KODI GSoC Proposal
(Intro/Outro Detection)

Name: Mohit Agarwal
Email Address: mohit15060@iiitd.ac.in
IRC Nick: mohit-0212
Location (City, Country, Time Zone): Delhi, India, UTC +5.30
College: IIIT-Delhi, India
Skype ID: mohitagarwal0212
Hangouts ID: mohit15060@iiitd.ac.in
Contact Number: +91-9873518144
Github Handle: mohit-0212

Possible Mentors: Razze

What is the goal of the project?

The goal of this project is to build a tool which can help the users to detect the intros/outros of their TV show files. Using this tool, users can get the timing and duration of intros/outros for their TV shows which will be stored in a file, which will then be used by a Kodi addon to let users have the option to skip the intros/outros.

What benefits does my proposed work have for Kodi and its community?

Intro/outro skipping can lead to a major improvement in user's watching experience. For example- A lot of people who binge watch TV shows find it really annoying to continuously manually skip the intros per episode and thus such a tool will greatly enhance the user experience. For Kodi, it will add to its already replete set of resources that are available to the users which constantly work towards improving the way a user uses Kodi. Also the tool(API) which will be made will be available for everyone to use, so non-Kodi users and media centres can use that to improve their viewing experience and platform as well.

Why am I interested in working with Kodi and on this coding project?

Kodi community continuously strives to make Kodi the ultimate entertainment center. Being part of such an enthusiastic and passion driven community makes me interested in working with them. This project is an ideal combination of research and development. Research is involved in coming up with the best approach to find a solution to the problem and development is integrating that solution with the Kodi codebase. The project will let me directly interact and learn from the pioneers in the field which will be crucial for my growth as a computer programmer. The frequent discussions and the prompt replies from mentors on the forum and IRC channel are an added bonus. All these incentives motivate me to reciprocate to the community by completing the project.

Project in Detail

Basic Outline of the Steps (Deliverables):

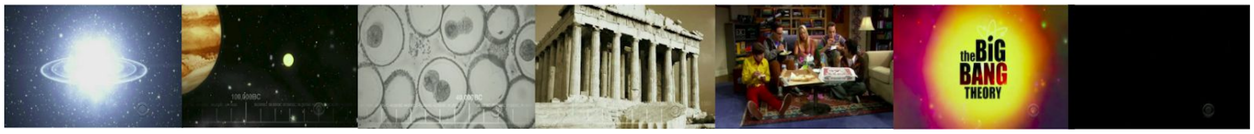
1. Code for audio fingerprinting intro covers and putting them in a database along with their durations.
2. Curate the above database with a small number of TV show intros. (for testing the algorithm)
3. Code to convert the video files to audio, then use it to detect scene changes.
4. Code to match audio fingerprints from a given audio segment to the ones present in the database.
5. We use our recognition code from step 4 on the received segments from step 3 and match where we get our intro cover. From this we will get <u>t1 = intro start time</u> and <u>(t1 + duration) = intro end time</u>. <u>Final segment of the audio is being termed as outro as of now</u>.
6. We store the intro/outro timings that we have received in a file corresponding to the TV episode file.
7. Once the setup above is complete and we start receiving results with a desired amount of accuracy, we proceed on to expand our database to include more TV shows.
8. The above pipeline will then be packed into a REST API and made into a tool which the user can use to get intro/outro timings file by simply pointing to file in the API tool.
9. Code a Kodi addon which exploits this generated timings file and enables a user to skip past intros.
10. User can easily install the addon and start binge watching by skipping the intros/outros. Kodi addon will/may also have the following features and more:
    - Option to skip or not skip the intro as per the choice.
    - Ability to auto skip the intros of next episodes after the user watches the first episode.
    - Send back a sort of feedback if a particular intro is not recognized/skipped, so that it can be added to the database if not present, else be checked for accuracy.

How has the approach developed?

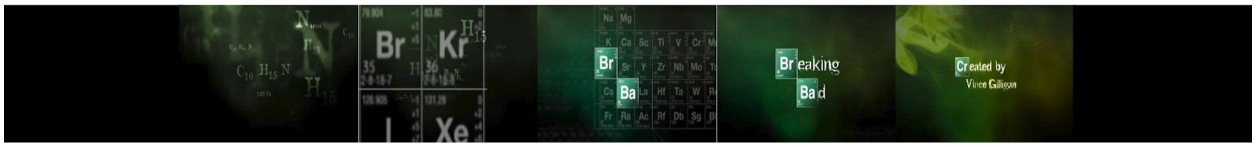
Intro/outro timings can not be statically stored in a file and just be used as it is for skipping as there are plenty of variations of video files present for each episode, therefore the need for such dynamic detection of intros/outros is necessary.

a. Initially, doing literature search for the topic, one of the first papers I came across was https://dl.acm.org/citation.cfm?id=2661714.2661729. This uses the idea of black screen detection which usually occurs every time after intro for a show ends. The implementation of this approach which I did can be found here https://gist.github.com/mohit-0212/31ffa1312b5c6c7a24e9cf9f845b51a3 . Although this approach did fairly well for many shows, it did not work well for some shows which I tested (for example- shows which had black screen even before the intro end time).

Example 1- TV show: Big Bang Theory. It worked fine for this show. As can be seen in some frame shots below, it ends distinctively on a black screen.



Example 2 (fail case)- TV show: Breaking Bad. As seen in the frame shots, intro for this starts and ends with a black screen, thus we can't get a definitive end time. Also there can be no heuristic we can use which generalises over all the TV shows.



Upon discussion with mentor, it was decided to have a show specific model which can give more accurate results, rather than having a single model which is not robust enough for all the shows.

b. The next thing I tried was using title covers present at the end of each intro sequence to detect the timings by comparing it with video frames. My basic implementation of this can be found here: https://gist.github.com/mohit-0212/e907e953458d2eb5bfc49c812b9952f4 . This approach shows our shift from single model to show based model approach. It worked fine on few of the shows I tested. But the problem with this is that a change in the title cover of the show which can usually happen may lead to a change in the results. An example of such a show ''One Piece' suggested by the mentor which usually has different title covers in different episodes can be seen below.

Such major changes in title cover can drastically alter the results of the intro timings.

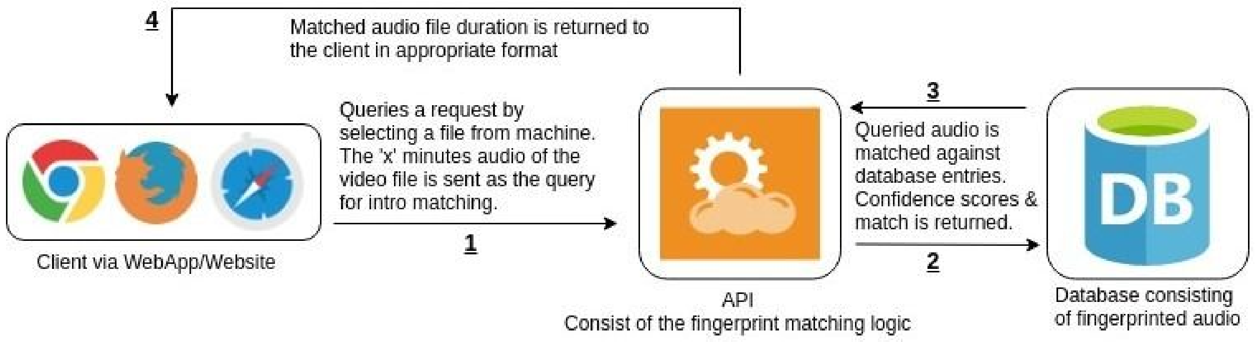
Problem with title cover led to the idea of using an entity of the show which doesn't (or rarely) change over episodes. Such an entity is the intro music sequence. To test it currently, I have used an available audio fingerprinting library for python here https://github.com/worldveil/dejavu . Implementation for this is here https://gist.github.com/mohit-0212/a1f8336bee540ca91540690341aca04a . Using this on several tests, I have seen that as soon as the intro sequence starts in the video file, it gives out the correctly recognized intro song name from the database continuously over multiple iterations

with a high confidence score. For other sequences where there is no song, the output recognized song is although incorrect, but with a very low confidence score.
Now using the ideas from (a), (b) above and implementing them in the case of audio, we can use amplitude variation and variance to get the points of scene changes. We can fine tune this to get better results. Although there might be some false positives for scene changes, we can get almost 100% result for true positives. From these 100% points for scene change, we can use recognition algorithm to get the starting time of the into The other false positive points will anyway not be recognized with a good confidence score. So this way we can get intro start and end time.
Metrics like accuracy, False Positive Rates (FPR), True Positive Rates (TPR) will be used to check for the output of our algorithm. Also to ensure best performance, output will be tested for files with poor quality audio or audio with background noise (collected from sources or added separately). This is to guarantee the performance of our algorithm on different types and quality of audios. Also since we will get a confidence score for each match, we can say with how much accuracy we can guarantee that match and thus use the confidence score accordingly.

Currently on discussion with the mentors, we have two approaches in mind to use the timings generated (which may change later altogether on further research):

- Using EDL (Edit Decision List) files- EDL files contain information about edits to be made to the video during playback. Addon loads this file just as we play the video and performs skip action on the given time duration. Benefits are:
    - easy to use
    - file created is independent of other entities i.e. does not affect other files
    - might be easier to use with the REST structure where the user simply receives a file which will be used by addon
- Creating video tags/editing video metadata- Tagging videos such that specific duration are marked as intros/outros. Benefits are:
    - possibly lesser overhead file cost
    - more tightly coupled with the video file
    - kodi addon has to simply look at video metadata or tags

The above discussion focussed mainly on the algorithm and output portion of the project. Now for the REST API to be made, I have drawn out the following basic flow of the tool:


4
Matched audio file duration is returned to the client in appropriate format
Queries a request by selecting a file from machine. The 'x' minutes audio of the video file is sent as the query for intro matching.
1
3
Queried audio is matched against database entries. Confidence scores & match is returned.
2
DB
Client via WebApp/Website
API
Consist of the fingerprint matching logic
Database consisting of fingerprinted audio

This flow process from query generation by the user to receiving the output will be improved as the project progresses. One such improvement which I can think of currently is rather than sending ‘x’ minutes in step 1, we send ‘t’ seconds of audio per scene change point which we have detected, this can substantially reduce the amount of query data we send to server. (Crude implementation of the above idea: https://github.com/mohit-0212/Intro-Detect)

Apart from figuring out the algorithm and flow for the REST API, I have compiled Kodi on my Ubuntu 16.04 machine from source and have had a look at different code files and addon files. I downloaded and used several addons from inside the Kodi environment to figure out how an addon is used.
I also looked at an addon structure and how can it be made. A basic addon can contain an xml file, a python file and a resource directory. This tool https://github.com/Razzeee/generator-kodi-addon from Kodi sets up a basic addon instantly. I have set this up on my machine and used it to get a sample addon. The readme file generated also contains several instructions on how can one proceed to add code to the addon. Overall with the combination of correct tools and mentor guidance, an addon for our standalone tool can be simply made.

The approach is subject to changes to make it more fine which can be done by further discussion with the mentors during the community bonding period and later on during the coding period as the project progresses.

What does it touch in Kodi?

For the development of the tool(API), this project does not depend on Kodi codebase. But for the python addon development which makes use of the tool, dependencies like *xbmc*, *xbmcgui* etc will need to be imported. It does not involve messing up with Kodi’s core codebase, thus it can be guaranteed of completion with minimal issues from Kodi core framework.

Overall the code will be made both python 2 and 3 compatible. Ffmpeg will be used for extracting and using audio. Basic python libraries like numpy for array operations will also be used. Pep8 style guidelines will be followed as strictly as possible, external tools can be used to ensure it. Although I have used many REST APIs personally and have made a simple HTTP application using GET/PUT operation for images using python flask, I will need more reading into how to make it up for a bigger scale like this. I am confident it won’t be an issue for me and I’ll be able to handle it.

What can be some “if time deliverables” or “extensions of the project”?

1. Making the above tool work from inside Kodi, thus reducing the user involvement and making it even more easier to skip intros.
2. In addition to skipping for TV shows, extend it to detect and skip ads from recorded TV.
3. Apart from ads, it can also be extended to filter out ‘unwanted’ content for children by using video frames and make a parental check or child section feature.
4. While figuring out the workflow for my implementation, I was told by mentors that renderCapture() in core Kodi is going to be deprecated due to poor maintenance. I wish to work on it and extend it for audio as well. This will enable us to get audio/video frames

from a file dynamically and then use them for the above tasks. This will make the process even more simpler for the user. As it can be a full project in its own, I wish to work on it after SoC period ends and thus be a continuous contributor to Kodi.
Also, if other ‘if time’ deliverables are not done within the SoC period, I plan to contribute to Kodi after that period by working on them and making my contribution even more holistic.

How do I plan to achieve completion of my project?
Before the community bonding period begins, I plan to read more about audio fingerprinting, how it works and how can it be efficiently utilized. Also I plan to read and learn how to setup and build up a full REST API system and how can it be made for our usecase.

Pre Coding Period - Community Bonding (April 23 - May 14)

Duration	**Expected Outcomes**
April 23 - April 30	Understand more about the codebase, the database structure used for Kodi and how to set it up.
May 1 - May 7	Discuss about addon development, know its finer details. Try coding out some basic addons.
May 8 - May 14	Revisit audio fingerprinting, REST APIs. Decide on the final approach after discussion with the mentors.

The project comprises of 3 main parts:
- Algorithm for audio fingerprint creation, matching
- Bundling the above in a REST API
- Integrating the generated file via API in Kodi

So I have split 3 months of my coding period roughly proportionately in the above tasks with slightly more time given to the algorithm.

First Month (May 14 - June 15)

Duration	**Expected Outcomes**
May 14 - May 21	Set up a local database. Code for audio fingerprinting.
May 22 - May 28	Curate a smaller dataset of TV show intros, use fingerprinting on it and add it to the database. Code to get audio from video files.
May 29 - June 4	Code to detect scene changes.

	Use the audio to detect scene changes and optimizing it to get all true positives and minimum false positives.
June 5 - June 11	Code for audio fingerprint recognition.
June 11 - June 15	First evaluation

Second Month (June 15 - July 13)

Duration	Expected Outcomes
June 15 - June 23	Test the recognition algorithm on new audio files and fine-tune it. **Expand the database.
June 24 - July 1	Compile the above in a user friendly form where the user can easily point the tool to the directory and a timings file is generated on its own. Set it up as a REST API. Start to code for Kodi addon with sample intro-outro file generated. **Expand the database.
July 2 - July 9	Complete REST API framework. Code for Kodi addon. **Expand the database.
July 9 - July 13	Second evaluation

(**Since the database can be big, the process of building it up will be carried out gradually over the weeks, rather than spending whole week on it.)

Third Month (July 13 - August 14)

Duration	Expected Outcomes
July 13 - July 21	Complete the Kodi addon along with its interface, i.e how will the user end up using it inside Kodi.
July 22 - July 29	Buffer period or time for if time deliverable (1,2).
July 30 - August 6	Code clean up, bug testing, documentation.
August 6 - August 14	Code submission and Final evaluation

When did you first start programming?
I first started programming in 6th grade of school. I started programming with QBasic, then I tried Visual Basic. In my high school I was taught Python and it has been 5 years for me using Python since then. Due to my interest in computer science since my early years in school, I

decided to pursue it further for my higher studies. In my freshman year of my college, I used C to learn the concept of pointers, memory, data structures. Later I used Java to get familiar with Object Oriented Programming. Along with these, Python was in constant use for machine learning, data science projects, basic scripting tasks.

What projects have I worked on previously?
I have worked on several projects in my college duration. Some of my major project works are:

1. *Animal Biometrics (Part of ongoing Undergrad thesis)*- It deals with detecting animals in the wild, identifying the type of animal and recognizing the animal by identity if present in the database. It can in brief be described as the FaceID for animals. I have started with primates and tigers initially. If done successfully, this project can have a huge impact as it can enable wildlife preservers to keep track of animals without using invasive technologies like animal tagging. Non deep learning methods like Haar, LBP were used to get baseline results but a proper deep learning architecture is used to get good results. This project is part of the Image Analysis and Biometrics (IAB) Lab, one of the most well established research labs here. (http://iab-rubric.org/)
Mentors: Dr Mayank Vatsa, Dr Richa Singh
Technology Used- Python and libraries

2. *Gaze Detection and Tracking*- Using geometry, transformations and proper computer vision methods to detect gaze and track it. The problem of gaze tracking is difficult due to various reasons like head pose variation, background occlusion. This project tries not to employ deep learning architectures and instead use traditional vision techniques to get a better understanding of the math and geometry behind them. The output of the tracking system is then used on an application like a small driving game or as a mouse pointer.
Mentor- Dr Saket Anand
Technology Used- Python and libraries

3. *Part-wise information retrieval for products using Amazon Reviews*- It involves getting part-specific reviews/ratings for products. So, for example, if we search for laptops on Amazon, it would be really nice to have part specific ratings available for that product like the display (4.5 stars), battery (3 stars) etc instead of going over all the customer reviews manually. This project focuses on this problem with a subset of such products.
Mentor- Dr Tanmoy Chakraborty
Technology Used- Python and libraries

4. *Working with multi-heuristic A* (MHA*) search*- MHA* algorithm was given by Dr Sandip Aine et al in the paper http://www.roboticsproceedings.org/rss10/p56.pdf . This project implements this algorithm on 2d graph and problems and makes comparison with the well known A* search. It was tested and compared on a large enough 2d graph (~400 nodes), tile sliding puzzle and taxi-driver problem (done using Google Maps API). The project code and report can be found here https://github.com/yashasvi97/AI-Project.

Mentor- Dr Mayank Vatsa
Technology Used- Python and libraries

5. *Cancer severity prediction-* This was a core machine learning project done on a Kaggle dataset. It involves extracting useful features from a large medical analysis text per sample and combining it with genetic variants and mutations available to output a severity class label from 1-9.
Mentor- Dr Saket Anand
Technology Used- Python and libraries

6. *Analysing Facebook and Twitter handles of different Policing accounts across the world-* It collects data from social media handles of different policing accounts using FB and Twitter REST APIs and this data is used to do different analysis like type of content (media or text) posted by them, user reaction on posts done, what type of post has the most positive/negative response, things which are done differently by one police handle over the other etc.
Mentor- Dr Ponnurangam Kumaraguru
Technology Used- Python and libraries

7. *DBLP Query Engine-* DBLP is a huge database for academic publications and papers. The whole database for this is available in an XML format. The project implements a GUI interface to query through the whole XML database with relevant filters and entity resolution. The project code and report can be found here https://github.com/akhil15126/DBLP .
Mentor- Dr Chetan Arora
Technology Used- Java and libraries

8. *Medical Records System Implementation-* The full database structure for medical records was developed from scratch from ER diagrams to functional dependencies. It was then created using MySQL and populated using a sample dataset. A GUI interface made using Java was used to interact with the MySQL database. The project code and diagrams can be found here https://github.com/mohit-0212/Medical-Records-Database .
Mentor- Dr Vikram Goyal
Technology Used- MySQL, Java and libraries

9. *Mobile Application for Sharing-* This is an Android mobile application allowing users to put up items for sharing and borrow them and communicate with another user for mutual needs. This project was decided with the aim to help my college students who usually have small, regular requirements on a daily basis.
Mentor- Dr Pushpendra Singh
Technology Used- Java, Android Studio

10. Some other side coding assignments and projects include, simple shell tool (using C); telnet client (using C); client-server CLI based communication (using C); webapp to detect and recognize faces of two Indian politicians in images (using Python and Flask); gesture controlled

car with camera streaming (using Arduino and Raspberry Pi); machine learning and feature extraction algorithms like Naive Bayes, TF-IDF, Neural Networks etc coded from scratch; etc.

What are my past experiences with the open source world as a user and as a contributor?
As a user, my work is surrounded with a range of open source applications from a simple text/code editor to complex machine learning tools like PyTorch.

As a contributor, I'm familiar with basic git commands and have also used command like squashing of different commits to combine them into one meaningful commit. Apart from committing to my own project repositories, I have done a successful PR and commit to INCF python-odml repository and also successfully completed the Hacktoberfest 2017 challenge. I use Github for version control and managing my projects whenever necessary.

What are the computer(s) and devices I have available for working on the SoC project?
I have got a decent laptop with Intel i7 5th generation processor, 8GB DDR3 RAM, 2GB Nvidia Graphics Card. I use Ubuntu 16.04 for all my tasks. Along with this, I have a Dell 22" LED monitor to help and ease the multi-tasking and development process.

Are you a user of Kodi? When did you first start using it? What do you primarily use Kodi for?
I have recently become a Kodi user. I started using it from the start of February and the fact that it is open to the developer community to continuously add on to what it already provides keeps me hooked to it. I have majorly used it to watch few movies and TV shows; and I really liked how easily I could move through the whole GUI or how easily I could add subtitles to what I was watching.

How much time will I be able to contribute? Do I have any other commitments?
My college vacations from start of May till end of July aligns perfectly well with the core GSoC coding period, so I would have no other study or job (or internship) to do. I will be completely available to dedicate all of my time towards this project. Since I have no other commitments, I can contribute 40-50 hours of work per week towards the project. I can always be accessible to the mentors at any decided time as per their convenience.