first_order_model_demo

April 27, 2020

1 Demo for paper "First Order Motion Model for Image Animation"

Clone repository

[4]:
```
!git clone https://github.com/AliaksandrSiarohin/first-order-model
```

```
Cloning into 'first-order-model'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 216 (delta 0), reused 0 (delta 0), pack-reused 215
Receiving objects: 100% (216/216), 71.45 MiB | 35.99 MiB/s, done.
Resolving deltas: 100% (106/106), done.
```

[5]:
```
cd first-order-model
```

```
/content/first-order-model/first-order-model
```

Mount your Google drive folder on Colab

[6]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
```

Add folder https://drive.google.com/drive/folders/1kZ1gCnpfU0BnpdU47pLM_TQ6RypDDqgw?usp=sha to your google drive.

Load driving video and source image

[7]:
```
import imageio
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from skimage.transform import resize
from IPython.display import HTML
import warnings
warnings.filterwarnings("ignore")
```

```
source_image = imageio.imread('/content/gdrive/My Drive/
 ↪first-order-motion-model/02.png')
driving_video = imageio.mimread('/content/gdrive/My Drive/
 ↪first-order-motion-model/04.mp4')


#Resize image and video to 256x256

source_image = resize(source_image, (256, 256))[..., :3]
driving_video = [resize(frame, (256, 256))[..., :3] for frame in driving_video]

def display(source, driving, generated=None):
    fig = plt.figure(figsize=(8 + 4 * (generated is not None), 6))

    ims = []
    for i in range(len(driving)):
        cols = [source]
        cols.append(driving[i])
        if generated is not None:
            cols.append(generated[i])
        im = plt.imshow(np.concatenate(cols, axis=1), animated=True)
        plt.axis('off')
        ims.append([im])

    ani = animation.ArtistAnimation(fig, ims, interval=50, repeat_delay=1000)
    plt.close()
    return ani


HTML(display(source_image, driving_video).to_html5_video())
```

[7]: <IPython.core.display.HTML object>

Create a model and load checkpoints

[0]:
```
from demo import load_checkpoints
generator, kp_detector = load_checkpoints(config_path='config/vox-256.yaml',
                            checkpoint_path='/content/gdrive/My Drive/
 ↪first-order-motion-model/vox-cpk.pth.tar')
```

Perfrom image animation

[9]:
```
from demo import make_animation
from skimage import img_as_ubyte

predictions = make_animation(source_image, driving_video, generator,␣
 ↪kp_detector, relative=True)

#save resulting video
```

```
imageio.mimsave('../generated.mp4', [img_as_ubyte(frame) for frame in␣
 ↪predictions])
#video can be downloaded from /content folder

HTML(display(source_image, driving_video, predictions).to_html5_video())
```

```
100%|| 211/211 [00:09<00:00, 21.89it/s]
```

[9]:
```
<IPython.core.display.HTML object>
```

In the cell above we use relative keypoint displacement to animate the objects. We can use absolute coordinates instead, but in this way all the object proporions will be inherited from the driving video. For example Putin haircut will be extended to match Trump haircut.

[10]:
```
predictions = make_animation(source_image, driving_video, generator,␣
 ↪kp_detector, relative=False, adapt_movement_scale=True)
HTML(display(source_image, driving_video, predictions).to_html5_video())
```

```
100%|| 211/211 [00:09<00:00, 21.96it/s]
```

[10]:
```
<IPython.core.display.HTML object>
```

1.1 Running on your data

First we need to crop a face from both source image and video, while simple graphic editor like paint can be used for cropping from image. Cropping from video is more complicated. You can use ffpmeg for this.

[11]:
```
!ffmpeg -i /content/gdrive/My\ Drive/first-order-motion-model/07.mkv -ss 00:00:
 ↪10.50 -t 00:00:08 -filter:v "crop=600:600:760:50" -async 1 hinton.mp4
```

```
ffmpeg version 3.4.6-0ubuntu0.18.04.1 Copyright (c) 2000-2019 the FFmpeg
developers
  built with gcc 7 (Ubuntu 7.3.0-16ubuntu3)
  configuration: --prefix=/usr --extra-version=0ubuntu0.18.04.1
--toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu
--incdir=/usr/include/x86_64-linux-gnu --enable-gpl --disable-stripping
--enable-avresample --enable-avisynth --enable-gnutls --enable-ladspa --enable-
libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio
--enable-libflite --enable-libfontconfig --enable-libfreetype --enable-
libfribidi --enable-libgme --enable-libgsm --enable-libmp3lame --enable-
libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-
libpulse --enable-librubberband --enable-librsvg --enable-libshine --enable-
libsnappy --enable-libsoxr --enable-libspeex --enable-libssh --enable-libtheora
--enable-libtwolame --enable-libvorbis --enable-libvpx --enable-libwavpack
--enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-
libzmq --enable-libzvbi --enable-omx --enable-openal --enable-opengl --enable-
sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-
```

```
chromaprint --enable-frei0r --enable-libopencv --enable-libx264 --enable-shared
  libavutil      55. 78.100 / 55. 78.100
  libavcodec     57.107.100 / 57.107.100
  libavformat    57. 83.100 / 57. 83.100
  libavdevice    57. 10.100 / 57. 10.100
  libavfilter     6.107.100 /  6.107.100
  libavresample   3.  7.  0 /  3.  7.  0
  libswscale      4.  8.100 /  4.  8.100
  libswresample   2.  9.100 /  2.  9.100
  libpostproc    54.  7.100 / 54.  7.100
Input #0, matroska,webm, from '/content/gdrive/My Drive/first-order-motion-
model/07.mkv':
  Metadata:
    ENCODER         : Lavf57.83.100
  Duration: 00:14:59.73, start: 0.000000, bitrate: 2343 kb/s
    Stream #0:0(eng): Video: vp9 (Profile 0), yuv420p(tv, bt709), 1920x1080, SAR
1:1 DAR 16:9, 29.97 fps, 29.97 tbr, 1k tbn, 1k tbc (default)
    Metadata:
      DURATION        : 00:14:59.665000000
    Stream #0:1(eng): Audio: aac (LC), 44100 Hz, stereo, fltp (default)
    Metadata:
      HANDLER_NAME    : SoundHandler
      DURATION        : 00:14:59.727000000
Stream mapping:
  Stream #0:0 -> #0:0 (vp9 (native) -> h264 (libx264))
  Stream #0:1 -> #0:1 (aac (native) -> aac (native))
Press [q] to stop, [?] for help
-async is forwarded to lavfi similarly to -af
aresample=async=1:min_hard_comp=0.100000:first_pts=0.
[libx264 @ 0x55d9d7198800] using SAR=1/1
[libx264 @ 0x55d9d7198800] using cpu capabilities: MMX2 SSE2Fast
SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2 AVX512
[libx264 @ 0x55d9d7198800] profile High, level 3.1
[libx264 @ 0x55d9d7198800] 264 - core 152 r2854 e9a5903 -
H.264/MPEG-4 AVC codec - Copyleft 2003-2017 - http://www.videolan.org/x264.html
- options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x113 me=hex subme=7 psy=1
psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 8x8dct=1 cqm=0
deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=3 lookahead_threads=1
sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0
constrained_intra=0 bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightb=1
open_gop=0 weightp=2 keyint=250 keyint_min=25 scenecut=40 intra_refresh=0
rc_lookahead=40 rc=crf mbtree=1 crf=23.0 qcomp=0.60 qpmin=0 qpmax=69 qpstep=4
ip_ratio=1.40 aq=1:1.00
Output #0, mp4, to 'hinton.mp4':
  Metadata:
    encoder         : Lavf57.83.100
    Stream #0:0(eng): Video: h264 (libx264) (avc1 / 0x31637661), yuv420p,
600x600 [SAR 1:1 DAR 1:1], q=-1--1, 29.97 fps, 30k tbn, 29.97 tbc (default)
```

```
    Metadata:
      DURATION        : 00:14:59.665000000
      encoder         : Lavc57.107.100 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: -1
    Stream #0:1(eng): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo,
fltp, 128 kb/s (default)
    Metadata:
      HANDLER_NAME    : SoundHandler
      DURATION        : 00:14:59.727000000
      encoder         : Lavc57.107.100 aac
frame=  240 fps= 31 q=-1.0 Lsize=     769kB time=00:00:08.01 bitrate=
786.1kbits/s speed=1.05x
video:634kB audio:125kB subtitle:0kB other streams:0kB global headers:0kB muxing
overhead: 1.297344%
[libx264 @ 0x55d9d7198800] frame I:1     Avg QP:20.75  size: 17585
[libx264 @ 0x55d9d7198800] frame P:60    Avg QP:21.68  size:  6731
[libx264 @ 0x55d9d7198800] frame B:179   Avg QP:26.54  size:  1268
[libx264 @ 0x55d9d7198800] consecutive B-frames:  0.4%  0.0%  1.2%
98.3%
[libx264 @ 0x55d9d7198800] mb I  I16..4: 20.5% 70.8%  8.7%
[libx264 @ 0x55d9d7198800] mb P  I16..4:  3.1%  8.0%  0.5%  P16..4:
35.2% 16.0%  6.6%  0.0%  0.0%    skip:30.5%
[libx264 @ 0x55d9d7198800] mb B  I16..4:  0.1%  0.2%  0.0%  B16..8:
34.7%  3.2%  0.3%  direct: 0.8%  skip:60.8%  L0:41.9% L1:52.6% BI: 5.5%
[libx264 @ 0x55d9d7198800] 8x8 transform intra:69.4% inter:80.3%
[libx264 @ 0x55d9d7198800] coded y,uvDC,uvAC intra: 42.5% 49.7% 8.0%
inter: 9.3% 6.8% 0.0%
[libx264 @ 0x55d9d7198800] i16 v,h,dc,p: 73%  7%  7% 13%
[libx264 @ 0x55d9d7198800] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 24% 12% 27%
5%  7%  6%  6%  7%  6%
[libx264 @ 0x55d9d7198800] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 22% 19% 13%
5% 11%  9%  9%  5%  7%
[libx264 @ 0x55d9d7198800] i8c dc,h,v,p: 50% 18% 24%  8%
[libx264 @ 0x55d9d7198800] Weighted P-Frames: Y:1.7% UV:1.7%
[libx264 @ 0x55d9d7198800] ref P L0: 61.3% 14.9% 16.2%  7.4%  0.1%
[libx264 @ 0x55d9d7198800] ref B L0: 92.2%  5.8%  1.9%
[libx264 @ 0x55d9d7198800] ref B L1: 97.2%  2.8%
[libx264 @ 0x55d9d7198800] kb/s:647.83
[aac @ 0x55d9d7199700] Qavg: 374.377
```

Another posibility is to use some screen recording tool, or if you need to crop many images at ones use face detector(https://github.com/1adrianb/face-alignment) , see https://github.com/AliaksandrSiarohin/video-preprocessing for preprcessing of VoxCeleb.

[12]:
```
source_image = imageio.imread('/content/gdrive/My Drive/
 ↪first-order-motion-model/ch.jpg')
driving_video = imageio.mimread('hinton.mp4', memtest=False)
```

```
#Resize image and video to 256x256

source_image = resize(source_image, (256, 256))[..., :3]
driving_video = [resize(frame, (256, 256))[..., :3] for frame in driving_video]

predictions = make_animation(source_image, driving_video, generator,␣
 ↪kp_detector, relative=True,
                              adapt_movement_scale=True)

HTML(display(source_image, driving_video, predictions).to_html5_video())
```

```
100%|| 240/240 [00:11<00:00, 21.77it/s]
```

[12]:
```
<IPython.core.display.HTML object>
```

[14]:
```
HTML(display(source_image, predictions).to_html5_video())
```

[14]:
```
<IPython.core.display.HTML object>
```