과제 #4: Capture motion and facial expressions from a single image in a video

전자공학과 120220117 박준호

1. 목적

   ✓ 자신만의 비디오 데이터를 이용하여 단일 이미지로부터 사람의 모션과 얼굴 표정을 SMPL로 렌더링하는 과제

2. 코드 분석

   A. SMPLpix_training.ipynb

   i. 위 파일 내에는 smplpix/train.py 코드를 실행시키는 셀만 존재하므로, 이 py 파일 위주로 분석

```
print("*************************************************************************************\n"+
      "***************************** SMPLpix Training Loop  ********************************\n"+
      "*************************************************************************************\n"+
      "******** Copyright (c) 2021 - now, Sergey Prokudin (sergey.prokudin@gmail.com) **********\n"+
      "*************************************************************************************+*\n\n")

args = get_smplpix_arguments()
print("ARGUMENTS:")
pprint.pprint(args)

if not os.path.exists(args.workdir):
    os.makedirs(args.workdir)

log_dir = os.path.join(args.workdir, 'logs')
os.makedirs(log_dir, exist_ok=True)

# if args.data_url is not None:
#     download_and_unzip(args.data_url, args.workdir)
#     args.data_dir = os.path.join(args.workdir, 'smplpix_data')
args.data_dir = os.path.join(args.workdir, 'smplpix_data')

train_dir = os.path.join(args.data_dir, 'train')
val_dir = os.path.join(args.data_dir, 'validation')
test_dir = os.path.join(args.data_dir, 'test')
```

   1. 학습 시작 전 log 및 data 경로 먼저 설정

```
train_dataset = SMPLPixDataset(data_dir=train_dir,
                               perform_augmentation=True,
                               augmentation_probability=args.aug_prob,
                               downsample_factor=args.downsample_factor,
                               n_input_channels=args.n_input_channels,
                               n_output_channels=args.n_output_channels)
train_dataloader = DataLoader(train_dataset, batch_size=args.batch_size)

if os.path.exists(val_dir):
    val_dataset = SMPLPixDataset(data_dir=val_dir,
                                 perform_augmentation=False,
                                 augmentation_probability=args.aug_prob,
                                 downsample_factor=args.downsample_factor,
                                 n_input_channels=args.n_input_channels,
                                 n_output_channels=args.n_output_channels)
    val_dataloader = DataLoader(val_dataset, batch_size=args.batch_size)
else:
    print("no validation data was provided, will use train data for validation...")
    val_dataloader = train_dataloader
```

   2. SMPLPixDataset 클래스를 통해 train_dataset, train_dataloader 및

val_dataset, val_dataloader를 정의

A. 정의 과정에서 데이터셋 디렉토리(train_dir, val_dir), augmentation 수행 여부(perform_augmentation), input 채널 수(args.n_input_channels), output 채널 수(args.n_output_channels), 배치 사이즈(args.batch_size) 등과 같은 hyper-parameter 설정

```
print("defining the neural renderer model (U-Net)...")
unet = UNet(in_channels=args.n_input_channels,
            out_channels=args.n_output_channels,
            sigmoid_output=args.sigmoid_output,
            n_blocks=args.n_unet_blocks, dim=2, up_mode='resizeconv_linear').to(args.device)

if args.checkpoint_path is None:
    ckpt_path = os.path.join(args.workdir, 'network.h5')
else:
    ckpt_path = args.checkpoint_path

if args.resume_training and os.path.exists(ckpt_path):
    print("found checkpoint, resuming from: %s" % ckpt_path)
    unet.load_state_dict(torch.load(ckpt_path))
if not args.resume_training:
    print("starting training from scratch, cleaning the log dirs...")
    shutil.rmtree(log_dir)
```

3. UNet 네트워크 정의 및 weight 파일 load

A. 학습에 쓰이는 네트워크인 UNet에 대한 hyper-parameter(e.g. args.n_input_channels, args.n_output_channels, args.n_unet_blocks 등)를 설정하여 모델을 정의

B. 그 후, 저장되어 있는 weight(h5 파일)가 있으면 그 파일을 불러오고, 없으면 따로 불러오지 않고 weight를 저장할 경로만 설정

```
print("starting training...")
finished = False
try:
    train(model=unet, train_dataloader=train_dataloader, val_dataloader=val_dataloader,
          log_dir=log_dir, ckpt_path=ckpt_path, device=args.device, n_epochs=args.n_epochs,
          eval_every_nth_epoch=args.eval_every_nth_epoch, sched_patience=args.sched_patience,
          init_lr=args.learning_rate)
    finished = True

except KeyboardInterrupt:
    print("training interrupted, generating final animations...")
    generate_eval_video(args, train_dir, unet, save_target=True)
    generate_eval_video(args, val_dir, unet, save_target=True)
    generate_eval_video(args, test_dir, unet, save_target=True)

if finished:
    generate_eval_video(args, train_dir, unet, save_target=True)
    generate_eval_video(args, val_dir, unet, save_target=True)
    generate_eval_video(args, test_dir, unet, save_target=True)

return
```

4. 미리 정의한 여러 variable을 train 함수에 넣고 학습 시작

A. train 함수는 아래와 같이 정의되어 있음

```
def train(model, train_dataloader, val_dataloader, log_dir, ckpt_path, device,
          n_epochs=1000, eval_every_nth_epoch=50, sched_patience=5, init_lr=1.0e-4):

    vgg = Vgg16Features(layers_weights = [1, 1/16, 1/8, 1/4, 1]).to(device)
    criterion_l1 = nn.L1Loss().to(device)
    optimizer = torch.optim.Adam(model.parameters(), lr=init_lr)
    sched = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                                       patience=sched_patience,
                                                       verbose=True)

    for epoch_id in tqdm(range(0, n_epochs)):

        model.train()
        torch.save(model.state_dict(), ckpt_path)

        for batch_idx, (x, ytrue, img_names) in enumerate(train_dataloader):
            x, ytrue = x.to(device), ytrue.to(device)
            ypred = model(x)
            vgg_loss = criterion_l1(vgg(ypred), vgg(ytrue))
            optimizer.zero_grad()
            vgg_loss.backward()
            optimizer.step()

        if epoch_id % eval_every_nth_epoch == 0:
            print("\ncurrent epoch: %d" % epoch_id)
            eval_dir = os.path.join(log_dir, 'val_preds_%04d' % epoch_id)
            val_loss = evaluate(model, val_dataloader, eval_dir, device, vgg, show_progress=False)
            sched.step(val_loss)

    return
```

i. Feature extractor인 Vgg16, L1 loss인 criterion, Adam optimizer, scheduler 등을 정의하고, 지정한 epoch 수만큼 반복적으로 학습을 진행

ii. 하나의 epoch에서 지정한 batch size만큼 train_loader가 input인 x을 꺼내고 이를 앞서 정의한 UNet model에 target인 ytrue와 비교하여 loss를 계산

1. Back-propagation 연산이 진행되는 지점으로써 실질적인 학습이 이루어지는 곳

iii. 지정한 epoch 마다 evaluation을 함으로써 validation loss를 계산

1. 이는 학습 진행이 잘 되고 있는지 주기적으로 확인하는 코드

5. 학습이 종료되면, evaluation 결과에 대한 video 생성

B. Convert_Video_to_SMPLpix_Dataset.ipynb

i. SMPLpix 데이터셋을 만들기 위해 필요한 요소들을 준비하는 코드

ii. 크게 다음과 같은 과정으로 진행

1. 사용하려는 비디오 다운로드
2. 비디오로부터 2D keypoints 추출
3. 비디오로부터 3D human mesh 추출
4. Input이 SMPL-X mesh render 이미지이고, output이 해당하는 ground-truth

video frame인 pair로 짝지어서 데이터셋을 생성

5. 데이터셋을 train/validation/test로 나누고 구글 드라이브에 복사

iii. 세부 진행 코드

1. 사용하려는 비디오 다운로드

```
import os
from google.colab import files

def download_youtube_video(img_url, save_path, resolution_id=-3):
  print("downloading the video: %s" % img_url)
  res_path = YouTube(img_url).streams.order_by('resolution')[resolution_id].download(save_path)
  return res_path

def download_dropbox_url(url, filepath, chunk_size=1024):
    import requests
    headers = {'user-agent': 'Wget/1.16 (linux-gnu)'}
    r = requests.get(url, stream=True, headers=headers)
    with open(filepath, 'wb') as f:
        for chunk in r.iter_content(chunk_size=chunk_size):
            if chunk:
                f.write(chunk)
    return filepath

!rm -rf /content/data

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

VIDEO_SOURCE = 'dropbox' #@param ["youtube", "upload", "google drive", "dropbox"]
if VIDEO_SOURCE == 'dropbox':
  DROPBOX_URL  = 'https://www.dropbox.com/s/rjqwf894ovso218/smplpix_test_video_na.mp4?dl=0' #@param
  VIDEO_PATH  = '/content/video.mp4'
  download_dropbox_url(DROPBOX_URL, VIDEO_PATH)
elif VIDEO_SOURCE == 'upload':
  print("Please upload the video: ")
  uploaded = files.upload()
  VIDEO_PATH = os.path.join('/content', list(uploaded.keys())[0])
elif VIDEO_SOURCE == 'youtube':
  !pip install pytube
  from pytube import YouTube
  YOTUBE_VIDEO_URL  = ''  #@param
  VIDEO_PATH = download_youtube_video(YOTUBE_VIDEO_URL, '/content/')
elif VIDEO_SOURCE == 'google drive':
  from google.colab import drive
  drive.mount('/content/gdrive')
  GOOGLE_DRIVE_PATH = '' #@param
  VIDEO_PATH  = GOOGLE_DRIVE_PATH
print("video is uploaded to %s" % VIDEO_PATH)
```

```
RES_DIR = '/content/data'
FRAMES_DIR = os.path.join(RES_DIR, 'images')
!rm -rf $RES_DIR
!mkdir $RES_DIR
!mkdir $FRAMES_DIR
!ffmpeg -i "$VIDEO_PATH" -vf fps=$FPS  -qscale:v 2 '$FRAMES_DIR/%05d.png'

from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

def load_img(img_path):

  return np.asarray(Image.open(img_path))/255

test_img_path = os.path.join(FRAMES_DIR, os.listdir(FRAMES_DIR)[0])

test_img = load_img(test_img_path)

plt.figure(figsize=(5, 10))
plt.title("extracted image example")
plt.imshow(test_img)
```

2. 비디오로부터 2D keypoints 추출

```
%cd /content
import os
from os.path import exists, join, basename, splitext

git_repo_url = 'https://github.com/CMU-Perceptual-Computing-Lab/openpose.git'
project_name = splitext(basename(git_repo_url))[0]
if not exists(project_name):
  # see: https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/949
  # install new CMake becaue of CUDA10
  !wget -q https://cmake.org/files/v3.13/cmake-3.13.0-Linux-x86_64.tar.gz
  !tar xfz cmake-3.13.0-Linux-x86_64.tar.gz --strip-components=1 -C /usr/local

  # clone openpose
  !git clone -q --depth 1 $git_repo_url
  # download models
  !wget -O /content/openpose/models/hand/pose_iter_102000.caffemodel https://polybox.ethz.ch/index.php/s/Oim76cuqrDVbdxm/download
  !wget -O /content/openpose/models/pose/body_25/pose_iter_584000.caffemodel https://polybox.ethz.ch/index.php/s/m5NQAhd7ukVPRoL/download
  !wget -O /content/openpose/models/face/pose_iter_116000.caffemodel https://polybox.ethz.ch/index.php/s/cEaF1FTpKjj3ZbH/download
  !sed -i 's/execute_process(COMMAND git checkout master WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}\/3rdparty\/caffe)/execute_process(COMMAND git checkout f019d0dfe86f49d1140961f8c7de
  # install system dependencies
  !apt-get -qq install -y libatlas-base-dev libprotobuf-dev libleveldb-dev libsnappy-dev libhdf5-serial-dev protobuf-compiler libgflags-dev libgoogle-glog-dev liblmdb-dev opencl-
  # install python dependencies
  !pip install -q youtube-dl
  # build openpose
  !cd openpose && rm -rf build || true && mkdir build && cd build && cmake .. && make -j`nproc`
```

```
# @title Run OpenPose on the extracted frames
%cd /content
KEYPOINTS_DIR = os.path.join(RES_DIR, 'keypoints')
OPENPOSE_IMAGES_DIR = os.path.join(RES_DIR, 'openpose_images')
!mkdir $KEYPOINTS_DIR
!mkdir $OPENPOSE_IMAGES_DIR

!cd openpose && ./build/examples/openpose/openpose.bin --image_dir $FRAMES_DIR --write_json $KEYPOINTS_DIR --face --hand --display 0   --write_images $OPENPOSE_IMAGES_DIR

input_img_path = os.path.join(FRAMES_DIR, sorted(os.listdir(FRAMES_DIR))[0])
openpose_img_path = os.path.join(OPENPOSE_IMAGES_DIR, sorted(os.listdir(OPENPOSE_IMAGES_DIR))[0])

test_img = load_img(input_img_path)
open_pose_img = load_img(openpose_img_path)

plt.figure(figsize=(10, 10))
plt.title("Input Frame + Openpose Prediction")
plt.imshow(np.concatenate([test_img, open_pose_img], 1))
```

3. 비디오로부터 3D human mesh 추출

```
%cd /content
!pip install chumpy
!pip install smplx
!git clone https://github.com/vchoutas/smplx
%cd smplx
!python setup.py install

#vposer
!pip install git+https://github.com/nghorbani/configer
!pip install git+https://github.com/sergeyprokudin/human_body_prior

!pip install torch==1.1.0
%cd /content
!git clone https://github.com/sergeyprokudin/smplify-x
%cd /content/smplify-x
!pip install -r requirements.txt
```

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

SMPLX_ZIP_PATH = '/content/gdrive/MyDrive/datasets/models_smplx_v1_1.zip' # @param {type:"string"}
VPOSER_ZIP_PATH = '/content/gdrive/MyDrive/datasets/vposer_v1_0.zip' # @param {type:"string"}

SMPLX_MODEL_PATH = '/content/smplx'
!mkdir $SMPLX_MODEL_PATH
!unzip -n '$SMPLX_ZIP_PATH' -d  $SMPLX_MODEL_PATH
VPOSER_MODEL_PATH = '/content/vposer'
!mkdir $VPOSER_MODEL_PATH
!unzip -n '$VPOSER_ZIP_PATH' -d  $VPOSER_MODEL_PATH
```

```
gender = 'male' #@param ["neutral", "female", "male"]

# @markdown Please keep in mind that estimating 3D body with SMPLify-X framewo

!rm -rf /content/data/smplifyx_results
%cd /content/smplify-x
!git pull origin
!python smplifyx/main.py --config cfg_files/fit_smplx.yaml \
    --data_folder  /content/data \
    --output_folder /content/data/smplifyx_results \
    --visualize=True \
    --gender=$gender \
    --model_folder /content/smplx/models \
    --vposer_ckpt /content/vposer/vposer_v1_0 \
    --part_segm_fn smplx_parts_segm.pkl
```

4. Input이 SMPL-X mesh render 이미지이고, output이 해당하는 ground-truth video frame인 pair로 짝지어서 데이터셋을 생성 및 데이터셋을 train/validation/test로 나누고 구글 드라이브에 복사

```
# @title Make train-test-validation splits, copy data to final folders

import shutil
train_ratio = 0.9 #@param
final_zip_path = '/content/gdrive/MyDrive/datasets/smplpix_data_test.zip' # @param {type:"string"}
target_images_path = '/content/data/smplifyx_results/input_images'
smplifyx_renders = '/content/data/smplifyx_results/rendered_smplifyx_meshes'
smplpix_data_path = '/content/smplpix_data'

train_input_dir = os.path.join(smplpix_data_path, 'train', 'input')
train_output_dir = os.path.join(smplpix_data_path, 'train', 'output')
val_input_dir = os.path.join(smplpix_data_path, 'validation', 'input')
val_output_dir = os.path.join(smplpix_data_path, 'validation', 'output')
test_input_dir = os.path.join(smplpix_data_path, 'test', 'input')
test_output_dir = os.path.join(smplpix_data_path, 'test', 'output')

!mkdir -p $train_input_dir
!mkdir -p $train_output_dir
!mkdir -p $val_input_dir
!mkdir -p $val_output_dir
!mkdir -p $test_input_dir
!mkdir -p $test_output_dir

img_names = sorted(os.listdir(target_images_path))
n_images = len(img_names)
n_train_images = int(n_images * train_ratio)
n_val_images = int(n_images * (1-train_ratio) / 2)
train_images = img_names[0:n_train_images]
val_images = img_names[n_train_images:n_train_images+n_val_images]
test_images = img_names[n_train_images:]

for img in train_images:
  shutil.copy(os.path.join(smplifyx_renders, img), train_input_dir)
  shutil.copy(os.path.join(target_images_path, img), train_output_dir)

for img in val_images:
  shutil.copy(os.path.join(smplifyx_renders, img), val_input_dir)
  shutil.copy(os.path.join(target_images_path, img), val_output_dir)

for img in test_images:
  shutil.copy(os.path.join(smplifyx_renders, img), test_input_dir)
  shutil.copy(os.path.join(target_images_path, img), test_output_dir)

%cd /content
!zip -r $final_zip_path smplpix_data/
```