Video Generation

1. Environment and Device Setup

Configure CPU and GPU (CUDA) settings, fix PyTorch thread count to 8, and determine the appropriate DEVICE (cuda or cpu).

```
import os
import torch
from diffusers import StableDiffusionPipeline
from TTS.api import TTS
from moviepy.editor import ImageClip, AudioFileClip, concatenate_videoclips

# cpu and device settings
os.environ["OMP_NUM_THREADS"] = "8"   # i did that cuz for accelerate
os.environ["MKL_NUM_THREADS"] = "8"
torch.set_num_threads(8)
torch.set_num_interop_threads(8)

if torch.cuda.is_available():
    DEVICE, DTYPE = "cuda", torch.float16
else:
    DEVICE, DTYPE = "cpu", torch.float32  # i dont have a gpu :// videos took 4.5 hours to produce

print(f"Running on {DEVICE} with dtype={DTYPE}")
```

2. Stable Diffusion Pipeline Loading

Load the Stable Diffusion 2 model from Hugging Face using the Diffusers library. Select torch_dtype based on device (float16 for GPU, float32 for CPU). Enable attention slicing for memory optimization.

```
# Install diffusers pipeline (model stabilityai/stable-diffusion-2)
pipe = StableDiffusionPipeline.from_pretrained(
    "stabilityai/stable-diffusion-2",
    torch_dtype=DTYPE,
    safety_checker=None
).to(DEVICE)
pipe.enable_attention_slicing()
```

3. Text-to-Speech (TTS) Setup

Use the TTS.api library with the LJSpeech VITS model to generate speech audio on CPU.

```
# TTS install
tts = TTS(model_name="tts_models/en/ljspeech/vits", gpu=False)   # I chose this because work fast in  CPU and it also supports text-to-speech
```

4. Data Definition

Define prompts, output file names, and narration texts for 9 locations across Turkey. First two images/audio may already exist and will be skipped.

```
data = [
    ("Istanbul Hagia Sophia at sunset, ultra-realistic",
     "ayasofia.png",
     "You are now seeing the fascinating picture of Hagia Sophia and it is truly magnificent.",
     "audio1.wav"),
    ("Sultanahmet Mosque with Ottoman Janissaries, cinematic style",
     "sultanahmet.png",
     "Sultanahmet Mosque and the Janissaries in front of it create a true Ottoman atmosphere.",
     "audio2.wav"),
    ("Bosphorus Bridge glowing at night, city lights reflecting",
     "bosphorus.png",
     "The Bosphorus Bridge glows brilliantly at night, its lights dancing on the water.",
     "audio3.wav"),
    ("Cappadocia hot air balloons over rock formations at dawn",
     "cappadocia.png",
     "Hundreds of hot air balloons rise over Cappadocia's unique rock formations at dawn.",
     "audio4.wav"),
    ("Pamukkale terraces with steaming pools at sunrise, surreal scene",
     "pamukkale.png",
     "Pamukkale's white terraces filled with steaming pools glow softly under the morning sun.",
     "audio5.wav"),
    ("Antalya coastline from above, clear turquoise waters and old town",
     "antalya.png",
     "Antalya's coastline features clear turquoise waters lapping against its historic old town.",
     "audio6.wav"),
    ("Ephesus ancient ruins with Celsus Library facade, sunlit",
     "ephesus.png",
     "The sunlit facade of Celsus Library stands proudly among Ephesus' ancient ruins.",
     "audio7.wav"),
    ("Mount Nemrut summit with giant stone heads, sunrise panorama",
     "nemrut.png",
     "At Mount Nemrut's summit, gigantic stone heads gaze over a breathtaking sunrise panorama.",
     "audio8.wav"),
    ("Rize tea plantations on rolling hills under misty skies",
     "rize.png",
     "Rize's rolling hills are carpeted with lush tea plantations beneath misty morning skies.",
     "audio9.wav"),
]
```

5. Core Functions

5.1 generate_image(prompt, filename)

Generates an image with the pipeline if the file does not exist, then saves it to the given filename. Skips generation if the image already exists.

```python
# Image creating (if exist skip)
def generate_image(prompt, filename):
    if not os.path.exists(filename):
        img = pipe(
            prompt,
            height=512,
            width=512,
            num_inference_steps=30,
            guidance_scale=7.5
        ).images[0]
        img.save(filename)
        print(f"Saved image: {filename}")
    else:
        print(f"Image exists: {filename}, skipped.")
```

5.2 generate_tts(text, filename)

Generates speech audio using TTS if the file does not exist, then saves it to the given filename. Skips if the audio already exists.

```python
# creating TTS(if exist skip)
def generate_tts(text, filename):
    if not os.path.exists(filename):
        tts.tts_to_file(text=text, file_path=filename)
        print(f"Saved audio: {filename}")
    else:
        print(f"Audio exists: {filename}, skipped.")
```

5.3 generate_videos()

Creates three videos, each composed of three segments. Each segment pairs an image with its narration, concatenating clips.

```python
# create videos
def generate_videos():
    # Step 1: Create video and voice / or skip
    for prompt, img_fn, txt, aud_fn in data:
        generate_image(prompt, img_fn)
        generate_tts(txt, aud_fn)

    # Step 2: 3 video, each one 3 clip
    for vid_idx in range(3):
        clips = []
        # triple segments: vid_idx*3 .. vid_idx*3+2
        for j in range(3):
            idx = vid_idx*3 + j
            img_fn = data[idx][1]
            aud_fn = data[idx][3]
            if os.path.exists(img_fn) and os.path.exists(aud_fn):
                audio_clip = AudioFileClip(aud_fn)
                clip = ImageClip(img_fn).set_duration(audio_clip.duration)
                clip = clip.set_audio(audio_clip)
                clips.append(clip)
            else:
                print(f"Missing for video{vid_idx+1}, segment{j+1}: {img_fn if not os.path.exists(img_fn) else aud_fn}")
        if clips:
            video = concatenate_videoclips(clips, method="compose")
            out_fn = f"video{vid_idx+1}.mp4"
            video.write_videofile(
                out_fn,
                fps=24,
                codec="libx264",
                audio_codec="aac",
                temp_audiofile=f"temp_audio{vid_idx+1}.m4a",
                remove_temp=True
            )
            print(f"Generated: {out_fn}")
```

- **Loop over 3 Videos**
  Iterate vid_idx from 0 to 2 to
  produce video1.mp4, video2.mp4, video3.mp4.

- **Build Each Video**
  - For each video, gather three segments
    (indexes vid_idx*3 to vid_idx*3 + 2).

- Check that both image and audio files exist before creating a clip, otherwise log a warning.

- **Create Segments**

  1. Load audio: AudioFileClip(aud_fn)
  2. Display image for audio duration: ImageClip(img_fn).set_duration(audio_clip.duration)
  3. Attach audio: clip.set_audio(audio_clip)
  4. Append to segment list.

- **Concatenate & Export**
  - Use concatenate_videoclips(clips, method="compose") to merge up to three segments.
  - Export as video{n}.mp4 with H.264 (libx264) and AAC (audio_codec="aac"), handling temporary audio files automatically.

6. Execution

Run the script using:

```
python GenerateVideo.py
```

The generate_videos() function performs all steps.

```
if __name__ == "__main__":
    generate_videos()
```