Multimedia Systems CS507

Final Project Report

Abhinav Jindal & Mattela Nithish
(2016csb1026) & (2016csb1042)

Introduction

The main objective of our project is to score how suitable is the given video clip for the audio clip of a song. This could be helpful in many circumstances, for instance, if a youtube music channel such as T Series wants to predict how suitable is the video for that song and maybe modify the video accordingly so it gives the best score and hence, the most suitable video for that song. We are using various different components from audio and video clips and comparing them together to give the most suitable score. Methodology and results are explained below

Method

First of all, for every music video audio and the video part is extracted to check their compatibility. Various types of components are extracted for both video and audio. The individual components are explained below.

Audio Transitions

For audio, first of all, important transitions to match with scene change in the video are calculated using vocal transitions and tempo changes. For vocal transitions, first the foreground i.e. the speech part from the song audio is extracted and the music part is discarded. Then appropriate thresholding is applied to check for where there are gaps in the vocals as these gaps denote some transitions in the song. We first threshold the obtained speech audio to remove the lower intensity noise and some music obtained and then find appropriate length 0 gaps which may be the points of transitions. We have tested various thresholds for both vocal energy thresholding and empty gap length to get the best output. Hence some of the important song transitions are obtained by this. For others, we use tempo change to get the transitions. Wherever there is some significant change in tempo, we expect it to be a transition point.

Video transitions

Video transitions refer to shot change/scene change in the video. We have used ‘pyscenedetect’ library to get these scene changes. These scenes are then stored to be compared to audio transitions.

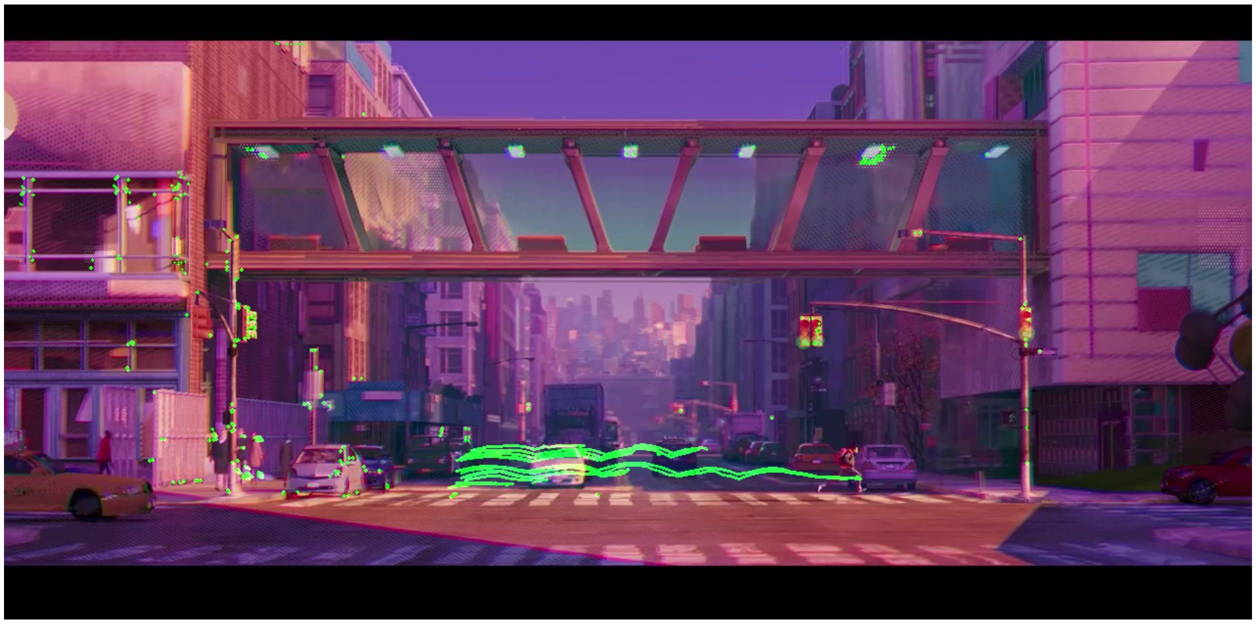
For every audio transition, we check if there is a video transition at that point. We count the number of audio transitions and those transitions points which correctly match in audio and video. We then take their ratio as the score for this comparison.

Audio Pace

We have used tempo as a measure of the pace of audio. Although it is not that accurate, but it is still workable and gives reasonable results. Higher tempo should mean higher speed actions in the video and vice versa.

Video Pace

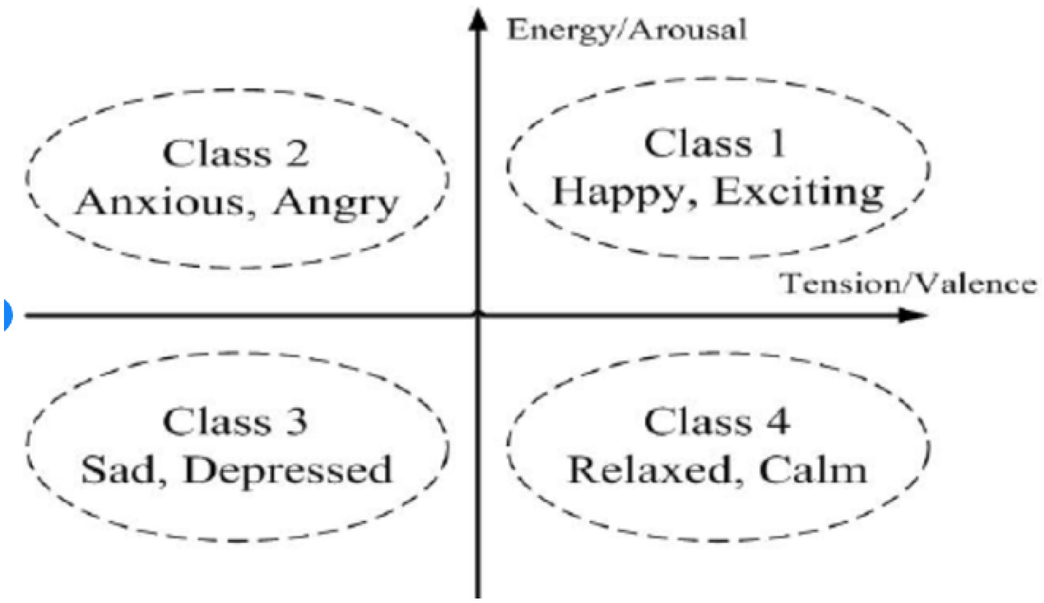
For detecting pace, we have calculated optical flow for each scene separately and compared it with the audio pace. We have calculated sparse optical flow. We first find some important feature points using the Harris detector and then use these points to get the optical flow. We then have used the count of all points detected for the optical flow for each scene and divide it by the number of frames to get the pace score for the video. An example optical flow for a scene is shown in the below image. The green lines show the movement of important feature points in the scene.



The variations from the mean tempo are used to calculate this score. Higher tempo should mean higher video pace or lower tempo for lower video pace for a positive score and else negative score is given.

Audio Emotions

For finding the emotions in song audio we have used Thayer's emotion model which used energy and valence as the criteria for categorizing emotions into 4 categories as shown in the image. For energy, we have used features like intensity and spectral intensity and for valence, we have used features like ZCR, variance, roll-off etc. An SVC model is trained on a small dataset and then this model is used to find the emotions for new songs.



Video Emotions

For detecting emotions of the video we first have face detection to get bounding boxes for faces in each frame and then applied emotion detection on these faces. So basically, we are using facial emotion as video emotion. An example of face detection for an image with happy emotion is shown in the below figure.

The number of matched emotions to the total number of emotions detected is taken as the emotion matching score.

Final Score

For calculating the final score, each of the 3 independent comparing scores is multiplied with appropriate weight to get the final score. The weights are calculated using various testing and according to the importance of each factor in the video audio compatibility.

Results and Observations

We have tested the scores on 2 sets of videos where each set has same song audio and the video varies. Each video is scored and then ranked according to the compatibility with that audio. The 2 sets tested are of 'sunflower' from spiderman and 'whole new world' from Aladin. We observe that our model performs good for both of these sets and rate the videos in the correct order which we thought it should be. As the score is a very subjective opinion for each video as some may prefer some video over the others and the exact scoring accuracy measurement was not possible but we have compared the ranking and checked if the output order was correct. The obtained order appropriately reflect the compatibility order which we had predicted.

Conclusion

We observe that our model performs fairly well for this problem. Although the score cannot be observed in more detail, the order certainly is coming what is expected. The model may be further improved using concept analysis for both song and video. Also some surveys for subjective scores can be used as potential future work on this project.

References

1.https://www.researchgate.net/publication/221370699_An_Efficient_Emotion_Detection_Scheme_for_Popular_Music#pf2

2.https://github.com/danz1ka19/Music-Emotion-Recognition

3.https://librosa.github.io/librosa_gallery/auto_examples/plot_vocal_separation.html

4.https://www.analyticsvidhya.com/blog/2018/12/introduction-face-detection-video-deep-learning-python/

5.https://nanonets.com/blog/optical-flow/

6.https://pyscenedetect-manual.readthedocs.io/en/latest/api/scene_manager.html#scenemanager-example