# System Design Document for Motion Transfer and Lip-Sync Pipelines

## 1 Part 1: Simple Pipeline for Motion Transfer and Lip-Sync

### 1.1 About the Simple Pipeline

The simple pipeline is designed to infer motion transfer and lip-sync outputs using custom images, driving videos, and pre-generated audio files. This pipeline includes three models:

- **AdaSR Talking Head** for motion transfer
- **LivePortrait** (alternative model for motion transfer)
- **MuseTalk** for lip-sync with custom audio files

#### 1.1.1 AdaSR Talking Head Model

**Inputs:**

1. A driving video (typically a talking face video of a person facing the camera, preferably till shoulder height)
2. An image source (typically a portrait of a person facing the camera, preferably till shoulder height)
3. (Optional) An audio file (required only if you want to run the MuseTalk model consequently)

**Instructions:**

- Upload custom files to the `Demo` directory and modify the paths in the `run_demo.sh` file.
- Run the `run_demo.sh` script to generate the motion-transferred video, which will be saved in the `AdaSR-TalkingHead` directory.

### 1.1.2 LivePortrait Model

**Inputs:**

1. A driving video (typically a talking face video of a person facing the camera, preferably till shoulder height)
2. An image source (typically a portrait of a person facing the camera, preferably till shoulder height)

**Instructions:**

```
python inference.py -s /path/to/your/image -d /path/to/your/driving_video
    --no_flag_pasteback
```

### 1.1.3 MuseTalk Model

**Inputs:**

1. The motion-transferred video from one of the above models.
2. An audio file (should be uploaded in the Demo folder of the AdaSR-TalkingHead directory).

**Instructions:**

- Run the `infer.sh` file to generate the final lip-sync result.
- The first inference will take longer as it creates and stores appearance-volumetric features for faster subsequent inferences.
- Results are saved in the MuseTalk directory's results folder. Delete the results folder or the avatar from the directory when working on a different person.

## 1.2 Discussion

**Limitations:**

- MuseTalk's output suffers from a "blurry mouth" issue.
- AdaSR is used over LivePortrait despite its lower resolution due to its versatile nature.
- LivePortrait may produce unusual mouth and cheek movements if the source image does not have an open mouth or visible teeth.
- The current pipeline supports shoulder-length portraits, which may look unrealistic compared to waist-length portraits.

**Potential Solutions:**

- Fine-tuning MuseTalk on a high-resolution dataset could address the "blurry mouth" issue.

- A fine-tuned MuseTalk combined with LivePortrait can achieve high-resolution talking heads.
- Using waist-length portraits can make the blurry mouth less evident. A hack involves cropping the face, processing it, and stitching it back to the waist-length portrait.

Please refer to the Readme file in the submitted GitHub repository for detailed inference instructions.

# 2 Part 2: Human Talking Heads

## 2.1 About Human Talking Heads

The Human Talking Heads project provides a complete UI integration to infer results through conversation. Note: The pipeline does not include a motion transfer model, as the motion transfer using AdaSR should be performed beforehand.

**Prerequisites:**

- Use the Simple Pipeline to create motion transfer videos of custom characters.
- Follow the instructions in the Readme file from the submitted repository for the installation of Django and other dependencies.
- Create a Groq API key from Groq Cloud and replace the one in `therapy/api_views.py` with it.

**Pipeline Details:**
**Inputs:**

1. A source image (upload to `media/pics`).
2. A motion-transferred video (upload to `media/motion_video`).

**Instructions:**

```
python manage.py runserver 7000
```

to host the UI on a local server. The UI is self-explanatory for navigating through the process.

**Models Included:**

- **StyleTTS2** for speech synthesis based on the output from the Groq LLM.

## 2.2 Discussion

**Limitations:**

- StyleTTS2 currently supports only English.
- Live streaming is not implemented.
- Suboptimal models for emotion recognition and speech transcription.

**Potential Solutions:**

- Replace the current PLBERT model from StyleTTS2 with a multilingual PLBERT and train it on a Japanese speech dataset.
- Consider "Hinglish" instead of "Hindi" and train on a "Hinglish" dataset, avoiding the need for a multilingual PLBERT or Devanagari script aligner.
- Implement live frame-by-frame streaming as we achieve 25fps on V100 and advanced GPUs.
- Use existing emotion recognition and speech transcription models from Therawin.

Please refer to the Readme file in the submitted GitHub repository for detailed inference instructions.

# 3 Part 3: Animated Talking Heads

## 3.1 About Animated Talking Heads

**Prerequisites:**

- Create a Groq API key from Groq Cloud.

The Animated Talking Heads project is a modified version of the SillyTavern-Extras extension, based on `tha3` models, and includes:

- Groq API integration
- AllTalk TTS
- Phoneme extraction from the generated audio and mapping them to suitable mouth animations
- Talking animation based on the extracted phonemes

**Pipeline Details:**
**Servers Required:**

1. For AllTalk TTS
2. For the talking head animator and the pipeline
3. For the user interface

**Inputs:**

- AllTalk TTS requires text to generate audio.
- The animator pipeline requires:
    1. An image of the character
    2. Text input for LLM inference

3. The speech or audio file
4. A JSON file with mouth animations and timestamps

**How the Animator Pipeline Works:** Several API endpoints handle various tasks. Refer to the `Animated_Talkinghead_API_endpoints.pdf` document for details.

Primary endpoints:

1. **Generate LLM response from Groq**: Needs a Groq API key, input text from the user, output text from the LLM.
2. **Generate audio and JSON for mouth animation**: Input text from LLM output and attributes like language, voice, etc.; output audio file and JSON for mouth animation with timestamps.
3. **Load character**: Input character image.
4. **Render livestream**: Output live animation of the talking head.
5. **Classify LLM response into emotion classes**: Input text from the LLM; output emotion for animation.

## 3.2 Discussion

**Limitations:**

- Classified emotion remains the only emotion on the character's face throughout the speech.

**How to Fix:**

- Modify the LLM prompt to include an emotion for each sentence in the output, delimited by asterisks (e.g., `*anger*`).
- Parse the LLM output to remove asterisks-delimited words before passing it to the TTS model.
- Along with the audio and JSON for mouth animation, create another JSON with timestamps for emotion changes.
- In the frontend, check for updates in this JSON file. If found, read the timestamps and send an API call to modify the payload to the animator endpoint with the new emotion.
- This changes the character's emotion as the speech progresses.

Please refer to the Readme file in the submitted GitHub repository for detailed inference instructions.

# 4 Part 4: VASA-1

## 4.1 About VASA-1

VASA-1 is a liberal implementation, referencing various research implementations. It comprises two primary parts:

1. **Latent Space Construction using MegaPortraits**
2. **Diffusion Transformer for High-Fidelity Facial Dynamics Generation**

**MegaPortraits Base Model Components:**

1. **Appearance Encoder (E_app):**
   - Takes the source image as input.
   - Extracts facial volumetric features $v_s$ and a global descriptor $\epsilon$.
2. **Motion Encoder (E_mtn):**
   - Processes both the source image and driving frames.
   - Predicts motion representations, including:
     - Explicit head rotations $R_{s/d}$
     - Translations $t_{s/d}$
     - Latent expression descriptors $z_{s/d}$
   - Uses these representations to predict 3D warpings $w_{s\to d}$ and $w_d$ via separate warping generators $W_{s\to d}$ and $W_d$.
3. **Warping Generators:**
   - **First Warping Generator $W_{s\to d}$:**
     - Removes the source motion from the appearance features $v_s$.
     - Maps them into a canonical coordinate space.
   - **Second Warping Generator $W_d$:**
     - Imposes the driver motion.
4. **3D Convolutional Network (G_3D):**
   - Processes the canonical volume.
5. **2D Convolutional Network (G_2D):**
   - Orthographically projects the driving volume $v_{s\to d}$ into 2D features.
   - Predicts the output image $x_{s\to d}$.

**Diffusion Transformer Architecture:** - Vaguely described in the paper; the implemented architecture includes Classifier-free guidance, dropping each of the input conditions randomly during training.

**Training the MegaPortraits Base Model:**

1. Modify the `configs/training/training_config.yaml` to replace the `vid_dir` key value with the path to the large video dataset.

2. Train high-resolution models on a high-resolution image dataset, while base and student models can be trained on video datasets.

Please refer to the Readme file in the submitted GitHub repository for detailed training instructions.

# 5 Conclusion

This system design document provides a comprehensive overview of the pipelines and projects for motion transfer, lip-sync, human talking heads, animated talking heads, and VASA-1 implementation. Each section includes details on inputs, outputs, and instructions, ensuring a clear understanding of the entire process. The potential solutions and limitations discussed offer insights into further improvements and future work.