MONOCULAR
DEPTH ESTİMATİON

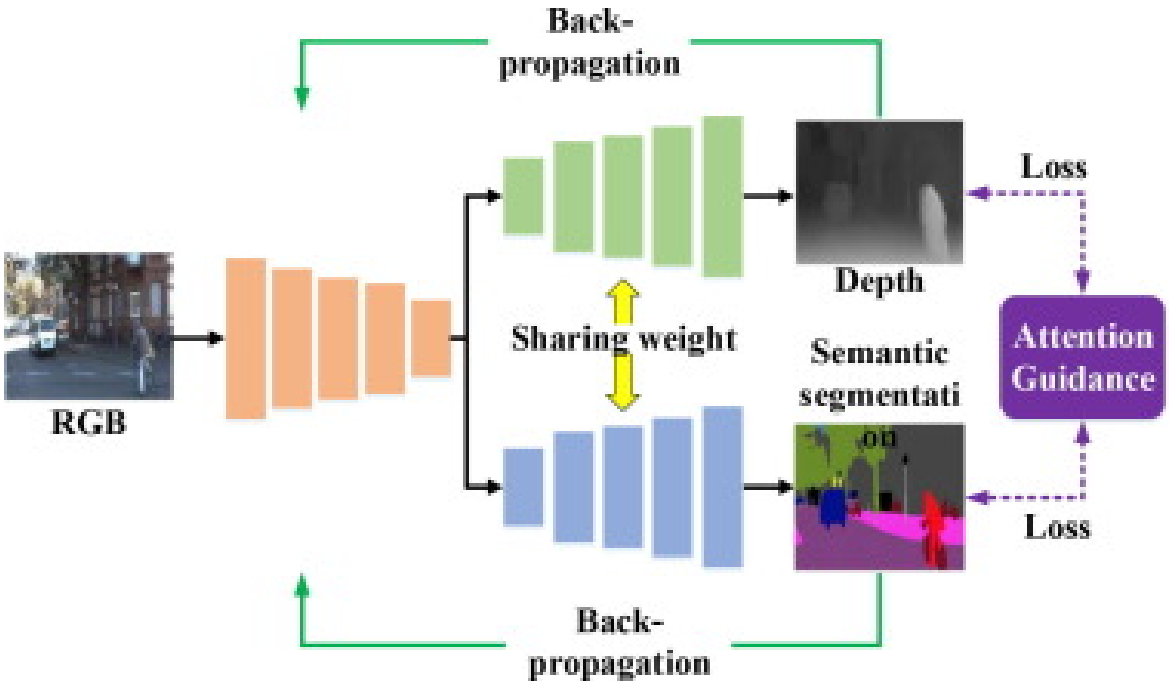
YONGATEK

İÇİNDEKİLER

PROJENİN AMACI

Derinlik tahmini, 2B görüntülerden sahne geometrisi çıkarmaya yönelik çok önemli bir adımdır. Monoküler derinlik tahminindeki amaç, girdi olarak yalnızca tek bir RGB görüntüsü verilen her pikselin derinlik değerini veya çıkarım yapan derinlik bilgisini tahmin etmektir. Bu örnek, bir convnet ve basit kayıp fonksiyonları ile bir derinlik tahmin modeli oluşturmaya yönelik bir yaklaşımdır.

Bu projede Midas makine öğrenimi modelini kullanılmaktadır. *Midas*, rastgele bir girdi görüntüsünden derinliği tahmin eder. Derinlik bilgisi içeren çeşitli veri kümeleri, ölçek ve sapma açısından uyumlu değildir. Bunun nedeni, stereo kameralar, lazer tarayıcılar ve ışık sensörleri dahil olmak üzere ölçüm araçlarının çeşitliliğidir. *Midas*, bu farklılıkları yok eden, böylece uyumluluk sorunlarını ortadan kaldıran ve aynı anda eğitim için birden fazla veri kümesinin kullanılmasına izin veren yeni bir loss işlevi sunar.



PROJEDE KULLANILAN ORTAMLAR

UBUNTU 18.04

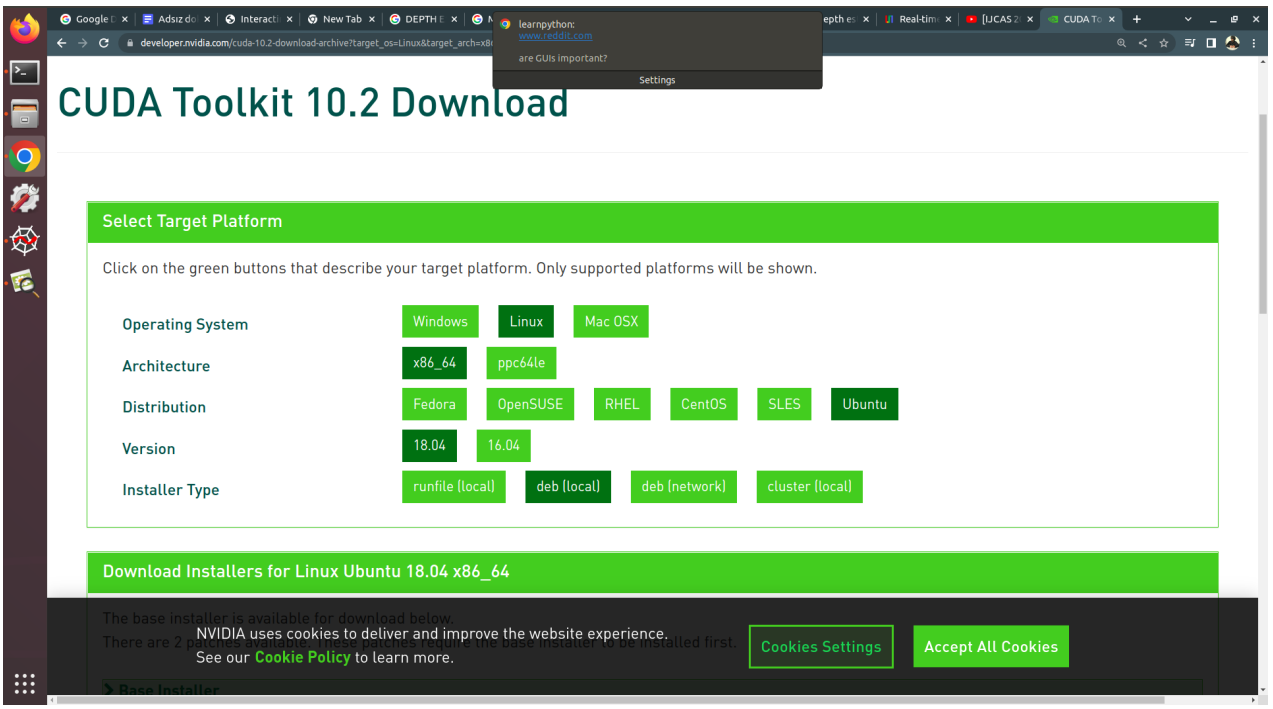
- **Kurulum Videosu**

🔗 https://www.youtube.com/watch?v=YvfhHCMA-yY

CUDA 10.2

- **İnternet Adresi:**

🔗 https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html



CUDA KURULUM:

```
$ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin
```

```
$ sudo mvcuda-ubuntu1804.pin/etc/apt/preferences.d/cuda-repository-pin-600
```

```
$ wget https://developer.download.nvidia.com/compute/cuda/10.2/Prod/local_installers/cuda-repo-ubuntu1804-10-2-local-10.2.89-440.33.01_1.0-1_amd64.deb
```

```
$ sudo dpkg-icuda-repo-ubuntu1804-10-2-local-10.2.89-440.33.01_1.0-1_amd64.deb
```

```
$ sudo apt-keyadd/var/cuda-repo-10-2-local-10.2.89-440.33.01/7fa2af80.pub
```

```
$ sudo apt-get update
```

```
$ sudo apt-get -y install cuda
```

KÜTÜPHANELER

Kütüphaneler

Cv2

```
import cv2
```

```
$ sudo apt update
$ sudo apt install python3-opencv
```

Pytorch

```
import pytorch
```

```
from torchvision.transforms import Compose
```

- **İnstall for Ubuntu=10.2**

```
$ conda install pytorch torchvision cudatoolkit=10.2 -c pytorch
```

```
$ version=1.10.1+cu102
```

Timm

- **timm documents**

🔗 https://timm.fast.ai/

- **Install**

```
$ pip install timm
```

```
$ pip install git+https://github.com/rwightman/pytorch-image-models.git
```

Midas

- **Install**

```
https://github.com/snayfach/MIDAS/blob/master/docs/install.md
```

```
from midas.dpt_depth import DPTDepthModel
```

```
from midas.transforms import Resize, NormalizeImage, PrepareForNet
```

Pyautogui

```
$ sudo apt-get install scrot

$ sudo apt-get install python3-tk

$ sudo apt-get install python3-dev
```

```
import pyautogui
```

ColorMap

```
from colormap import rgb2hex
```

- **Install**

```
$ sudo apt-get update -y

$ sudo apt-get install -y python-colormap
```

KOD

```
#------------------------------ Libraries
-----------------------------------------------------
import cv2
import torch
import time
import numpy as np
import os
from midas.dpt_depth import DPTDepthModel
from midas.transforms import Resize, NormalizeImage, PrepareForNet
from torchvision.transforms import Compose
import pyautogui
from colormap import rgb2hex
#------------------------ Load model type Move model to GPU if available
---------------------------------

model_type = "DPT_Large"
model_path =
os.path.join(os.getcwd(),"weights/dpt_large-midas-2f21e586.pt")
midas = DPTDepthModel(
            path=model_path,
            backbone="vitl16_384",
            non_negative=True,
        )
device = torch.device("cuda") if torch.cuda.is_available() else
torch.device("cpu")
midas.to(device)
transform = Compose(
        [
            Resize(
                384,
                384,
                resize_target=None,
                keep_aspect_ratio=True,
                ensure_multiple_of=32,
                resize_method="minimal",
```

```
                image_interpolation_method=cv2.INTER_CUBIC,
            ),
            NormalizeImage(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]),
            PrepareForNet(),
        ]
    )

midas.eval()

# ---------------------------------- Optimize
------------------------------------------------

optimize  = True
if optimize==True:

        if device == torch.device("cuda"):
            midas = midas.to(memory_format=torch.channels_last)
            midas = midas.half()
cap = cv2.VideoCapture(0)

#----------------------------  Main Loop
—-------------------------------------------------
while cap.isOpened():

    success, img = cap.read()
    success, img1 = cap.read()
    start = time.time()
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)/255.0

#-------------------------------- Apply input
transforms---------------------------------------
    input_batch = transform({"image": img})["image"]

#----------------- Prediction and Resize to original
resolution----------------------------------
    with torch.no_grad():
            sample =
torch.from_numpy(input_batch).to(device).unsqueeze(0)
            if optimize==True and device == torch.device("cuda"):
                sample = sample.to(memory_format=torch.channels_last)
                sample = sample.half()
            prediction = midas.forward(sample)
            prediction = (
                torch.nn.functional.interpolate(
                    prediction.unsqueeze(1),
                    size=img.shape[:2],
```

```
                    mode="bicubic",
                    align_corners=False,
                )
                .squeeze()
                .cpu()
                .numpy()
            )
    depth_min = prediction.min()
    depth_max = prediction.max()
    max_val = (2**(8*1))-1

    if depth_max - depth_min > np.finfo("float").eps:
        out = max_val * (prediction - depth_min) / (depth_max -
depth_min)
    else:
        out = np.zeros(prediction.shape, dtype=prediction.type)

    end = time.time()
    totalTime = end - start
    fps = 1 / totalTime

#------------------------------Calculate distance from rgb values
----------------------------
    x, y = pyautogui.position()
    r,g,b = pyautogui.pixel(x, y)
    hexa=rgb2hex(r,g,b) # r,g,b to hex
    decimal = int(hexa[1:], 16)
    #Lineer Equation for calculating distance
    distance= (-1.870*10**-5)*decimal+362.3
#------------------------------ Showing Images and Colormap
-----------------------------------
    cv2.putText(img, f'FPS: {int(fps)}', (20,70),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0,255,0), 2)
    out1 = out.astype(np.uint8)
    out = out.astype(np.uint8)
    out = cv2.applyColorMap(out , cv2.COLORMAP_MAGMA)
    cv2.rectangle(out,(x-150,y-150),(x-100,y-100),(0,0,255),2)
    cv2.putText(out, f'distance: {int(distance)}', (20,70),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0,255,0), 2)
    cv2.imshow('Orijinal-Image', img1)
    cv2.imshow('Depth Map-Magma', out)
    cv2.imshow('Distance-Grey', out1)


#------------------------ Exit
-------------------------------------------------------------
```

```
    if cv2.waitKey(5) & 0xFF == 2:
        break

cap.release()
```

REFERANSLAR

🔗https://github.com/niconielsen32/ComputerVision/blob/master/MonocularDepth/midasDepthMap.py

🔗https://pytorch.org/hub/intelisl_midas_v2/

🔗https://drive.google.com/file/d/1kyBpYKYPmzx8pkFfQEq9mMgdd8qOb5le/view?usp=sharing

🔗https://www.youtube.com/watch?v=MNzdybzH0kM