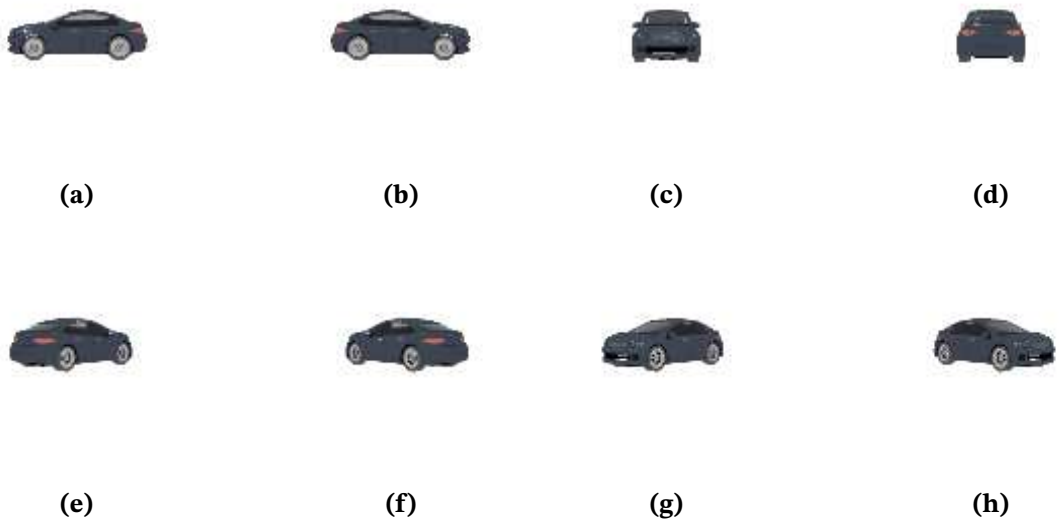
Slika 5.29. Osam ulaznih slika

Slika 5.30. Generirani novi pogledi na žuti automobil koristeći osam ulaznih slika 5.29.



Slika 5.31. Osam ulaznih slika

Slika 5.32. Generirani novi pogledi na žuti automobil iz nižeg kuta kamere koristeći osam ulaznih slika 5.31.



Slika 5.33. Osam ulaznih slika

Slika 5.34. Generirani novi pogledi na plavi automobil koristeći osam ulaznih slika 5.33.

5.7. Rezultati dobiveni koristeći šesnaest ulaznih slika

Šesnaest ulaznih slika nalazi se na prethodnom grafičkom prikazu 5.2.

Slika 5.35. Generirani novi pogledi na žuti automobil koristeći šesnaest ulaznih slika 5.2.

Slika 5.36. Generirani novi pogledi na žuti automobil s nižeg kuta kamere koristeći šesnaest ulaznih slika

Slika 5.37. Generirani novi pogledi na plavi automobil koristeći šesnaest ulaznih slika

6. YOLO detekcija automobila u rezultatima

Koristeći YOLO [29] platformu za detekciju objekta provedena je analiza kvalitete rekonstrukcije automobila. Rezultati analize pokazuju da kvaliteta detekcije više ovisi o tipu korištenog modela automobila, nego o broju korištenih slika. Unatoč tomu, YOLO dobro prepoznaje model plavog automobila, pa se na modelima koji primaju četiri, osam i šesnaest ulaznih slika plavog automobila može provesti analiza. Tablica 6.1. prikazuje ovisnost odziva o broju ulaznih slika. Iz ovih rezultata može se zaključiti da s rastom broja ulaznih slika pada kvaliteta rekonstrukcije modela što potvrđuje da je *pixelNeRF* kreiran za manji broj ulaznih slika.

Tablica 6.1. Ovisnost odziva o broju ulaznih slika za model plavog automobila

Broj ulaznih slika	Točne detekcije	Pogrešne detekcije	Odziv
4	21	3	0.8750
8	19	5	0.7917
16	15	9	0.6250

7. Analiza rezultata

Usporedbom na Slici 5.4. zaključeno je da skripta *pixelnerf/scripts/preproc.py* iz *pixelNeRF* repozitorija može poslužiti za obradu realne fotografije automobila, ali definitivno nije najbolja praksa s obzirom na to da uveliko utječe na kvalitetu slike obrađujući ju na ovaj način. To je potvrđeno izradom novih pogleda na automobil koristeći dvije slike istog automobila. Slike automobila generirane su u *Blenderu* s bijelom pozadinom, prva je zatim obrađena 5.3.a koristeći navedenu *preproc.py* skriptu i takva predana kao ulaz u *pixelnerf/eval/eval_real.py* skriptu. Druga slika 5.3.b jednostavno je predana u navedenu *eval_real.py* skriptu bez daljnje obrade. Generirani novi pogledi na automobil koristeći neobrađenu ulaznu sliku 5.4.a izgledaju jasnije i prikazuju više detalja od onih koristeći dodatno obrađenu sliku 5.4.b. Stoga je za ispitivanje modela na većem broju ulaznih slika korišten skup slika izrađenih u *Blenderu* bez dodatne obrade.

Prije pokretanja modela na više ulaznih slika, zanimljivo je pokrenuti ga na jednoj bočnoj slici 5.5.a i jednoj prednjoj 5.5.b. Kao i očekivano, ova dva modela ispadaju potpuno drugačije zato što modelu koji koristi potpuno bočnu sliku 5.6.a nedostaju ikakve informacije o izgledu prednje i zadnje strane automobila, a modelu koji koristi potpuno prednju sliku 5.6.b nedostaju informacije o bočnoj strani automobila i pojam dužine automobila.

Model je s ulazne dvije slike pokrenut na tri različita skupa slika. Prvi skup slika generirane su slike žutog automobila s povišenjem kuta pogleda kamere korištene do sad 5.2. Drugi skup slika generirane su slike istog žutog automobila, no u ovom slučaju pogled kamere je horizontalan. Treći skup slika sadrži slike generiranog plavog automobila nešto drugačijeg oblika čiji pogled kamere je također horizontalan. Rezultati prvog 5.8., drugog 5.10. i trećeg 5.12. slučaja na žutom automobilu s pogledom odozgora prikazuju da model najbolje generira nove poglede na predmet koristeći dvije bočne slike, dok su

pogledi u drugom i trećem slučaju znatno lošiji. Ovdje se može zaključiti da bočni prikaz automobila daje modelu puno više relevantnih informacija nego prednji ili zadnji, koji ga u ovom slučaju i zbune budući da se radi o automobilu s mnogo detalja na prednjem i zadnjem kraju. Također, bitno je primijetiti da su pogledi na automobil koristeći dvije bočne ulazne slike 5.8. generirani znatno bolje od pogleda generiranih koristeći jednu bočnu ulaznu sliku 5.6.a Tako je isto važno primijetiti da rezultati dobiveni koristeći jednu prednju i jednu bočnu sliku 5.10., iako manje jasni od rezultata 5.8., opet bolje prikazuju stvarni automobil od zasebnih rezultata za istu ulaznu bočnu sliku 5.5.a i zasebnih rezultata za istu prednju sliku 5.6.b

Rezultati na slikama žutog automobila s horizontalnim pogledom kamere 5.14. i 5.16. dokazuju da je model dosta osjetljiv na kut pogleda kamere, pa generirani pogledi zasigurno bolje izgledaju u ovakvom slučaju za razliku od povišenih pogleda na automobil. Slika 5.18. prikazuje generirane nove poglede s ulaznom prednjom i zadnjom slikom 5.17. Ovi pogledi očekivano ispadaju najgore zbog već spomenutog problema nedostatka ikakvih informacija o bočnoj strani i duljini automobila. Slika 5.22. pokazuje generirane nove poglede na plavi automobil zadan ulaznim slikama 5.21., a zanimljivo je primijetiti da generirani pogledi na plavi automobil ispadaju dosta loše s ulaznim slikama 5.19. koje nisu pod pravim kutom 5.20.

Rezultat sinteze novih pogleda na automobil koristeći sve četiri ulazne slike pod pravim kutom 5.23. pokazuje se kao najbolji rezultat do sad. Ovi generirani pogledi na automobil su jednako dobri iz svakog kuta pogleda kamere, no kod žutog automobila s povišenom kamerom rezultati 5.24. ispadaju mutniji nego kod istog automobila s horizontalnim pogledom kamere 5.26. Odlični pogledi generiraju se i za plavi automobil s horizontalnim pogledom kamere 5.28.

Međutim, povećanjem broja ulaznih slika s četiri na osam 5.29., rezultat modela žutog auutomobila znatno se pogoršao 5.30., a model s ulaznih šesnaest slika stvara još gore poglede od prethodnog 5.35. Kod druga dva automobila rezultati za osam ulaznih slika znatno su bolji od prethodnih, a prikazuju ih slike 5.32. i 5.34. Slike 5.36. i 5.37. prikazuju generirane nove poglede za šesnaest ulaznih slika koji su, očekivano, najbolji do sad. Ova činjenica dodatno ukazuje na osjetljivost modela na kut kamere.

8. Zaključak

Nakon provedenog istraživanja nad *Neural Radiance Fields* [1] metodom koja koristeći potpuno povezane duboke neuronske mreže za prepoznavanje složenih struktura svjetla unutar scene omogućuje generiranje novih pogleda na predmet, može se zaključiti da ova metoda zaista postiže nevjerojatne rezultate.

Specifično, korištenjem pixeNeRF [5] okvira za učenje uz pomoć kôda i dokumentacije dostupnih na službenom github repozitoriju [9], provedeno istraživanje nad skupom realističnih fotografija automobila kao i sintetičkim skupom slika pokazalo je očekivano dobre rezultate.

Ova metoda uspijeva generirati nove poglede na automobil, pa samim time i njegov 3D model, koristeći samo jednu ulaznu sliku. Istraživanje je prošireno dodavanjem mogućnosti ulaza proizvoljnog broja slika stapajući ih zajedno u tenzor podatke korištene za sintezu novih pogleda na predmet. Na odabranom sintetički generiranom skupu slika žutog automobila 5.2. s mnogo detalja i precizno definiranim pozicijama kamere, testiranje je provedeno za dvije, četiri, osam i šesnaest različitih ulaznih slika. Isto je provedeno za slike tog automobila iz različitog kuta pogleda kamere, kao i za novi sintetički generiran skup slika plavog automobila nešto drugačijeg oblika.

Ovo testiranje dovodi do zaključka da su generirani novi pogledi na automobil bolji s povećanjem broja ulaznih slika u slučaju kamere koja horizontalno gleda na objekt. Model žutog automobila s povišenim pogledom kamere loše provodi sintezu novih pogleda na automobil u slučaju osam i šesnaest ulaznih slika. Iako su modeli s više slika kvalitetniji, pokazani su odlični rezultati koristeći već četiri ulazne slike.

Literatura

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, i R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis", u *ECCV*, 2020.

[2] Ben Mildenhall and Pratul P. Srinivasan and Matthew Tancik and Jonathan T. Barron and Ravi Ramamoorthi and Ren Ng, https://uploads-ssl.webflow.com/51e0d73d83d06baa7a00000f/5e700a025ff238947d682a1f_pipeline_website-03.svg, [Online; Accessed: February 2024].

[3] ——, https://uploads-ssl.webflow.com/51e0d73d83d06baa7a00000f/5e700ef6067b43821ed52768_pipeline_website-01-p-800.png, [Online; Accessed: February 2024].

[4] Matthew Tancik, https://www.matthewtancik.com/nerf, [Online; Accessed: February 2024].

[5] A. Yu, V. Ye, M. Tancik, i A. Kanazawa, "pixelNeRF: Neural radiance fields from one or few images", u *CVPR*, 2021.

[6] Alex Yu, https://alexyu.net/pixelnerf/, [Online; Accessed: February 2024].

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, i F. Yu, "ShapeNet: An Information-Rich 3D Model Repository", Stanford University — Princeton University — Toyota Technological Institute at Chicago, teh. izv. arXiv:1512.03012 [cs.GR], 2015.

[8] V. Sitzmann, M. Zollhöfer, i G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations", 2020.

[9] Alex Yu, https://github.com/sxyu/pixel-nerf, [Online; Accessed: February 2024].

[10] Google Research, https://colab.research.google.com/, [Online; Accessed: February 2024].

[11] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, i C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows", u *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides i B. Schmidt, Ur. IOS Press, 2016., str. 87 – 90.

[12] Python package index - pypi, condacolab tool. [Mrežno]. Adresa: https://pypi.org/project/condacolab/

[13] "Anaconda software distribution", 2020. [Mrežno]. Adresa: https://docs.anaconda.com/

[14] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, i SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", *Nature Methods*, sv. 17, str. 261–272, 2020. https://doi.org/10.1038/s41592-019-0686-2

[15] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, i T. E. Oliphant, "Array programming with NumPy", *Nature*, sv. 585, br. 7825, str. 357–362, rujan 2020. https://doi.org/10.1038/s41586-020-2649-2

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, i S. Chintala, "Pytorch: An

imperative style, high-performance deep learning library", u *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019., str. 8024–8035. [Mrežno]. Adresa: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[17] Python package index - pypi. [Mrežno]. Adresa: https://pypi.org/

[18] J. D. Jonathan Krause, Michael Stark i L. Fei-Fei, "3d object representations for fine-grained categorization." u *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13),*, Sydney, Australia, 2013.

[19] Pretrained weights - pixelNeRF, https://drive.google.com/file/d/1UO_rL201guN6euoWkCOn-XpqR2e8o6ju/view?usp=sharing, [Online; Accessed: February 2024].

[20] A. Kirillov, Y. Wu, K. He, i R. Girshick, "PointRend: Image segmentation as rendering", 2019.

[21] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, i R. Girshick, "Detectron2", https://github.com/facebookresearch/detectron2, 2019.

[22] Facebook research, https://github.com/facebookresearch/detectron2, [Online; Accessed: February 2024].

[23] ——, https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5, [Online; Accessed: February 2024].

[24] Image resource, https://www.bmw.co.uk/content/dam/bmw/common/all-models/m-series/i4-m50/2021/onepager/bmw-i4-m50-onepager-gallery-impressions-wallpaper-04.jpg, [Online; Accessed: February 2024].

[25] ——, https://www.motor1.com/photos/740533/hyundai-i30-n-2020/#5200914_hyundai-i30-n-2020, [Online; Accessed: February 2024].

[26] ——, https://st.depositphotos.com/1455321/1564/i/450/depositphotos_15645257-stock-photo-yellow-toy-car.jpg, [Online; Accessed: February 2024].

[27] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Mrežno]. Adresa: http://www.blender.org

[28] P. Umesh, "Image processing in python", *CSI Communications*, sv. 23, 2012.

[29] J. Redmon, S. Divvala, R. Girshick, i A. Farhadi, "You only look once: Unified, real-time object detection", u *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016., str. 779–788. https://doi.org/10.1109/CVPR.2016.91

Sažetak

Sinteza novih pogleda na predmet iz dvodimenzionalnih slika korištenjem neuronskih mreža

Iva Hrastnik

Provedeno je istraživanje *Neural Radiance Fields* metode i *pixelNeRF* okvira za učenje koji korištenjem neuronskih mreža provodi sintezu novih pogleda na predmet iz jedne ili nekoliko ulaznih slika. Korišten je *PixelNeRF* model treniran na *ShapeNet* skupu slika automobila prilagođen ulazu stvarnih fotografija automobila. Provedeno je testiranje modela s jednom do nekoliko ulaznih slika koristeći fotografije različitih automobila, kao i sintetički generiran skup slika. Kvaliteta novih pogleda na predmet očekivano raste s povećanjem broja ulaznih slika automobila, no već odlične rezultate prezentira i sinteza pogleda iz dvije te četiri ulazne fotografije.

Ključne riječi: Neuronska polja zračenja; NeRF; PixelNeRF; Neuronska mreža; Sinteza novih pogleda; Računalni vid; Pozicija kamere; 3D rekonstrukcija; Osvjetljenje; 3D Model; Rezolucija slike;

Abstract

Generating new perspectives of an object from two-dimensional images using neural networks

Iva Hrastnik

Research was conducted on the *Neural Radiance Fields* method and the *pixelNeRF* learning framework, which uses neural networks to synthesize new views of the subject using one or several input images. A *PixelNeRF* model trained on a *ShapeNet* set of car images and adapted to the input of real car photos was used. The model was tested with one to several input images using photos of different cars as well as a synthetically generated set of images. As expected, the quality of new views of the subject increases with the number of input car images, but excellent results are also presented by the synthesis of views from only two or four input photos.

Keywords: Neuronal radiance fields; NeRF; PixelNeRF; Neural network; Synthesis of new views; Computer vision; Camera pose; 3D reconstruction; Illumination; 3D Model; Image resolution;