

Họ và tên: Trần Nguyễn Long Hưng
Mssv: 20127180
Lớp: 20TGMT

Practice #2 - Final

(Ứng dụng xử lý ảnh số và video số)

Chủ đề được sử dụng trong bài báo cáo này là [“Motion Representations for Articulated Animation - CVPR 2021”](#):

- Paper: <https://arxiv.org/abs/2104.11280>
- Video: https://youtu.be/gpBYN8t8_yY
- Code: <https://github.com/snap-research/articulated-animation>

Bảng đánh giá

Stt	Phân đoạn	Kết quả
1	Giới thiệu	100%
2	Cài đặt thử nghiệm	100%
3	Kết luận	100%

I. Giới thiệu

Bài báo đề xuất một phương pháp mới để tạo chuyển động các đối tượng có “khớp nối”, như con người và động vật. Công trình sử dụng mô hình FOMM (First Object Motion Model) để mô hình hóa chuyển động của các thành phần của đối tượng. FOMM hoạt động bằng cách dự đoán chuyển động của các điểm đặc trưng từ một chuỗi hình ảnh (video), từ đó tạo ra các chuyển động cho nhiều đối tượng khác nhau. Việc sử dụng FOMM đem lại một cải thiện đáng kể về chất lượng các đối tượng được tạo hình chuyển động. Các đối tượng trở nên sống động và tự nhiên hơn, và chúng có khả năng di chuyển một cách mượt mà và chính xác hơn.

II. Cài đặt và thử nghiệm

Báo cáo này nay sẽ cài đặt và thử nghiệm trên colab.

1. Chuẩn bị và cài đặt

1.1 Kiểm tra môi trường colab

Bước này để kiểm tra và hiển thị thông tin GPU được sử dụng trong TensorFlow và PyTorch

▼ Step 01. Check the colab enviroment

This is formatted as code

The GPUs available in Colab often include Nvidia A100, T4, V100.

```
import tensorflow as tf
print(tf.test.gpu_device_name())

from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())

!cat /proc/meminfo
```

```
[2] import torch

print(torch.cuda.current_device())
print(torch.cuda.device(0))
print(torch.cuda.device_count())
print(torch.cuda.get_device_name(0))
print(torch.cuda.is_available())

# setting device on GPU if available, else CPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Using device:', device)
print()

# additional info when using cuda
if device.type == 'cuda':
    print(torch.cuda.get_device_name(0))
    print('Memory Usage:')
    print('Allocated:', round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
    print('Cached:   ', round(torch.cuda.memory_cached(0)/1024**3,1), 'GB')
```

1.2 Kết nối GoogleDrive và clone source git

▼ Step 02. Mount drive, Clone repository, Install git lfs, Pulling all checkpoints

Mount drive

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

Mounted at /content/gdrive

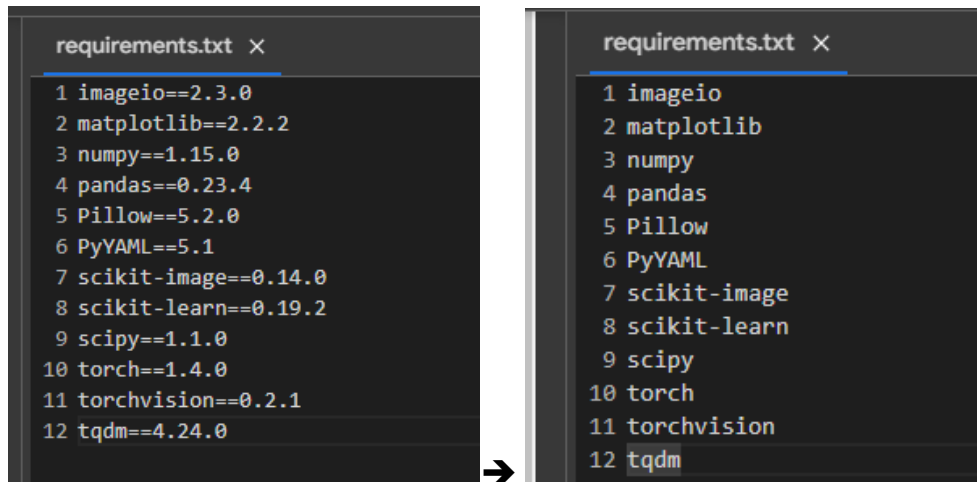
```
%cd /content/gdrive/My\ Drive
%mkdir UDXLAV_Practice
%cd /content/gdrive/My\ Drive/UDXLAV_Practice
!rm -rf articulated-animation

!curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
!sudo apt-get install git-lfs
!git lfs install
!git clone https://github.com/snap-research/articulated-animation.git
%cd /content/gdrive/My\ Drive/UDXLAV_Practice/articulated-animation
```

Checking for curl...

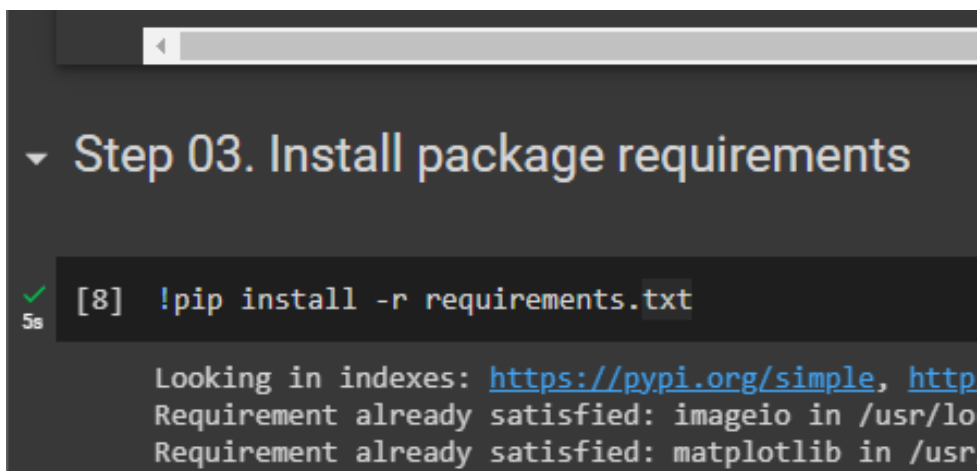
1.3 Cài đặt các package cần thiết

Lưu ý: Trước khi chạy cài đặt, ta cần chỉnh sửa lại file *requirement.txt*. Vì lý do một số package cũ công trình sử dụng không còn được hỗ trợ trên phiên bản colab mới nhất và để tránh gặp lỗi.



```
requirements.txt ×
1 imageio==2.3.0
2 matplotlib==2.2.2
3 numpy==1.15.0
4 pandas==0.23.4
5 Pillow==5.2.0
6 PyYAML==5.1
7 scikit-image==0.14.0
8 scikit-learn==0.19.2
9 scipy==1.1.0
10 torch==1.4.0
11 torchvision==0.2.1
12 tqdm==4.24.0
```

```
requirements.txt ×
1 imageio
2 matplotlib
3 numpy
4 pandas
5 Pillow
6 PyYAML
7 scikit-image
8 scikit-learn
9 scipy
10 torch
11 torchvision
12 tqdm
```



```
Step 03. Install package requirements

[8] !pip install -r requirements.txt

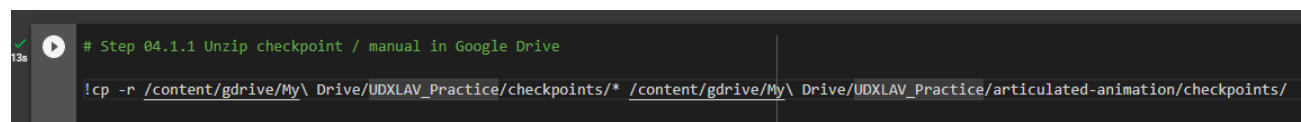
Looking in indexes: https://pypi.org/simple, https://
Requirement already satisfied: imageio in /usr/loc
Requirement already satisfied: matplotlib in /usr/
```

1.4 Load các pre-trained checkpoint

V các file .pth (pre-trained checkpoint) của folder “articulated-animation/checkpoint” khá nặng không thể git trực tiếp về source code. Ta cài đặt thủ công bằng cách tải về và upload lên folder “articulated-animation/checkpoint”

Link tác giả: https://drive.google.com/drive/folders/1jCeFPqFU_wKNYwof0ONICwsj3xHlr_tb?usp=share_link

Ở đoạn code sau đã download và upload ở thư mục drive riêng nên chỉ cần dùng lệnh để copy các file sang folder “articulated-animation/checkpoint”



```
# Step 04.1.1 Unzip checkpoint / manual in Google Drive

!cp -r /content/gdrive/My\ Drive/UDXLAV_Practice/checkpoints/* /content/gdrive/My\ Drive/UDXLAV_Practice/articulated-animation/checkpoints/
```

2. Thực nghiệm

2.1 Những lưu ý trước khi chạy demo

Trước khi chạy demo thử ta cũng cần chỉnh lại một số đoạn code ở các file sau:

- *logger.py*: ở các phiên bản mới của *skimage*, hàm *circle* được đổi thành *circle_perimeter*

```

logger.py
@@ -13,7 +13,7 @@
13 13 import imageio
14 14
15 15 import os
16 - from skimage.draw import circle
16 + from skimage.draw import circle_perimeter
17 17
18 18 import matplotlib.pyplot as plt
19 19 import collections
@@ -146,7 +146,7 @@ def draw_image_with_kp(self, image, kp_array):
146 146     kp_array = spatial_size * (kp_array + 1) / 2
147 147     num_regions = kp_array.shape[0]
148 148     for kp_ind, kp in enumerate(kp_array):
149 -         rr, cc = circle(kp[1], kp[0], self.kp_size, shape=image.shape[:2])
149 +         rr, cc = circle_perimeter(kp[1], kp[0], self.kp_size, shape=image.shape[:2])
150 150     image[rr, cc] = np.array(self.colormap(kp_ind / num_regions))[3]
151 151     return image
152 152

```

- *demo.py*: từ phiên bản PyYAML 5.1 trở lên. Câu lệnh được chỉ định đối tượng “Loader” nhằm bảo vệ, ngăn chặn các mã độc hại thông qua YAML

```

demo.py
@@ -33,7 +33,7 @@
33 33
34 34 def load_checkpoints(config_path, checkpoint_path, cpu=False):
35 35     with open(config_path) as f:
36 -         config = yaml.load(f)
36 +         config = yaml.load(f, Loader=yaml.FullLoader)
37 37
38 38     generator = Generator(num_regions=config['model_params']['num_regions'],
39 39                          num_channels=config['model_params']['num_channels'],

```

2.2 Chạy demo

Ta chạy dòng lệnh sau:

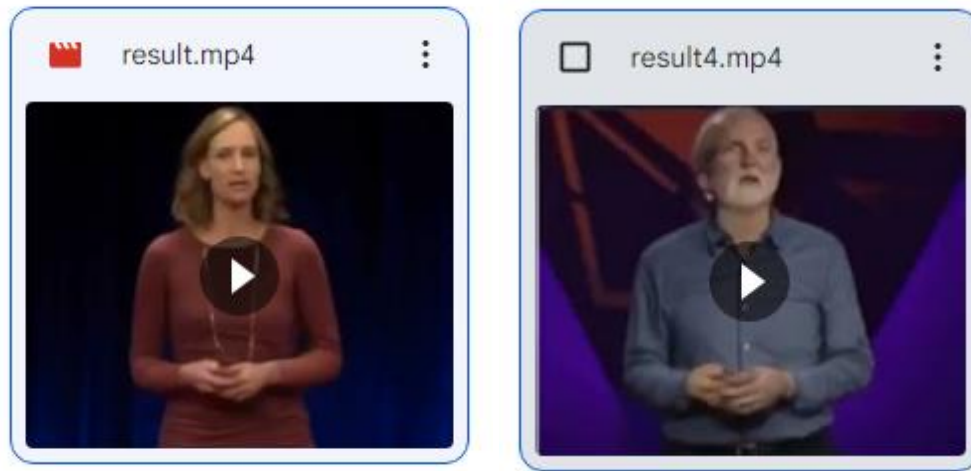
1. `python demo.py --config config/dataset_name.yaml --driving_video path/to/driving --source_image path/to/source --checkpoint path/to/checkpoint`

```
▼ Step 05. Run demo

!python demo.py --config config/ted384.yaml --driving_video sup-mat/driving.mp4 --source_image sup-mat/source.png --checkpoint checkpoints/ted384.pth

/content/gdrive/MyDrive/UDXLAV_Practice/articulated-animation/demo.py:104: DeprecationWarning: Starting with ImageIO v3 the behavior of this function w
source_image = imageio.imread(opt.source_image)
/usr/local/lib/python3.10/dist-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming release, it will be required to pass the i
return_VF.meshgrid(tensors, **kwargs) # type: ignore[attr-defined]
0% 0/133 [00:00<?, ?it/s] /usr/local/lib/python3.10/dist-packages/torch/nn/functional.py:4236: UserWarning: Default grid_sample and affine_grid behavi
warnings.warn(
100% 133/133 [00:09<00:00, 13.48it/s]
```

Kết quả là file “result.mp4”



Nhận xét: Các kết quả output cho ra chuyển động gần như tương đồng với chuỗi ảnh (video) “driving”. Có thể áp dụng cho nhiều trường hợp ảnh đầu người đầu vào như như già trẻ, nam, nữ,... đều cho ra kết quả tốt.

III. Kết luận

Phương pháp được đề xuất là một bước tiến lớn so với các phương pháp trước đây trong việc tạo hình chuyển động các đối tượng. Triển vọng cho việc tạo hoạt hình chuyển động cho các đối tượng trong nhiều ứng dụng khác nhau, như video game, phim ảnh, thực tế ảo (VR).