University of Southampton

School of Electronics and Computer Science

A Group Design Project report submitted for the award of
MEng Computer Science

by

Michael Harris (mh27g08)

Jonathan Harrison (jh31g08)

Samantha Kanza (sk11g08)

Jennifer Lantair (jl17g08)

Project Supervisors: Paul Lewis

Second Examiner: Enrico Costanza

Summarising Audiovisual Information

Date: December 15, 2011

Abstract

The BBC has an ever increasing archive of broadcast audiovisual footage that dates back over fifty years, which is stored for the purposes of content preservation and ensuring that the BBC's high standards are maintained. There is limited information attached to their archive causing the contained data to be difficult to summarise, hard to effectively classify and reuse. The lack of effective automated methods to allow for such summarisation costs the BBC a large amount of resources due to the manual processing involved, which is time consuming and allows mistakes.

In order to assist with the solution, the team's aim is to develop a piece of software which allows users to process and summarise a collection of material by applying video, audio and natural language processing techniques to produce a summary as well as XML files containing information about the collection based on a users criteria.

The report details all of the aspects of the project, such as how the team tackled the problem together, an in depth analysis of the software and the team's performance, as well as discussions upon the future work which could be done as an extension to the project.

Acknowledgements

We would like to thank Paul Lewis and John Hare for their time, assistance and advice throughout our project. Thanks must also be given to our client Matthew Addis who has managed to meet us regularly and examine our work as we progressed, despite a busy schedule. Finally we would like to thank the OpenIMAJ team who have provided advice and support in using the OpenIMAJ library and helping us to tackle the difficult problems of video and audio processing.

Contents

1 Project Description

1.1 Introduction

The British Broadcasting Corporation (BBC) has a huge archive of millions of hours of audiovisual data which they have broadcasted over the past eighty four years. It is very difficult to effectively manage an archive of that size and it would be unfeasible to attempt to process all of the data manually. There is great potential to re-use this material in a multitude of ways such as building new programme material from existing content or reselling and re-broadcasting past programmes. However, currently any manipulation and analysis of this data has to be performed manually as there is no automated system in place to aid the BBC archiving department. These obviously cost both time and money, therefore the archived material is unable to be used to its full potential.

Such an issue costs the BBC a significant amount of resources and the automation of these functions should enable the archiving department to not only to save a significant amount of resources, but to fully utilise the material they have available. The real world issues involved with the project and the non-trivial nature of cataloging audiovisual data is what motivates the team to set goals such as the full automation of these processes.

1.2 Project Problem

The system is primarily aimed at people working within the BBC who need to summarise their archived content in order to do activities such as intellectual property (IP) rights assessment or finding unbroadcast footage of specific programmes. However, it is also aimed at potential customers when the BBC want to sell a television series to another company. The team's customer has provided us with three potential scenarios that they wish us to consider when working out the systems goals. The provided potential scenarios are as follows:

- **Archive Monetisation** - The BBC wish to provide short summaries of items/collections from their archived content to prospective buyers. Unfortunately the BBC has limited resources and may not have the time to effectively create summaries for any given set of videos from their collection, an automated process though could potentially prove cheaper in terms of resources and more effective. Summaries need to represent all main elements of the video such as main characters and main action sequences.

- **Rights Assessment** - In order to be able to reuse some of their archived content the BBC need to be able to identify which Intellectual Property rights (e.g for musical pieces, particular actors) need to be cleared before they can do this. However, whilst the BBC have subtitles for their content they do not necessarily include the actors names, so they would like to be able to produce a summary of who is featured in the video content so that they can clear the IP rights accordingly.

- **Archive Content Selection** - The BBC may want to use some of their unbroadcasted archived content to produce extended versions of episodes or films. An automatic analyser of the content that is able to identify the range of topics/locations that made it into the final cut of the programme would allow an extended edition to be simply created by matching the unseen content appropriately.

The team has decided to examine all of these scenarios as there are obvious similarities between them. An analysis of all scenarios is intended to produce as a comprehensive

system as possible in the available time.

1.3 Project Goals

The projects goals were produced through the use of the provided project brief (Appendix A) and the initial meeting with the team's client. The project goals which the team hopes to achieve are as follows:

- The ability to accept a range of different audiovisual formats as input.
- The ability to detect shots in video.
- The ability to detect when people appear in video.
- The ability to detect where people are mentioned in subtitles/audio.
- The ability to detect where locations are mentioned in subtitles/audio.
- The ability to detect loud sections in audio.
- The ability to detect music segments in audio.
- The ability to identify key shots in video.
- The ability to identify main characters.
- The ability to identify main locations.
- The ability to identify music segments.
- The ability to summarise input files using user selected metrics.
- The ability to export summary data produced.
- The use of IAM's library OpenIMAJ.

1.4 Final Project Scope

The scope of the team's project is to deliver a polished, user friendly piece of software that may be used easily by a non-technical client. The project shall incorporate all the initial goals shown above and any new potential goals that emerge throughout the project.

2 Project Dynamics

Before commencing work upon the project, the team has discussed and finalised how they intend to operate as a unit which includes a software methodology to follow, a set of common software tools and meeting dynamics.

2.1 Team Member Strengths

The strengths of each team member have been identified and are summarised in the following table:

Team	Strengths
Michael Harris	Coding, LaTeX and leading
Jonathan Harrison	Note taking, organisation, time keeping, web design and coding
Samantha Kanza	LaTeX, organisation, web design and coding
Jennifer Lantair	Coding, LaTeX, proof reading and organisation

Table 1: Team Member Strengths

2.2 Job Roles

Using each team member's strengths as described in Section 2.1, as well as knowledge from previous experiences together, the team members have been assigned the following roles. These roles are not all inclusive, that is to say that no role is intended to describe all of a team member's work. The roles are laid out in the following table:

Team	Roles
Michael Harris	Project Manager
Jonathan Harrison	Secretary and Time Manager
Samantha Kanza	System Architect
Jennifer Lantair	Documenter

Table 2: Team Roles

In addition to these roles each team member shall also be working upon the tasks involved with each stage of the project such as the system implementation and documentation. Each team member is intended to be working for approximately equal time periods over the ten weeks of the project to ensure fair division of the work load.

2.3 Software Model

The project will be completed in a relatively short time scale (only ten weeks) therefore it calls for a rapid iterative software engineering process to ensure that no time is wasted. Therefore we have decided to go with the Iterative Waterfall model; the basic processes and the way the feedback works is shown in the following figure:

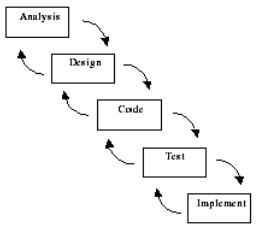

Figure 1: The Iterative Waterfall model

The Iterative Waterfall model, which is a variation of the original waterfall model, takes into account the work of Winston Royce [27] who advocated the use of feedback in the waterfall model. The Iterative Waterfall model allows feedback to be generated in such an area as design and to step back a stage to the requirements if necessary. The ability to repeat a stage was believed by Royce to produce superior products. Royce was also a great believer in the power of strong contacts with a team's client. Such strong ties with the client meets the teams needs well because the production of a superior product which satisfies the client's requirements is desired.

2.4 Project Tools

The tools that the team have chosen to use throughout the project are detailed in the following Section. Tools chosen must be platform independent due to the different environments that each team member uses and they are familiar with them so that work can begin immediately with no additional learning curve.

2.4.1 Communication Aids

The ability for the team to communicate effectively is very important, therefore the following tools were chosen in order that members could be contacted at all times:

- Email
- Google Chat/Talk
- Mobile Phones

2.4.2 Project Management

Project management tools are used as a means to manage elements of the project such as task and time management as well as ensuring all team members know when and where meetings and group work sessions shall occur. The following project management tools were selected:

- Google Calendar
- Microsoft Project

2.4.3 Documentation

Documentation tools are those used to produce documentation during the project. The following documentation tools were used:

- Google Documents
- LaTeX Editors
  - Kile
  - TexWorks
  - BibDesk

2.4.4 Implementation

The implementation tools which shall be used are as follows:

- Maven
- Eclipse with the following plugins:
  - M2Eclipse (Maven plugin)
  - SVN Team Provider (SVN plugin)

2.4.5 Version Management

A number of version management tools were considered for managing the teams documentation and coding bases. Those chosen were the following tools:

- UGForge
- Google Documents

2.5 Meetings

The team has organised their internal and external meetings in the following sections. By organising these meetings the team hopes to ensure a structured environment for their project.

2.5.1 Client/Supervisor Meeting

The team intents to meet their supervisor and client at least once a week. These regular meetings will last a maximum of an hour and allow the team to discuss and evaluate current progress and tasks (which includes any problems that the team are having) and to plan and approve future work that will be done for next meeting.

The evaluation of progress and work during these regular meetings means that the team will get immediate feedback on their work ensuring that work meets the customer's expectations and requirements. These meetings lend themselves to the chosen software model detailed in Section 2.3.

2.5.2 Team Meetings

Beside Client/Supervisor meetings, the team shall have regular private meetings to allow the team to judge their current progress against their predicted progress as decided upon during the initial week of their project (see Appendix B). By doing this it will gauge whether the team are on track for their deadlines or if they are behind. In case of slippage the team may at these meetings call into effect their contingency plans which will place them back on schedule.

These meetings shall last a maximum of an hour and will take place at the start and end of each week. Meetings at the beginning of the week shall occur on Mondays and will contain discussions about what the team has done over the weekend, plan the weeks goals and how they are to be divided and completed.

Finally, end of week meetings shall be every Friday and will follow Client/Supervisor meetings to allow for further discussion of points made in the previous meeting as well

as plan work to be done over the weekend.

2.6 Work Techniques

The techniques that team members will be using whilst working on tasks throughout this project are detailed below.

2.6.1 Logbooks

Logbooks will be used by each team member to keep a record of their involvement with all areas of this project. Firstly, logbooks shall be used to plan activities, for example drawing out user interface designs and planning the implementation of modules.

As well as planning, while undertaking tasks the logbooks will be used to detail the activity being done, the team members involved with that activity and the number of minutes spent upon it. This is in order to help with keeping track of the project's progress and to accurately update the projects Gantt chart.

Finally logbooks should also be used as a private record of an individual's thoughts throughout the project in order that they can be used as reference when team members write their individual reports about the project.

2.6.2 Peer Review

Throughout the project, the team will peer review any completed tasks whereby team members besides those who completed the task will review the completed work in great detail in areas such as content and how it is written. Therefore any errors or inconsistencies will be easily spotted and can be corrected. Having these reviews ensures that work is completed to the same standard and is of the highest possible quality.

2.6.3 Pair Programming

During the implementation of the project where appropriate the team will use the pair programming technique, where on one workstation one team member enters the code and the other team member overlooks the activity. Using this technique has a number of advantages:

- It's shown to reduce the number of man hours required to complete a project and it's size [15].
- It significantly decreases the likeliness of procrastination and the amount of time wasted on procrastination [34].
- For a project where no individual team member has had prior experience in the subjects associated with this project e.g. video and audio processing. It provides benefits where the solution to a problem is unknown to the developers; two heads are better than one.
- Two team members will have an in depth understanding of any area of code.

2.7 Project Management

The tools that the team members will be using to manage the project have already been stated in Section 2.4, how we are going to use them are detailed below.

2.7.1 Project and Task Management

To successfully manage the project and to keep track of the progress of the project as a whole, a Gantt chart was produced using Microsoft Project 2010 and can be viewed in Appendix B. The Gantt chart contains:

- The project stages as per our Software Model (see section 2.3).
- All the key tasks that are a part of this project.
- Milestones which include:
  - Project deadlines such as Progress Seminars and Report Deliverables.
  - Project stage milestones such as when the Implementation will be completed.
  - Client/supervisor meetings.
  - Contingency time to allow for the risks identified in Section 8.3.1 to be overcome.

This was updated on a weekly basis throughout the project prior to client/supervisor meetings held at the end of the week.

Using this chart, it allowed the team to evaluate their progress each week to ensure that all tasks were progressing at a reasonable rate and none were forgotten about. Furthermore, it has been used to evaluate our project which is detailed in Section 8.3.1.

2.7.2 Organisation

In order to make sure that all team members had the same understanding of:

- When the project milestones occurred, as detailed in the Gantt chart.
- When and where any meetings were taking place.
- Other module deadlines/clashes that team members had.

We decided to use Google Calendar due to the familiarity everyone had with using this software, it allowed everyone to be kept in sync and also, it could be viewed anywhere and at anytime across varying platforms and devices.

2.8 Programming Styles

There are a large number of different coding practises and styles which we may choose from; the ones which best suited the projects requirements and teams methodology are detailed below.

2.8.1 Modularity

Modular programming is a programming style which breaks down the programs functions into modules, a section of code which performs one function only and contains all of the source code to do so. Following are the advantages gained from producing modular code:

- The software becomes easier to debug and maintain.

- Allows the team to edit modules at a later date without affecting other modules.

To implement this, team members will always be examining functions to see if they may be reduced any further.

2.8.2 Simplicity

The code produced shall strive to be simple and logical to understand in order that all code is easy to read.

2.8.3 Scalability

Due to the projects requirements the system needs to be scalable in order that it will be able to still perform in the same way despite a range of differently sized workloads, from individual short programmes to a whole series of a television show. Therefore, all components of the system will be constructed with scalability in mind and tested using a range of workloads [14].

2.8.4 Extensibility

Due to the short timescale of this project it is unlikely that all requirements of this system will be completed therefore in order to allow for future work the system will be constructed in order to be as extensible as possible. This will be achieved through the use of the previously mentioned styles such as modularity.

2.8.5 Documented

Throughout the implementation of the project the team will continually document their code as they work upon it. This is in order that during development any team member can be interchanged with another and allow for them to quickly understand the code and allow work to continue. Furthermore, it will also help achieve extensibility of project as theoretically people outside the project would be able to work on it.

2.9 Testing Styles

Similar to programming styles, there are a large number of testing styles [23] to chose from therefore we have chosen the best suited testing styles to fit our software engineering model (see Section 2.3).

2.9.1 Unit Testing

The project will be implemented in modules as discussed in Section 2.8.1 therefore as we develop these modules will be unit tested using JUnit.

2.9.2 System Testing

When all modules are fully integrated the team shall conduct these tests manually to ensure there arent any issues.

2.9.3 User Goal Testing

The final product will also be tested to see how it meets the user goals.

2.10 Evaluation Techniques

To evaluate the system produced during the project the team will evaluate the results produced by the modules that are based upon opinion and aesthetics and cant be tested in a machine manner, e.g Name and Location detection, Video Shot Detection, Facial Recognition as shown in 5.1 . To test these modules and the techniques used the following tasks will be performed:

2.10.1 Manual Inspection

Manual inspection will be performed on our test television programmes in order to provide the times taken to do the process manually as well as ground truth data in order to compare and evaluate the effectiveness of the programmatic techniques. Manual inspection will result in a superior product, but it does take a greater number of man hours to complete than an automated process.

2.10.2 Module Testing

Using the same test television episode files which was inspected previously; for each technique used in that module it will be ran on that data and the results produced will be recorded and appropriately structured to allow for comparison.

2.10.3 Manual and Module Comparison

Comparing the ground truth and produced data, a comparison of how effective the techniques used will be discussed including any identified issues causing false positives. Any solutions to these problems will become part of the future work.

2.10.4 Manual Face Comparison & Inspection

The faces extracted from the video will be manually inspected to see how many of them are genuine faces and how many are false positives. These statistics will be compared to ascertain how accurate the facial detection is.

2.10.5 Summary Trailer Evaluation

Unlike testing the independent modules, the final summary produced will be evaluated against how long it takes to produce and more importantly, the content of the trailer whether the data produced by the different modules selects the most representative parts for a television programme. This will be done by comparing existing trailers for television programmes with the one that the team's system produces. An example of this comparison which was highlighted by the team's customer is Top Gear due to the introduction sequence containing a short summary of the entire programme.

3 Project Analysis

3.1 Stakeholder Analysis

Stakeholders can be put into two categories which are primary and secondary where:

- Primary stakeholders are the people and groups who are directly affected by the outcome of the project.
- Secondary stakeholders are people and groups who arent directly affected by the outcome of the project, but still have an interest in it.

Using these definitions, the stakeholders for this project are as follows:

Primary:

- The customer (Southampton IT Innovation Centre).
- The development team.

Secondary:

- The projects supervisor.
- The BBC.
- The School of Electronic and Computer Science
- The University of Southampton.

3.2 Requirements Analysis

3.2.1 Assumptions

The assumptions made by the team whilst constructing the requirements of this project are as follows:

- Input video files will be the "finished" broadcast material e.g no out-takes and clapper boards
- Input subtitle files will be in W3C Time Text Markup language format.
- The subtitle file associated for each TV programme is accurate enough that further speech-to-text analysis from the audio stream of a video is not required.
- The trailer containing plot spoilers for a TV programme is not a priority for the customer based upon the Archive Monetisation scenario (see Section 1.2).

3.2.2 Functional Requirements

The function requirements which the team have agreed upon with the customer are below. They are split into the sections: must have and could have. Use case diagrams of these requirements are in Appendix C.

Must Have

MF1: Ability to import video file(s).

MF2: Ability to import subtitle file(s).

MF3: Ability to breakdown video into video shots.

MF4: Ability to breakdown audio into samples.

MF5: Ability to detect when people appear in video.

MF6: Ability to detect when people are mentioned in subtitles/audio.

MF7: Ability to detect when locations are mentioned in subtitles/audio.

MF8: Ability to detect loud sections in audio.

MF9: Ability to detect music segments in audio.

MF10: Ability to identify main characters using MF5 & MF6.

MF11: Ability to identify main locations using MF7.

MF12: Ability to identify music segments using MF9.

MF13: Ability to identify key shots for trailer based upon MF5 - MF9 and MF14.

MF14: Ability to adjust summary based on:

- TV Genre
- Duration
- Trailer type

MF15: Ability to produce a summary (trailer, data, etc) using MF10 - MF14.

MF16: Ability to export summary produced.

MF17: Ability to view and control everything in system via a GUI.

Could Have

CF1: Ability to determine locations from video.

CF2: Ability to perform speech to text analysis when a subtitle data file isn't available.

3.2.3 Non-Functional Requirements

NF1: Accept formats used by the BBC archive.

NF2: Export summary data in a standard formats (e.g. video in mp4, data in XML).

NF3: Time to process all input and build summary must be significantly less than the time taken to watch it.

NF4: Application must provide feedback (e.g. progress of processing) to the user.

NF5: Application must be multi threaded.

NF6: Build on OpenIMAJ software.

3.3 Risk Analysis

Below are the risks that have been identified by the group which may affect the project. In order to counter these possible risks, strategies have been devised. The columns Likelihood and Impact use a scale from 1 - 5 where 1 is the least and 5 is the maximum [30].

Risk	Likelihood	Impact	Overall Risk	Plan
Project size / goals have been underestimated.	3	5	15	If progress reflects this, discuss with the customer and supervisor to identify requirements which arent key and may be removed in order that the project can meet the scheduled deadlines.
Loss of a team member due to illness or other circumstances.	3	4	12	Throughout the project the team will work in pairs (pair programming) so that at least two people are managing a task. When a member is "lost", the other team member can continue work and if necessary, more members can be allocated to a task if the team sees fit. Also, report the situation to Supervisor.
Designs arent as required and need to be reworked.	3	3	9	An iterative waterfall development process has been chosen to allow changes during the project. As a team review the designs and change them as necessary to meet the project's needs.
Project schedule slips.	4	2	8	Contingency time has been built into the project plan to allow for slippages in progress yet ensure deadlines are met. The focus of team members can be rearranged to help with ongoing delayed tasks.
Unable to find appropriate libraries to complete a requirement.	2	4	8	View current progress and discuss with the customer and supervisor. If there is enough time left in the project and the requirement is necessary, perform research and to the best of the team's abilities construct a solution to complete the task.

Requirements change.	1	5	5	An iterative waterfall development process has been chosen to allow changes during the project. Review changes in the requirements with the Customer and Supervisor. Where possible and if time permitting the team may rework the current work to match these changes.
Unable to meet with Supervisor or Customer.	5	1	5	An email will be sent detailing the current progress of the project and also, to arrange a meeting with the Supervisor or Customer when they are next available.
Loss / damage to teams computers.	4	1	4	Throughout the project work will be backed up in two places. • Documents on Google Docs • Code on UGForge Therefore, any loss/damage to the teams computers is insignificant as work can continue using backed up work using alternative computer equipment.
Tools fail to perform as expected.	2	1	2	Seek out alternative tools which meet requirements.

4 Background Research

4.1 Introduction

The goal of the system is to analyse audiovisual content and to produce a variety of different summaries based upon the user's input. The research focus has been broken down into three main areas. The first area explores the different algorithms used in video summarisation such as general summarisation algorithms, facial recognition and detection and varying audio analysis algorithms. Following that are descriptions of the different techniques used to create audiovisual summaries and then examples of different video summarisation systems.

4.2 Algorithms

There are a variety of different algorithms that have been created for video summarisation, some purely aim to create film summaries out of the given content and others delve further with techniques such as facial recognition and audio analysis to provide more extensive types of summaries. All of these techniques are detailed below.

4.2.1 Video Summarisation Algorithms

The following video summarisation algorithms involve working with rushes video which requires extra processing prior to constructing a summary. This is because rushes video is raw unedited video recorded for television programmes or film which may contain repeated recordings of a single event, clapper boards, colour bars, monochrome frames and other non-important information.

Binary Tree Algorithm
A scalable summarisation algorithm has been created based on the dynamic generation of binary trees. The algorithm firstly considers the incoming video as a block of individual frames which are the basic unit used in this algorithm. These basic units are either classed as "inclusion" or 'discard" which corresponds to whether they are included or excluded in the created summary. The tree is built with an empty root node at the top, and then upon receiving each unit an instance of the two classes is appended to all existing nodes where each branch represents a possible video summary. Through this method all possible summarisations of this video exist within this tree, each leaf node represents a potential video summary from the root node to a specific child node. In order to obtain the optimum summary a score is associated with each leaf node in the tree and the route from root to leaf node with the highest score is the best summarisation possible [33]. This technique would probably be quite time costly for the team's system if fully implemented although the idea of splitting up the frames into the ones that are used or not could potentially be used.

rushes Summarisation Algorithm
AT & T Labs have created an algorithm that aims to create video summaries of rushes video with minimum redundancy and duration up to 2% of the original video. Prior to summarisation, this approach performs redundancy detection and junk frame removal to remove unimportant information. Following this, sub-shot segmentation is performed to allow easy capture of content changes within a shot. This uses an average salience map (which scores frames according to how interesting they would be to humans) as well as how visually different a frame is compared to a previous frame, and enables the key frames to be chosen from the video. A technique is then applied to the key frames in order to meet the duration requirement stated earlier before finally arranging the frames in a logical order

to produce a simple understandable summary video [20]. This algorithm could potentially work to create the trailer part of our system as it produces a summary with minimum redundancy which would be a vital consideration.

Similarity Algorithm
At Brno University of Technology a system for producing video summaries has been created by identifying similar clips and covering all different flavours of shots. This system performs junk frame removal before clustering similar shots in the video so that shots can be selected using the following criteria: variability inside the shots, length of the shots, and ensuring that at least one representative from each cluster is chosen. By doing this iteratively a sequence of shots for the summary is gathered and then adjusted accordingly (for speed etc) to produce the final summary [13]. This algorithm could potentially work for some genres of video content, however in our system we are aiming to produce general code that could summarise any type of video and in areas such as action or sport there may be too many similar shots to produce an effective summary.

Auto Action Movie Trailer Algorithm
This algorithm uses an approach for automatically selecting shots from action movies to assist in the creation of trailers. Significant shots are identified using colour histograms to catch shot boundaries, and an analysis of the movie audiotrack is performed to detect different aspects such as speech, music and silence, in order to give a percentage of each of these different types of sound to a frame which can indicate important shots in a film. Finally the amount of motion and the percentage of camera movement is also detected for each shot. Utilising these features the shots for the trailer are selected [29]. This algorithm is quite specific to action movies and our system will be looking at all different types of genres although its techniques could potentially be adapted for additional types of video content or it could be an algorithm that is specifically applied when the genre action is selected.

4.2.2 Facial Recognition Algorithms

Facial recognition is a complex goal to achieve and there are a number of factors that make it so. Characters are constantly moving, speaking and performing a range of actions and with the addition of changing camera angles and background scenery it becomes very difficult to keep track of the same face. To combat this face detection algorithms tend to analyse additional information as well as the pure video content in order to maintain recognition of the different characters.

An example of this is a system that analyses extra data associated with the video in order to label each character in each frame of the popular television series “Buffy the Vampire Slayer”. This algorithm works by first analysing and aligning both the scripts and subtitles in order to make predictions about which characters are speaking and when. Once this has been done a face detection algorithm is performed on the video to record the characters clothes and mouth regions. This is followed by a lip motion detection algorithm, which combined with the initial textual analysis enables the system to identify and label the characters within the video [17]. Whilst our system does not have access to the scripts of the video content that will be used, a majority of that will come with subtitle data, which can be analysed to enhance our character detection.

In addition to the constant movement of characters an additional problem in facial detection and recognition is the high level of variation in illumination within a sequence of frames. In video content such as action films or sports matches there will be an even higher level of movement than most genres and particularly in outside sports such as football matches,

there will be the problem of constant changes in the lighting.

One solution to combat these problems is a system which is designed specifically for sporting video content, with a special focus on soccer matches. The system works by detecting faces using a modified adaboost face detector and then keeping track of key facial patches such as the eyes and mouth, in order to more accurately identify the players using a SIFT algorithm by L.Ballan et al. [12]. By focusing on smaller points of the face it increases the likelihood that these points would still be able to be identified in different lighting and in different positions. This is a technique that should be both useful and possible in the team's system as the OpenIMAJ library provides the ability to detect smaller portions of the face, which will enable the system to keep better track of the different characters throughout the scene changes.

4.2.3 Audio Analysis Algorithms

It is important to remember that sequences of frames that make up video content are only half the information. There is also the whole world of audio data that comes with it such as speech, music and additional genre based sounds such as laughter for comedy and crowd response for sporting matches.

Often in action based video content such as action films or sporting matches there is a great deal of background noise and this can be used to aid analysis of the video. An example of this is a system that was designed to generate metadata for TV programs (in particular football games) by analysing the crowd noise to work out the most significant moments. These scenes then had the commentators speech analysed to identify additional information such as which players were involved. This was done using morphological and syntactic analysis with two dictionaries [28]. This technique would enable our system to identify the main scenes in the submitted video content so that they could be analysed to check that they werent giving away the end of the film/episode by identifying their placement within the video and then combined to make a summary trailer.

The other element of audio data that is often useful to analyse is music within a video. This is difficult to achieve because musical pieces are often used as background music behind speech and additional noise within a scene. However, despite these difficulties systems have been written to identify the different musical pieces used in a video. An example of such a system works by continually extracting audio signals from broadcasts and using these signals to construct retrieval keys, which are a series of feature vectors. Once constructed, the keys index into a large music database to identify the different musical pieces [31]. This technique enables the system to be robust against non-stationary noise and would be a useful technique in detecting and identifying music within a video.

4.3 Techniques for producing Audiovisual Summaries

There are three different techniques that are commonly used to produce audiovisual summaries: Internal, External and Hybrid [24]. Each of which are explained in more details in the following points:

- Internal video summarisation techniques produce video summaries by analysing low level features that are only present within the video stream such as colour, shape, object motion, speech or on-screen text.

- External video summarisation techniques produce video summaries by analysing information external to the video stream such as time and location of the video and

user based information such as their description of the content.

- Hybrid video summarisation techniques use a combination of the above to provide additional levels of detail to reduce ambiguity.

4.4 Existing Video Summarisation Solutions

This paper by P. Over et al. [25] describes a comparison and evaluation of thirty-one different research groups approaches to summarising video who had to summarise rush video with the aim to remove redundant and insignificant information and to have a duration of at most 2% of the original. These processes were judged according to measures such as how long it took the system to produce the summary, how long the summary produced was and the quality of the summary. Some of the systems in this paper have been looked at in the section below:

ELVIS - Entertainment-Led Video Summaries
The ELVIS system is an external video summarisation system that produces personalised video summaries based on segments which are chosen according to real time user physiological responses. ELVIS uses five physiological response measures which are electro-dermal response (EDR), heart rate (HR), blood volume pulse (BVP), respiration rate (RR), and respiration amplitude (RA) which are monitored as a user watches the video content. These responses are mapped to scenes within the video then those scenes with the most physiological responses are then chosen to be part of the summary [24].

RISPlayer
The RISPlayer systems is an internal video summarisation system in which summaries are generated and visualised simultaneously where the generation parameters can be modified at the same time. The system uses the binary tree based algorithm detailed in Section 4.2.1 [32].

Home Movie Summarisation System
This system is a home video summarisation system that performs automatic video summarisation before allowing a user to manually edit the summary through a video editing software type interface. The system uses the principle of sub-shot footprints to provide three different summary types, Prominent, Coverage and All. Additional user inputs such as target length and coverage can further refine the summaries produced [16].

Web-based Real Time Content Processing and Monitoring Service for Digital TV Broadcast
The system in question here is a real time content processing and monitoring service for Digital TV Broadcast which provides users with a web interface where they can tune to different TV channels or pre recorded content and view the video/metadata produced by the system. The system works as follows: it receives Digital TV broadcasts and processes this input to collect metadata. Firstly they gather programme information using the PSIP protocol which is part of the TV broadcast and additional information from web sites such as IMDB to obtain more information. Then automatic speech recognition (ASR) is performed using the AT & T Watson speech recogniser to collect ASR transcripts as well as extraction of the subtitles from a programme. The subtitles and ASR transcripts are then normalised in order to extract a list of keywords. Finally using a combination of shot boundary detection, face detection and redundancy removal is performed to select a sample of key shots. The key shots, key words, programme information and the original content are all presented in the web interface [19] .

4.5 Conclusion

After researching the wide selection of techniques the team came to the conclusion about which aspects to include in the system.

The algorithms these solutions use have been compared to those that are available in OpenIMAJ to see which would fit into our system. In regards to the overall video summarisation algorithm, our system will implement an algorithm similar to the Auto Action Movie Trailer Algorithm explained in Section 4.2.1.

The OpenIMAJ library [8] that is used within the system also uses colour histograms to detect shot boundaries so the system will use these methods and build upon them. The aforementioned algorithm also uses sound analysis to indicate important shots within a film, a technique that is also demonstrated in Section 4.2.3. The system will be utilising this technique to pick out the main shots that should be included in the trailer.

5 Design

5.1 Design Overview

The system will collect and infer information about the loaded video(s) in a number of different ways. The generated information will be displayed in multiple ways to the user on a per video basis as well as a summary of all the gathered data.

The number of possible ways data could be gathered is very large, a project of this size will be unable to tackle a great number of these approaches. Depending on the progress of implementing the chosen elements, the team may choose to do further work and add additional features. The project team are new to the discipline of video analysis, and so might not be fully aware of the complexity of certain techniques, or even all of the potential possibilities.

For the reasons highlighted in Section 2.8 the team will aim to make the system modular so that each type of analysis can be executed (and tested) independently. The following diagram illustrates the modular system design:

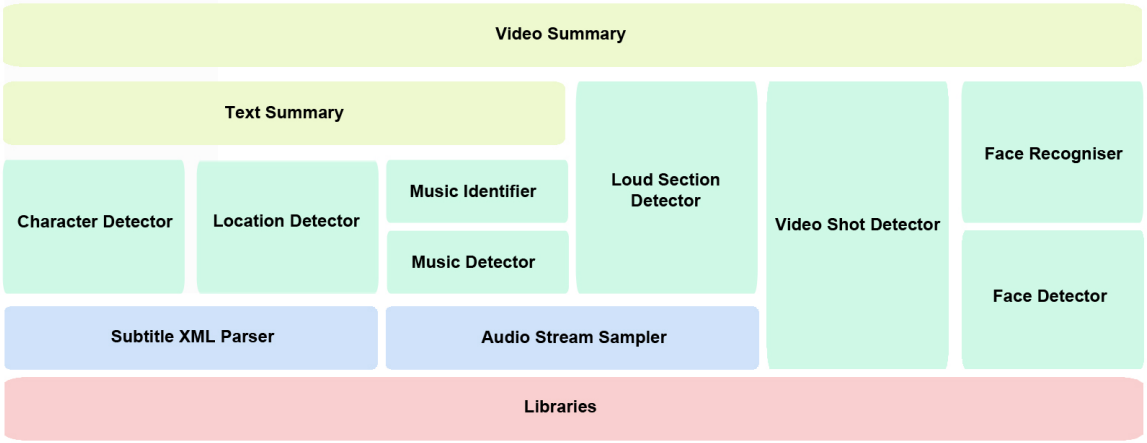

Figure 2: Design Overview Diagram

5.1.1 Text Based Analysis

As detailed in the requirements (see Section 3.2) it is assumed that the system will have access to a DVB subtitle file for each video. This is in XML format so this will need to be processed by an XML parser.

Natural Language Processing (NLP) can follow to assist in finding information which describes the video. This is likely to be faster from a computational perspective. Some interesting information to extract at this stage would be the main characters and the locations.

5.1.2 Audio Analysis

The next resource that the team can take advantage of is the audio track associated with each video. This will involve extracting and segmenting the audio stream, with a focus on the volume of each segment from the video which can be used to produce a list of the interesting timestamps. Assuming that for the most part a loud sequence in a video represents an exciting or interesting activity, the system will mark this as such and make it available for the summary stage.

In addition to this frequency analysis will be implemented for the purpose of discovering music segments in a video, which can then be identified. This is related to the IP Rights Assessment scenario in Section 1.2.

5.1.3 Facial Detection Analysis

In order to identify the main characters of a video or set of videos face detection will be a necessary feature of our system. This will involve detecting the faces in the different frames of the video, clustering them together to differentiate between the different characters, and finally trying to identify the actors playing those characters. The facial detection and recognition module shall be implemented in the system, using the pre-existing OpenIMAJ features.

5.1.4 Video Shot Segmentation

Part of the analysis will involve splitting a video in appropriate places i.e. performing video processing to find shots in the video. This functionality is part of the OpenIMAJ library and so no algorithms need to be produced for this section. References to video shots will be held in memory and this information can then easily be used by other modules.

5.1.5 Textual Summary

The aim of the system will be to produce a textual summary of the interesting features of the videos. In addition the system will have the functionality to navigate to the interesting sections of the source video by using hyperlinks, which was suggested by the customer.The textual summary can contain information from each of the modules and can be combined together where applicable.

5.1.6 Video Summary

The goal of this module is for interesting sequences of video to be joined to produce a video summary of the original video to a target length, whilst still providing a good representation of its content. This will use the information gathered from the various modules to find interesting points in the video then using the video segmentation stage; the interesting shots can then be combined into sequences in order to create a smooth output video.

5.2 Design Patterns

5.2.1 Evaluation of Patterns

The system essentially involves a user submitting a set of video data (either an episode, a series of episodes or a film) and selecting from a set of different options to generate a summary of that data. Due to its complicated nature various design patterns that involve simplifying the structure and the use of algorithms have been evaluated for their usefulness.

Model View Controller (MVC)
Description: The MVC pattern involves splitting up the code so that the model deals with the data within the program, the View renders the display based on the data held in the Model and allows the user to interact with it whilst the Controller performs the actions specified by the user. The Controller has access to both the Model and the View and the View has access to the Model.
Evaluation: This pattern would be useful in structuring the code for this project as it contains a user interface that runs varying summarisation algorithms on the data submitted by the user so it fits perfectly into the pattern.

Strategy Pattern
Description: The strategy pattern involves defining a set of separate interchangeable algorithms that can be selected at runtime, independent of the user.
Evaluation: This pattern would be very useful for the project as a number of different algorithms will be necessary to create the different summaries of the video content. The algorithms can be defined in different classes and can be selected at runtime depending on which summary options the user selects.

Template Pattern
Description: The template pattern involves outlining the skeleton of the main algorithm of the program with some of the steps being defined in subclasses so that the algorithm can run in different ways without changing its structure.
Evaluation: This pattern could be appropriate for this project in the sense that every operation performed on the data will be a summarisation algorithm. However, given the range of different summaries that this project is looking to perform (main characters, main action sequence, potentially even music rights) it would be difficult and perhaps restricting to incorporate everything into one algorithm.

Composite Pattern
Description: The composite pattern aims to ignore the difference between compositions of objects and individual objects.
Evaluation: This pattern could be put to good use in this project, as generic algorithms could be defined to do either one scene of an episode, a whole episode or even a collection of episodes.

Proxy Pattern
Description: The proxy pattern provides an identical interface to the subject interface so that in certain instances a proxy can be substituted in place of the real subject.
Evaluation: This pattern could potentially be utilised in this project by creating a proxy for the video so that only the relevant parts could be loaded into the system with the rest being replaced by the proxy as suitable.

5.2.2 Proposed Solution

The proposed solution is to combine the two most useful patterns, MVC and Strategy. The structure of MVC will be used to split up the code and the strategy pattern will be implemented in the Controller to separate the different summarisation algorithms needed to perform the operations specified in the View. This will keep with the theme of only showing the user the necessary components (the View) and will hide the Controller's actions (the different algorithms) from the user, as they should be independent from them. Other elements of the aforementioned patterns could also be added in as necessary, such as defining the strategy algorithms so that they could be ran on a composition of objects or an individual object (from the composite pattern). The functionality described by the proxy pattern could also be used when appropriate to provide a substituting interface dependent on the data entered into the system [18].

5.2.3 Architecture Structure Diagram

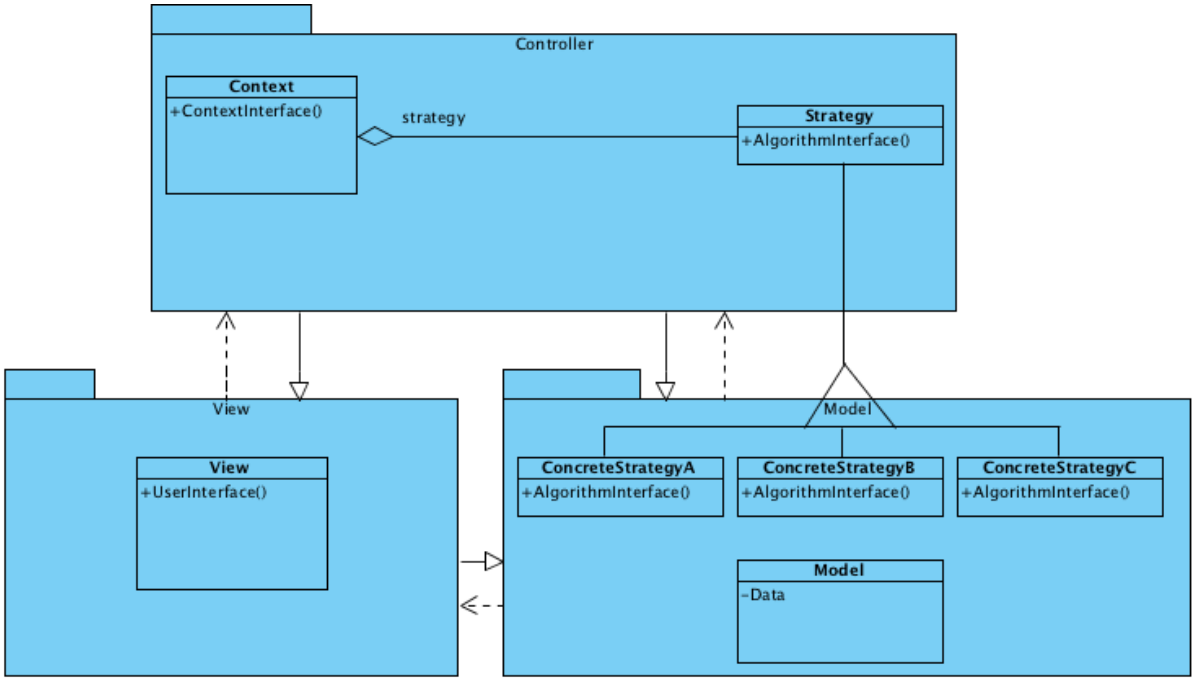

Figure 3: Design Overview Diagram

5.3 User Interface Design

The design of the team's user interface for the system is below:

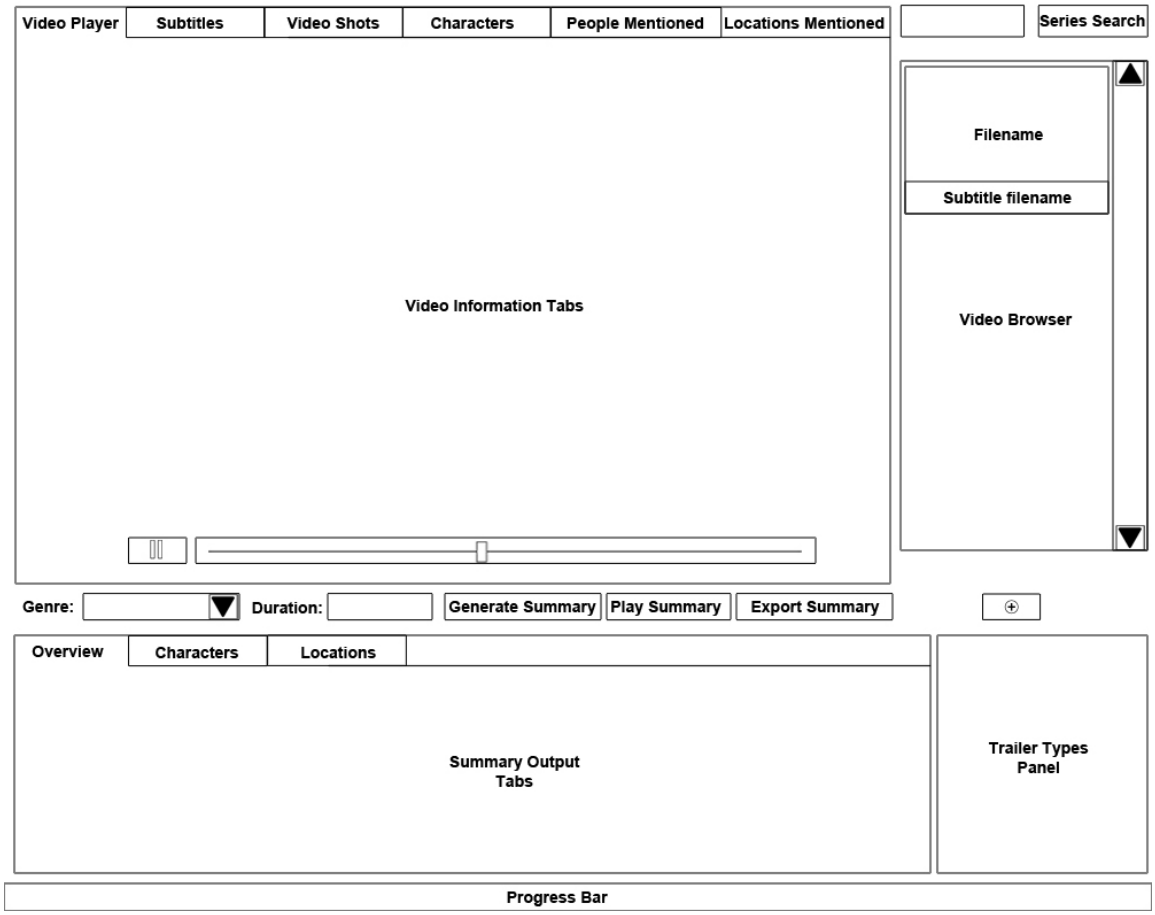

Figure 4: User Interface Diagram

This has been designed in to enable users who have had previous experience with video editing programs to quickly get to grips with how to use the system effectively. Furthermore, by having a user interface it hides the complexity of the background processing, which also allows the video summarisation and the results it produces to be easily demonstrated to the team's supervisor, customer and anyone else who attends the presentations that are a part of the assesment criteria.

5.4 Prototype

The prototyping stage had the main aim of getting the team to set up their development environments and ensure that they could work with the various tools and technologies that had been chosen in Section 2.4. For example, Maven [6] is used to help with managing the building and deployment processes. Once Maven was set up on each development system and the project had been checked out from UGForge using SVN, it was straightforward to progress as a team.

The first stage was to construct the basic Graphical User Interface (GUI) (the design of which is shown in Section 5.3), followed by building the MVC packages and finally, producing skeleton classes for all backend modules as shown in Section 5.1 in order to allow the structuring of the application correctly. Next, the ability to load video and subtitle files into the application was implemented.

The modules were divided up amongst the individual team members in order that each individual could focus on a particular element or module of summarisation and were partially integrated into the GUI. The semi-complete features at the prototyping stage were shot detection, face detection and subtitle parsing, all of which produced output to the GUI at intermediate stages. This division allowed further understanding of the software libraries and also meant that a good demonstration of what was going on behind the scenes was available to show at Progress Seminar one.

The prototype provided a good foundation on which to start the implementation stage but also during the process, the team learnt about the OpenIMAJ software library and discovered a number of bugs particularly concerning how efficiently the software would process the video. The team was in frequent contact with the developers of the software and were able to pass on any issues and receive feedback very quickly.

6 Implementation

The system involves a series of core modules such as video shot detection, audio processing and subtitle processing, which are shown in Figure 2.

The following section outlines the implementation of each of these core modules including class diagrams to demonstrate the structure as well as high level flow diagrams to show how the different components are chained together.

These modules typically use the Objects defined in the package:
uk.ecs.gdp.avsumamriser.model.section; the structure of which is shown in the class diagram below:

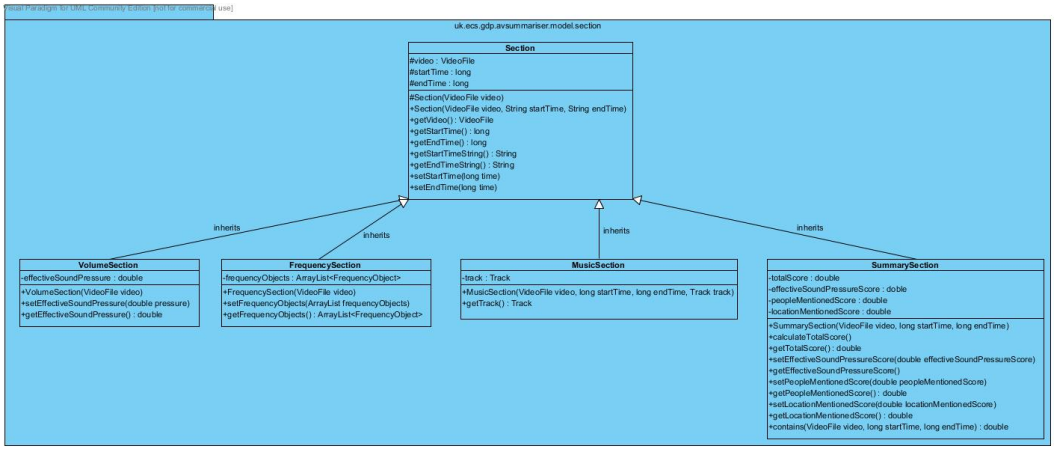

Figure 5: The Section Package Diagram

A Section object (the parent class) contains a VideoFile object and two longs values which are start and end times in milliseconds. These objects are used to define when something occurred in a video and the data that belongs to it, the additional information that the different objects contain are:

- VolumeSection - a double value for effective sound pressure (ESP) of this section.
- FrequencySection - an ArrayList of FrequencyObjects.
- MusicSection - a Track object.
- SummarySection - four double values which represent ESP score, People mentioned score, Locations mentioned score and a total score.

6.1 Software Libraries Used

Before describing each of the modules, the software libraries that have been used within this system and for what purposes they were used for are presented in the table below.

Software library / Resource name	Source	Usage
OpenIMAJ	[8]	All modules
JDOM	[2]	XML parsing of subtitle files & XML export of summary data
OpenNLP	[7]	Natural language processing of subtitle files to find person's names and location's names
Online TV Database	[9]	Open source database containing information about television programmes
javatvdbapi	[10]	API to communicate with thetvdb.com
Echonest Music Intelligence Platform	[1]	Open system that provides a music identification service.
jen-api	[3]	API to communicate with Echonest platform.
json-simple	[5]	Dependency for jen-api
jfreechart	[4]	Produces graph of data for each video.

6.2 Online Television Database Integration

Package name: uk.ecs.gdp.avsummariser.model.tvdb.

6.2.1 Class Diagram

The class diagram below details the structure of the classes within the package.

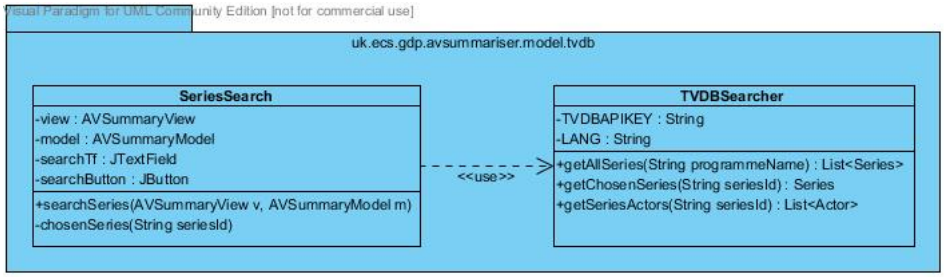

Figure 6: The tvdb Package Class Diagram

The purpose of this package is to communicate with an open source database containing information about television programmes 6.2 using the library javatvdbapi [10]. The classes shown in figure 6 are explained further in the following sections:

6.2.2 TVDBSearcher

A static class used to communicate with the database and to gather all the necessary information such as:

- Obtain a list of Series (describes the television series such as its air date, a text overview and genres) objects given a search String.
- Obtain a Series object for the chosen television series.
- Obtain a list of Actor objects which represents the main characters for that series (each Actor object contains an actor name, character name and an image of the character).

6.2.3 SeriesSearch

A class called from the View and deals with the user's interaction, for example it presents a list of search results as a pop up message, sets the Model and updates the View after a user has chosen the television series. It uses the TVDBSearcher class to collect data from the database.

The information which this module gathers is used in Face Recognition (see Section 6.5) and Name detection in subtitle files (see Section 6.3.9).

6.2.4 User Walkthrough

Below are two screenshots indicating how to link to the TV Database:

Step 1: Add video, type in the series name, press return and select available series.

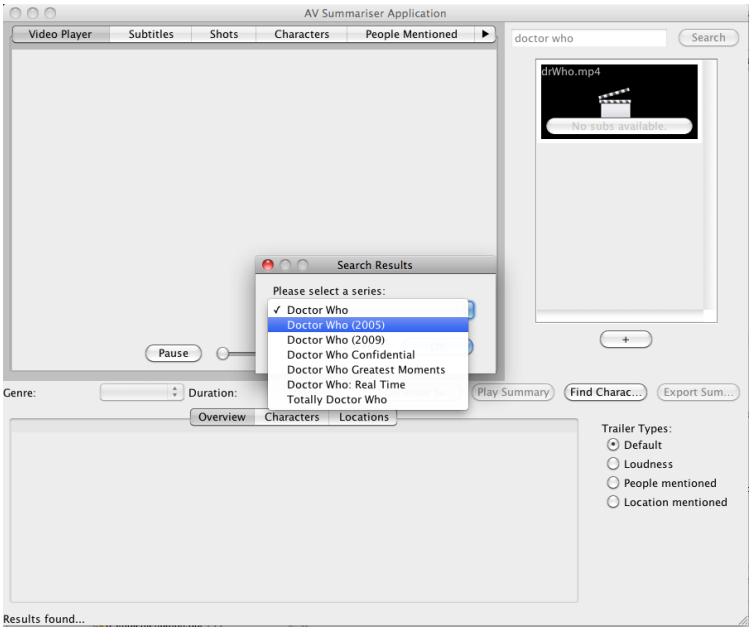

Figure 7: Selecting a Series

Step 2: Press Generate Summary and the information is loaded into the Overview Tab.

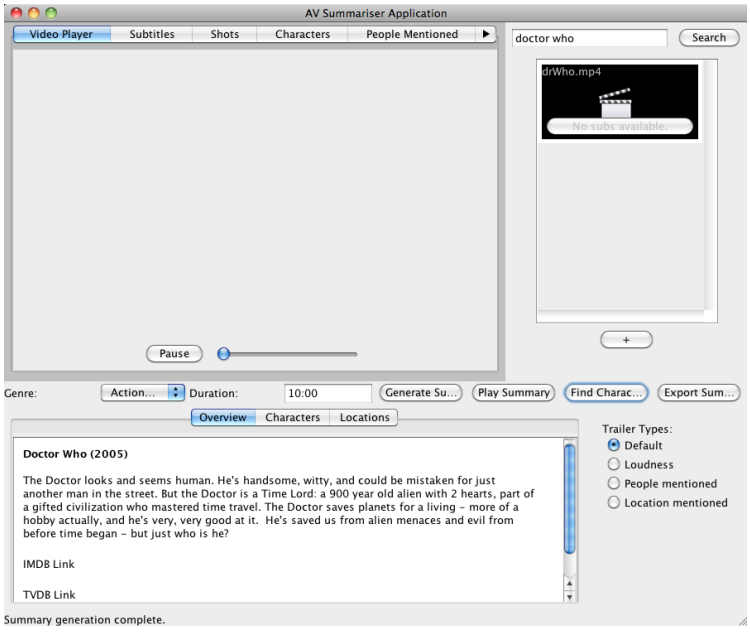

Figure 8: Viewing the Series Information

6.3 Subtitles

Package name: uk.ecs.gdp.avsummariser.model.subtitles

The purpose of this package is to contain all of the Java code which interacts with the subtitle files that are associated with VideoFile objects. It performs:

- XML parsing of the subtitle files into the Subtitle Objects.
- Natural language processing to detect the main person's and location's names. This may include using data produced as a part of the Online Television Database Integration 6.2

6.3.1 Process Diagram

A diagram of how these processes are chained together is given below:

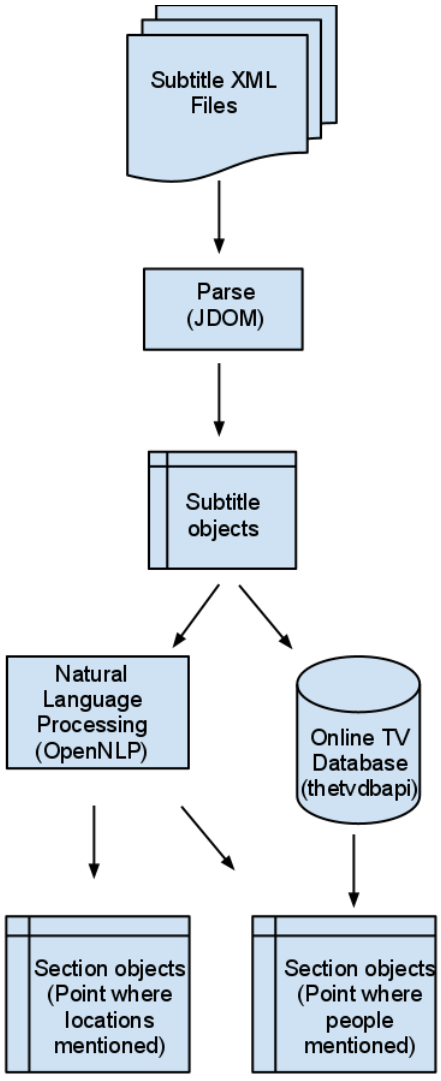

Figure 9: The Subtitle Parsing Process Diagram

6.3.2 Class Diagram

The structure of the classes used within this package are shown in the class diagram below with further detailed information following.

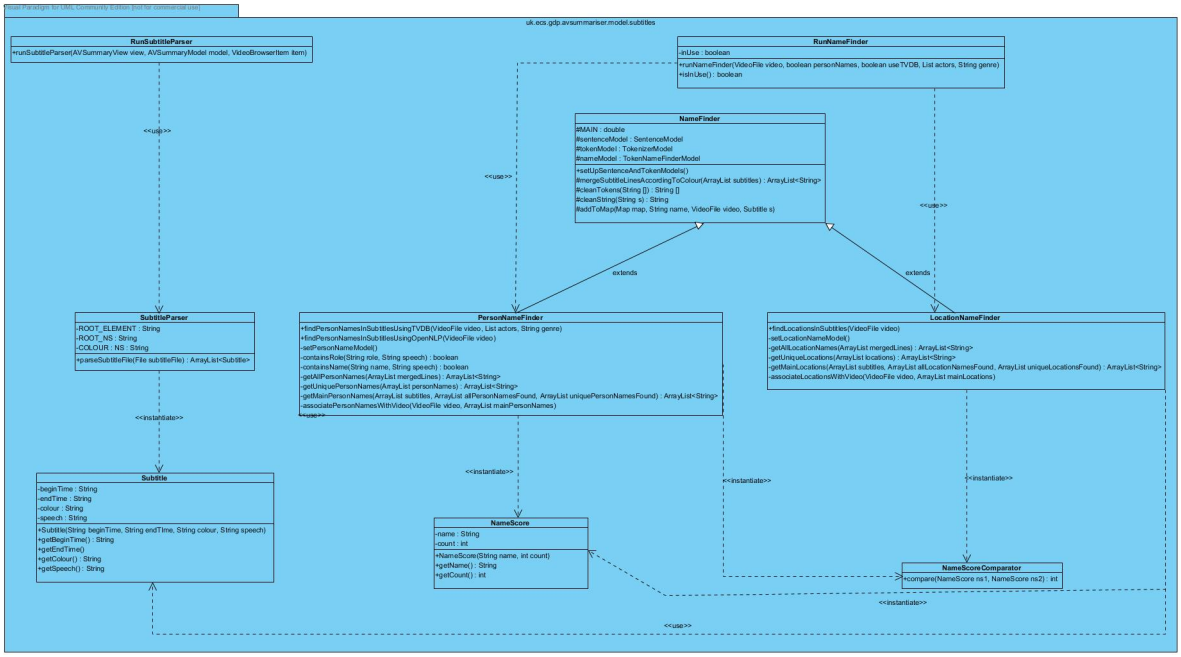

Figure 10: The Subtitles Package Class Diagram

6.3.3 Subtitle

This class represents an individual subtitle in a DVB subtitles file which contains a:

- Start time String.
- End time String.
- Speech String.
- Colour String which refers to a unique speaker in a video.

6.3.4 SubtitleParser

This class parses an XML subtitle file (if it is of the correct format i.e. DVB format) to produce an ArrayList of Subtitle objects, which is done using the JDOM library [2].

6.3.5 RunSubtitleParser

This class is used to interact with the user when they select to load a subtitle file i.e. displaying a filechooser, updating the View and Model. This class then calls the SubtitleParser class to do the XML parsing.

6.3.6 NameScore

This class is used for name ranking purposes within the name finder classes below. It contains a String which is a name (e.g. person or location) that was found in the Subtitle Objects and a Integer which is a count of the number of times that name occurred in different Subtitle Objects.

6.3.7 NameScoreComparator

This is a class which implements the Comparator interface and is used to sort an ArrayList of NameScore Objects in order of decreasing count value.

6.3.8 NameFinder

This class is the parent class for the PersonNameFinder and LocationNameFinder classes, it contains variables and methods that are common to both classes.

6.3.9 PersonNameFinder

A static class used to detect the main person names mentioned in an ArrayList of Subtitle Objects. This can be done in either two ways; using natural language processing using the OpenNLP library [7] or using information from the Online Television database Section 6.2:

The natural language processing method works as follows:

Stage 1: Firstly, the speech variables in each Subtitle Object are merged together according to the String colour where they occur consecutively. This produces an ArrayList of String Objects.

Stage 2: Using the ArrayList produced previously, an ArrayList of String Objects containing all person names found is constructed by:

- Splitting each String Object into sentences.
- Splitting sentences into tokens.
- Tokens are then cleaned to remove punctuation and non speech items like ALARM SOUNDS.
- Finally, all person names are located.

Stage 3: This ArrayList contains all person names so it is then analysed to produce an ArrayList containing all unique person names where full names take priority i.e. if this system detects a name, Ian, and another name, Ian Hislop, then in the unique names list it will only contain, Ian Hislop, and not, Ian.

Stage 4: For each unique name in this list the system counts how many times that name appears in the Subtitle Objects and creates a NameScore Object. These NameScore Objects are stored in another ArrayList. Next the ArrayList of NameScore Objects is sorted in descending order of count then the top 5% of person names are chosen as the main people.

Stage 5: Finally, using the main people's names and an ArrayList of Subtitle Objects, Section Objects are created for each occurrence of each main person name in the Subtitle Objects which is added to a Map object (where each String which is a main person name maps to a ArrayList of Section objects). Once this process has completed, the Map Object is associated with that video.

Alternatively to natural language processing, the system can use the information provided by the TVDB database which works as follows:

Stage 1: For each Subtitle Object it loops through the television series's Actor Objects and depending on the genre selected in the View either the character name or actor name Strings are used by the system and occurrences of these in each Subtitle Object are detected. Section Objects are created for each occurrence and adds it to a Map Object. Once this process has completed, the Map Object is associated with that video.

6.3.10 LocationNameFinder

A static class used to detect the main location names mentioned in an ArrayList of Subtitle Objects. Which is done using natural language processing using OpenNLP library [7] and works as follows:

Stage 1: Firstly, the speech variables in each Subtitle Object are merged together according to the String colour where they occur consecutively. This produces an ArrayList of String objects.

Stage 2: Using the ArrayList produced previously, an ArrayList of String Objects containing all location names found is constructed by:

- Splitting each String Object into sentences.
- Splitting sentences into tokens.
- Tokens are then cleaned to remove punctuation and non speech items like ALARM SOUNDS. Finally, all location names are found.

Stage 3: The ArrayList contains all the location names so it is then analysed to produce an ArrayList containing all unique location names.

Stage 4: For each unique location in the list the system counts how many times that location appears in the Subtitle objects and creates a NameScore Object. These NameScore Objects are stored in another ArrayList. Next the ArrayList of NameScore Objects is sorted in descending order of count, then the top 5% of location names are chosen as the main locations.

Stage 5: Finally, using the main location names and an ArrayList of Subtitle Objects the Section Objects are created for each occurrence of each main location name in the Subtitle Objects which is added to a Map Object (where each String which is a main location name maps to an ArrayList of Section Objects). Once this process has completed, the Map Object is associated with that video.

6.3.11 RunNameFinder

A class constructed to overcome the problems of the non thread safe TokenNameFinder-Model and NameFinderME objects in OpenNLP [7], which throws a NullPointerException when different name finder threads are running i.e. one performing PersonNameFinder and another performing LocationNameFinder. Therefore this class was created to only allow one NameFinder process to run at a time.

6.3.12 User Walkthrough

Below are two screenshots indicating how to load in the subtitles:

Step 1: Add video, select the subtitles section and add the appropriate subtitles.

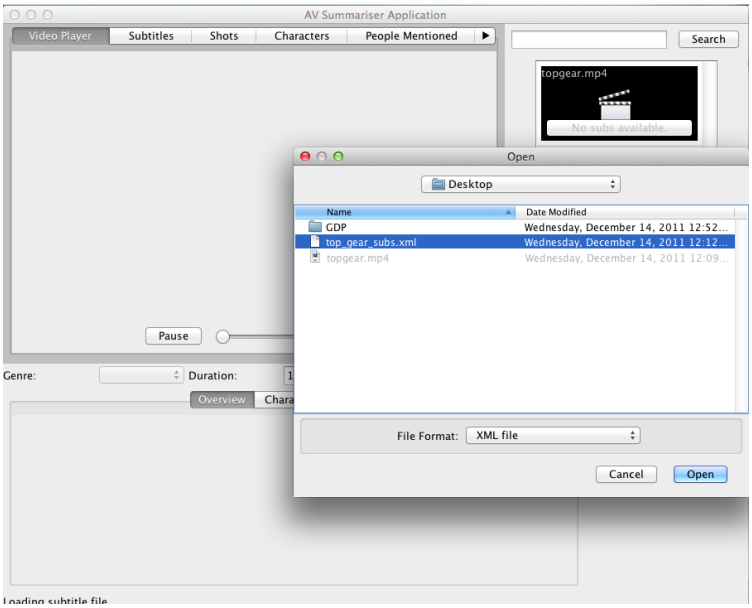

Figure 11: Adding Subtitles to a Video

Step 2: Select the video and the subtitles are loaded into the the Subtitles tab.

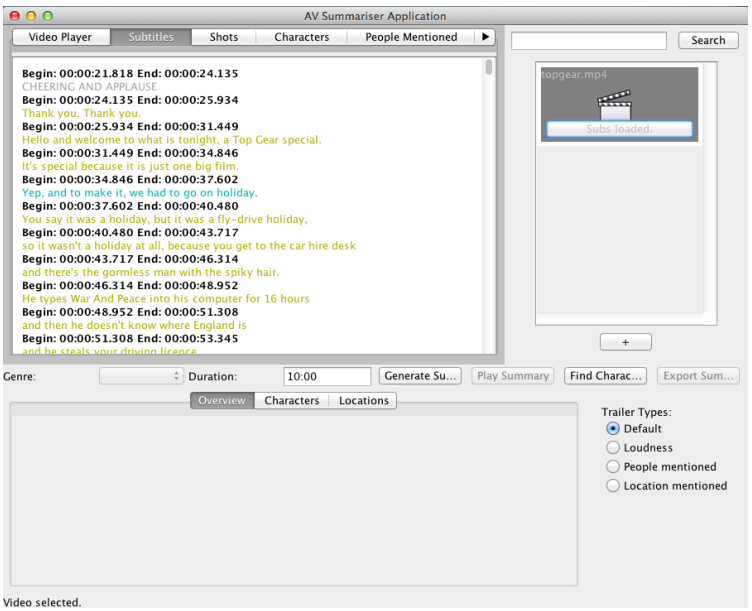

Figure 12: Viewing the Subtitles

6.4 Audio Processing

Packages:

- **uk.ecs.gdp.avsummariser.model.audio**
- **uk.ecs.gdp.avsummariser.model.music**

The purpose of this package is to contain all the Java code which processes the audio of video files. This has been implemented using the OpenIMAJ library [8] and for an audio stream it:

- Determines the loud sections and produces an ArrayList of VolumeSection Objects.
- Determine the frequency and intensity of an audio stream and produces a list of FrequencySection Objects.
- Identify music sections.

6.4.1 Process Diagram

A diagram of how this process chains together is given below:

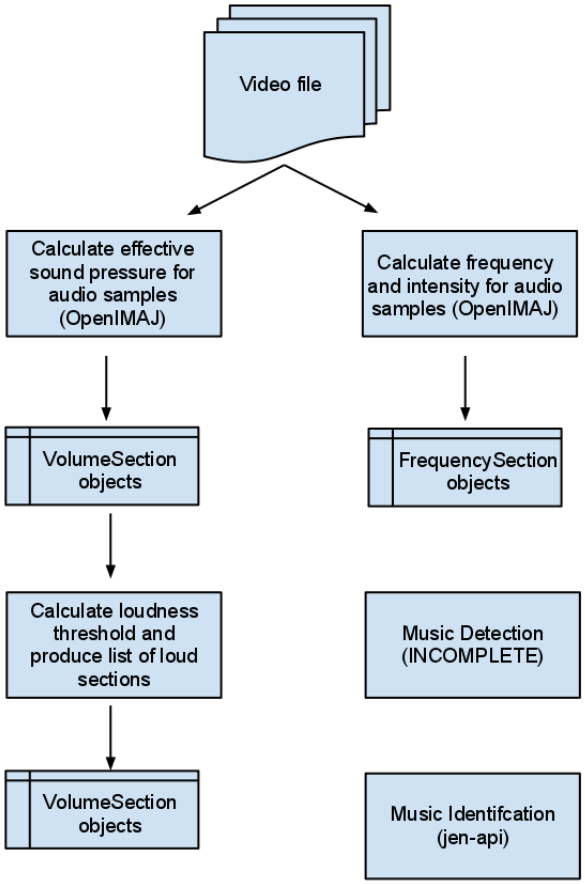

Figure 13: The Audio Parsing Process Diagram

6.4.2 Class Diagram

The structure of the classes used within this package are shown in the following class diagrm:

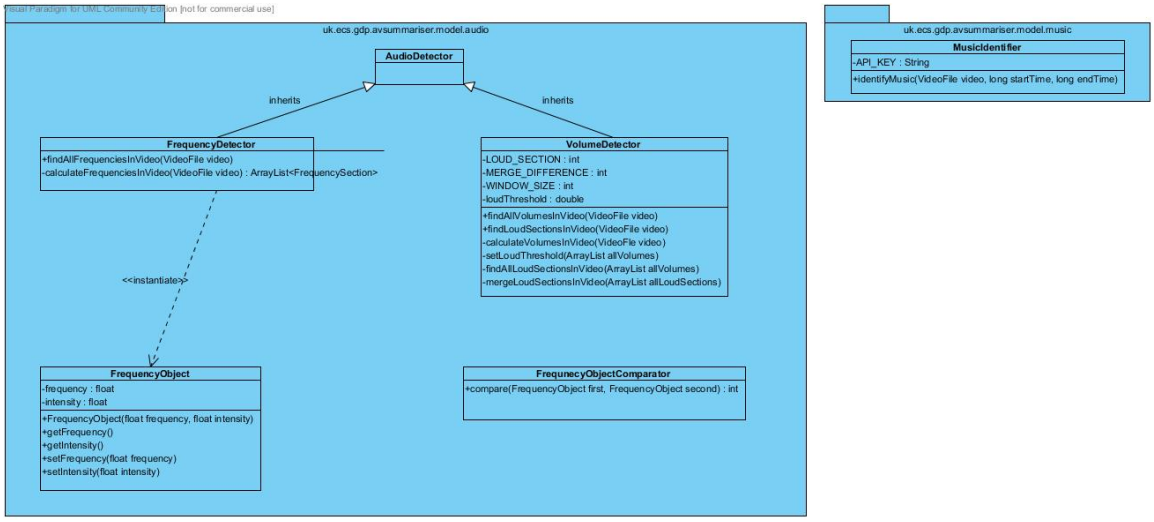

Figure 14: The Audio Processing Package Class Diagram

6.4.3 FrequencyObject

A class used to store the frequency and intensity float values for each Fast Fourier Transform value calculated from an audio sample. An ArrayList of these Objects are stored within each FrequencySection Object.

6.4.4 FrequencyObjectComparator

A class which implements the Comparator interface and is used to sort an ArrayList of FrequencyObject Objects in an ascending order of frequency value.

6.4.5 VolumeDetector

A static class used to calculate the effect sound pressure (ESP) of the audiosample and determine the loud sections, this is done in the following way:

Stage 1: For an audio stream, the system samples it into chunks and calculates the ESP for that chunk and creates a VolumeSection Object. This produces an ArrayList of VolumeSection Objects for the entire video.

Stage 2: Next using this ArrayList, it sorts the VolumeSection Objects in descending order of the ESP values and then selects the index of the last VolumeSection Object in the top 1%, the value of this Object will be the loud threshold.

Stage 3: Now the VolumeSection Objects are looped through in time order and any Object with ESP value greater than or equal to the loud threshold is kept, so now there is a ArrayList of loud sections.

Stage 4: Finally, these loud VolumeSection Objects are then merged if they are within a second of one another to produce the finalised ArrayList of VolumeSection Objects, which are then associated with a video.

6.4.6 FrequencyDetector

A static class used to calculate the frequency of audio samples within an audio stream of a video file. The calculation of the frequency of audio samples is done in the following way:

Stage 1: For an audio stream, the system samples it as chunks and performs the Fast Fourier Transform (FFT) for each chunk. For each value produced by the FFT it works out its frequency and intensity and creates a FrequencyObject Object. Then a FrequencySection is created for that chunk and it contains an ArrayList of FrequencyObject Objects. Once this process is completed, an ArrayList of FrequencySection Objects is associated with that video.

The second purpose of the class was to then use the data calculated above to detect if and where music occurred in an audio stream of a video. However this section of the detector is incomplete for a number of reasons firstly is that the team was unable to find any suitable libraries to perform this process. The set of tools called jMIR [22] which was found to include JAudio [21], however the documentation was hard to understand and no tutorials were found therefore it was hard to judge whether it was fit for purpose and so no use to the system. Therefore, as per the risk strategy defined in Section 8.3.1, the team opted to try and implement our own solution. Firstly a process of detecting frequency patterns was constructed to detect music although this became apparent that it wasnt generalized and only worked for the example that it was being tested on. The next attempt was using beat detection, prior to OpenIMAJs addition of BeatDetector, the Frequency selected sound energy algorithm detailed in this paper [26] was chosen to be implemented due to FFT being implemented in OpenIMAJ already. However after testing this method it was also unsuccessful.

Due to the time restrictions of the project and the lack of successful results, this part of the class is incomplete and therefore has formed part of our future work, shown in Section 10. Furthermore, the frequency detection has not been integrated into the final product although the code exists within the system.

6.4.7 Music Identifier

A static class used to communicate with the EchoNest service [1] to identify music using the jen-api [3]. This works in the following way:

Stage 1: Given a VideoFile Object and a start and end time in milliseconds, it creates a new temporary video file for that specified segment.

Stage 2: This file is then uploaded to the EchoNest service for analysis then one of the following can occur:

- If the analysis completes and is successful in identifying, the track object which contains information such as the artist's name, song's title and the album's name is returned.
- If the analysis completed and is unsuccessful in identifying so a Track object is returned, but with all information set to null.
- Finally, if the analysis fails then nothing is returned.

Stage 3: Then using either a Track Object or null, a MusicSection Object is created containing the video used, the times specified as well as the Track or null Object.

6.5 Facial Recognition

Package: uk.ecs.gdpavsummariser.model.facialdetection

6.5.1 Face Detection Techniques

Before describing the different classes in this section, the systematic set of techniques used to produce these final classes and the findings made throughout the duration of the project are detailed.

Simple Face Detection
Initially the face recognition section purely dealt with identifying faces, it used a face detector from the OpenIMAJ library [8] to run through the frames of the video and pick out and draw the faces that were found.

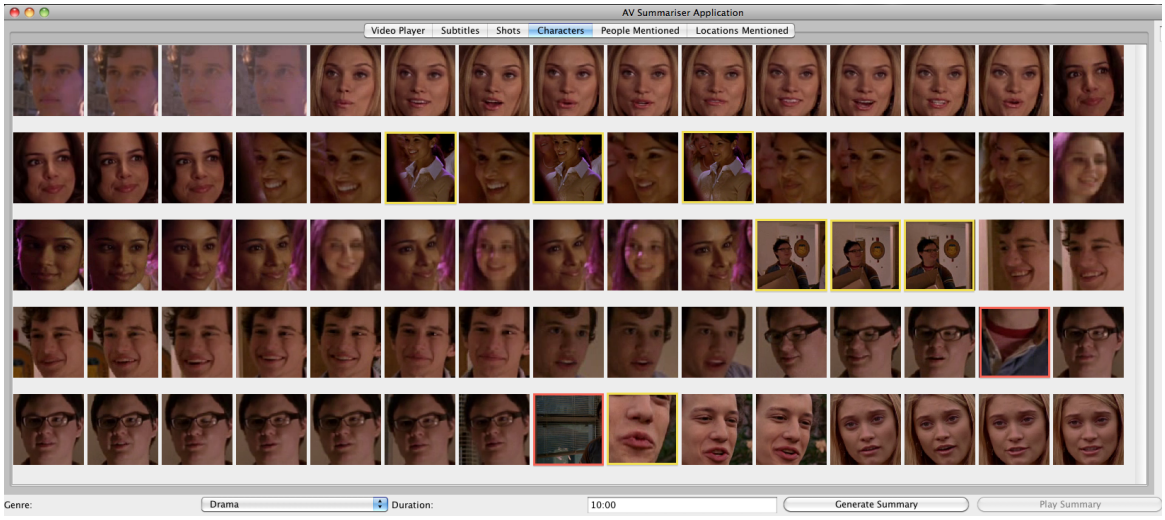

Figure 15: Greek Facial Detection

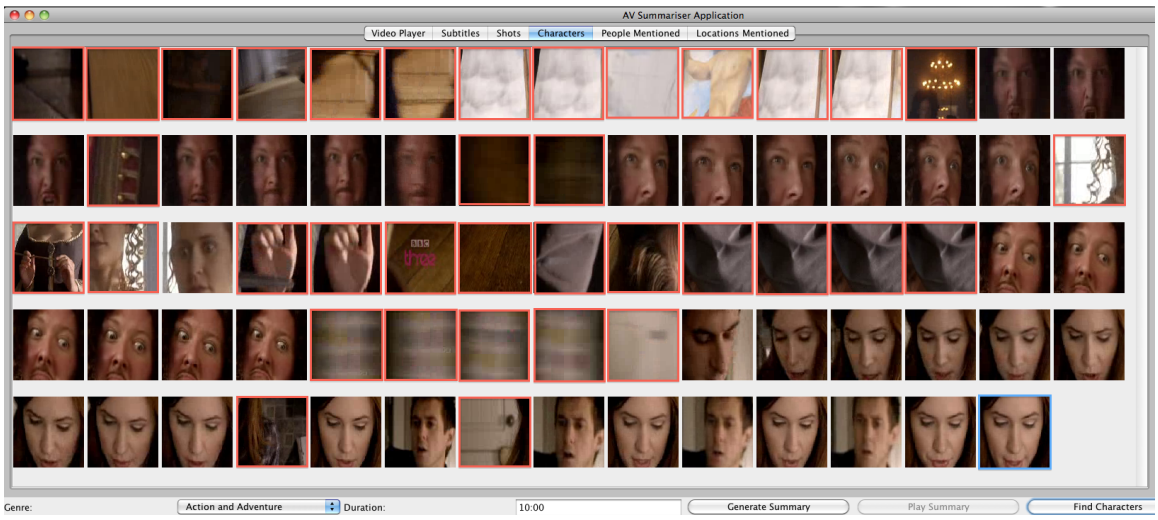

Figure 16: Doctor Who Facial Detection

As you can see from the images above in each instance there are still some false positives coming through (although significantly less in Figure 17). Following are the statistics of the percentage of correctly extracted faces from the previous images.

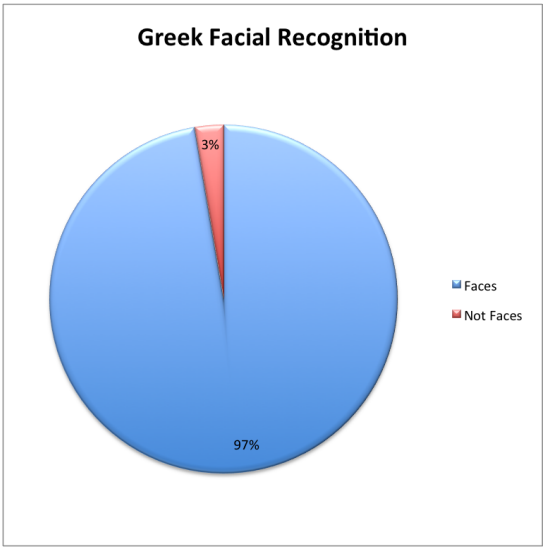

Figure 17: Greek Pie Chart

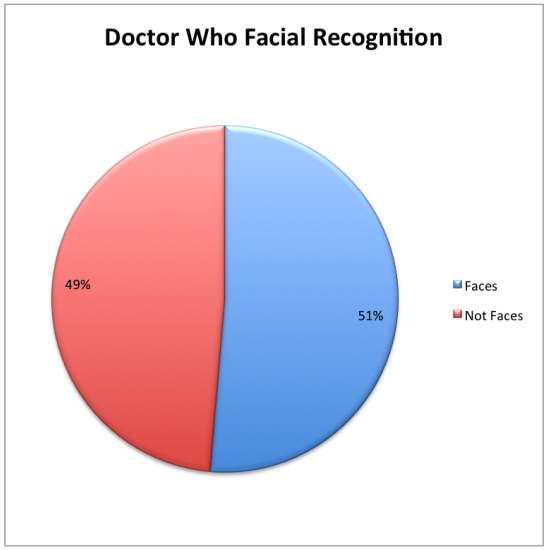

Figure 18: Doctor Who Pie Chart

As can be seen from the pictures above, this produced varied results based on the genre of the video. A light comedy video as displayed in Figure 17 produced significantly better results than a poorly lit, action filled video as displayed in Figure 31.

Advanced Face Detection

In order to achieve advanced face detection a minute of footage was gone through looking at every fifth frame and each detected face that was collected within these frames was logged as either a specific character match or a non-character. It is well known that facial detection is not an exact science and that when faces aren't fully frontal facing in good lighting conditions its extremely difficult to identify them accurately. Therefore its unsurprising that whilst the OpenIMAJ library detects a majority of the faces that appear in the video occasionally other objects get detected as faces also. Therefore these methods were ran to attempt to determine the following:

- How to tell if a detected face was actually a face.
- How to tell if two faces were actually the same person.

OpenIMAJ provides methods to create histograms of images and to compare these for similarity. These methods can be ran with various different parameters such as different types of detected faces, and different aligners to try and match the images. To gather statistics these methods were ran with the aligner that were deemed most appropriate and some interesting results were produced. Whilst looking at the different values produced for the different characters it didn't give any real indication of scale, looking at the correlation between faces and false positives gave some more useable results:

	Min Value	Max Value
True Faces	7.23	11.62
False Faces	7.61	17.29

Table 4: Correlations between faces and false positives

These results show that all the actual faces only come within a certain range of values, despite the false positives also producing values within that range, they also produced values that are significantly outside it. Using these results, any detected face that came outside the boundary that was likely to be a face was discarded. The face detection was then ran with various boundary values based off these statistics. Below are some pictures illustrating that. In each picture the incorrectly identified faces are outlined in red, and a common face to all images is outlined in blue to show the difference in facial extraction per the same number of frames.

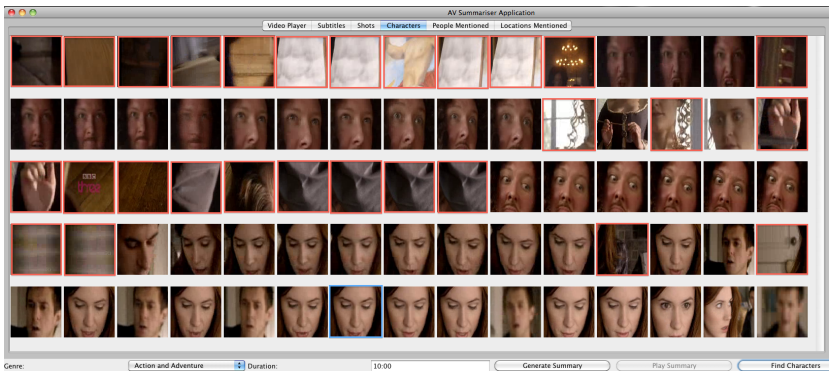

Figure 19: Facial Detection on Doctor Who: min value = 7.5 & max value = 11.5

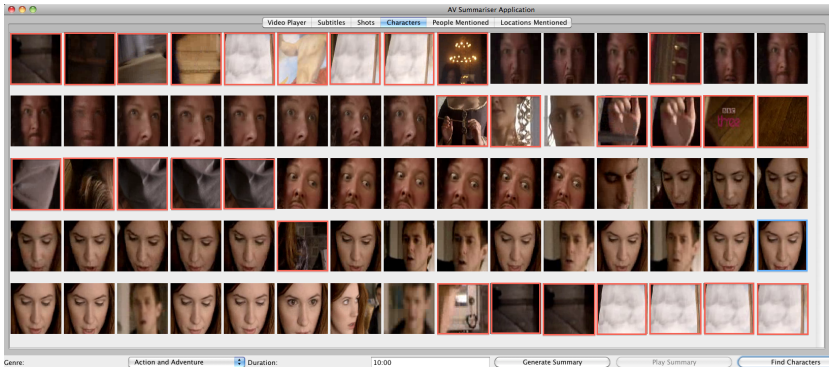

Figure 20: Facial Detection on Doctor Who: min value = 7 or 8 or 9 & max value = 11

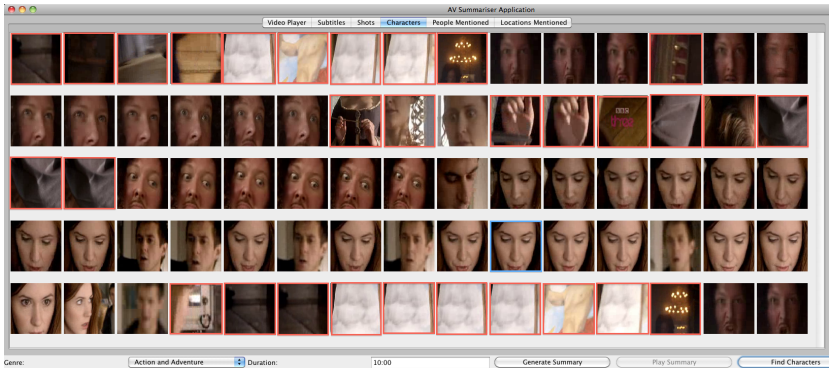

Figure 21: Facial Detection on Doctor Who: min value = 10 & max value = 11

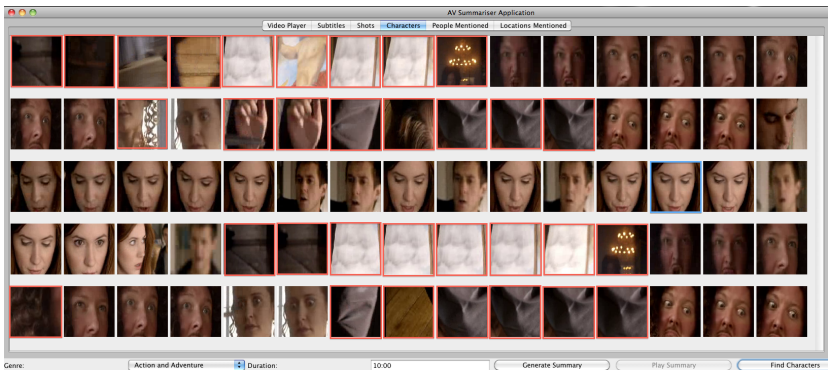

Figure 22: Facial Detection on Doctor Who: min value = 10.2 & max value = 11

These pictures show that by adjusting these ranges different images will be registered as faces. However, it is difficult to tell from these images which method is the most accurate in terms of getting the highest percentage of positive faces. It is also misleading to solely judge this by the percentage of matches displayed in the panel as two ranges could produce the same numbers but different sets of images. Below is a table that shows the ranges against the percentage of faces in the panel as a whole and the percentage of faces in a set of frames common to all experiments.

Histogram Range	No Faces that fill the panel	% Faces that fill the panel	No Faces in common frames	% Faces in common frames
None	38/74	51.3%	38/74	51.3%
7.5-11.5	46/75	61.3%	38/67	56.7%
7-11	46/75	61.3%	38/60	63.3%
8-11	46/75	61.3%	38/60	63.3%
9-11	46/75	61.3%	38/60	63.3%
10-11	45/75	60.0%	35/55	63.6%
10.2-11	43/75	57.3%	26/43	60.5%

Table 5: Ranges and Percentages of Faces

These results show that looking purely at the percentage of faces that fill the panel it would be most efficient in terms of getting the highest percentage of matches to use the range 7.5-11.5 or 7/8/9-11. However, the percentage of matches in the common frames suggests that the most accurate range to use is 10-11, supported by the fact that it had the same number of face matches as the wider ranges of 7/8/9-11, but it had 8 less false positives. Therefore that range is the one that is used in the final face detection methods.

An additional technique that was added to this for the darker programmes was a pixel comparison method. The middle square of each face was compared to get its mean value and statistics were run in the same way as above and the results were as follows:

	X Min Value	X Max Value	Y Min Value	Y Max Value
True Faces	0.000000059	0.89	0.000000029	0.85
False Faces	0.0	0.98	0.0	0.98

Table 6: X and Y Values of True and False Faces

Following the same pattern as above two different range of x,y values were tried and below are the pictures to illustrate this experiment.

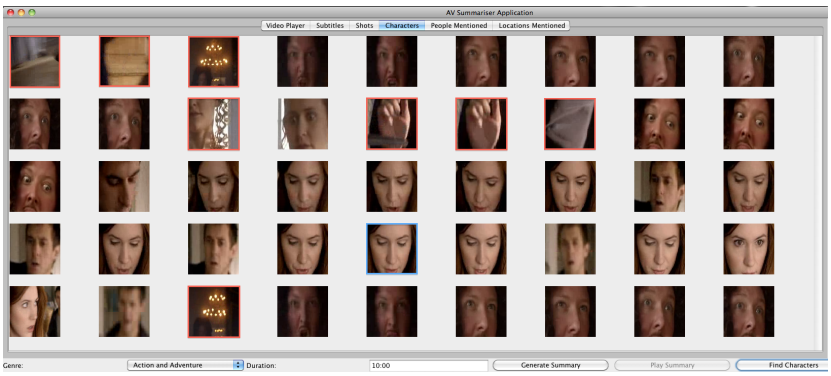

Figure 23: Facial Detection on Doctor Who: min x,y value = 0.2 & max x,y value = 0.8

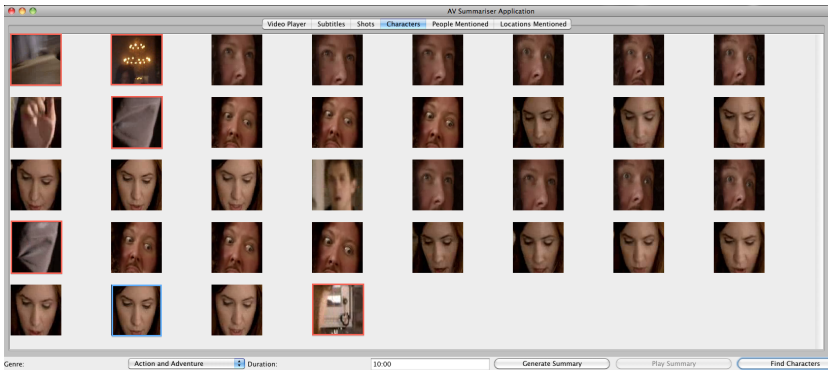

Figure 24: Facial Detection on Doctor Who: min x,y value = 0.3 & max x,y value = 0.6

As you can see from the pictures above these are primarily now made up of faces with a lot of the false positives irradicated. However, whilst this stricter method makes for a set of almost all faces, it still throws away some positive matches. Below is a table of stats to reflect these images.

X,Y Values	No Faces that fill the panel	% Faces that fill the panel	No Faces in common frames	% Faces in common frames
0.3,0.6	29/36	80.5%	28/37	75.6%
0.2,0.8	37/45	80.0%	25/32	78.1%

Table 7: Face Data for previous Figures

As you can see above, when comparing the common set of frames the wider range proves more accurate and still keeps a majority of the faces detected in the methods described above. It is debatable as to whether it is a worthwhile trade to throw away some positive matches in exchance for reducing the amount of false positives. The reasoning here has been that because facial clustering is such a hard problem, to maximise the chances of accuracy it would help to work from a set of accurately judged faces. In addition to this the data this section aims to produce for the summary is a log of a character's appearances and a set of timestamps where it is very likely that a character is in that frame. Therefore this method was implemented as an additional step for the darker programmes under the genres of "Action & Adventure" and "Science-fiction".

Face Clustering
After refining the facial detection techniques, the next stage was to attempt to cluster them. Four different techniques were tried to cluster the faces successfully:

- SIFT Comparison of Facial Keypoints
- Histogram Comparison using an aligner to account for faces in different positions
- Pixel Comparison of the center and boundary areas of the detected faces
- SIFT Comparison of Image Keypoints

All of these techniques were tried on the lighter comedy video to see how well the clustering would work on an easier video first before looking into darker videos.

SIFT Comparison of Facial Keypoints: Using this method the different facial keypoints (e.g mouth and nose) were compared using a SIFT Comparator, similar to the method described in L. Ballan et al. paper [12]. On paper this seemed like the obvious solution but in practice it was very difficult to construct a scale from the returned values to ascertain what made a match. Therefore this technique was quickly abandoned.

Histogram Comparison using an aligner to account for faces in different positions: This technique uses the same principles as the one mentioned above in regards to advanced face detection. However, instead of working on a range, this technique tried to pinpoint how close the image match results were to the middle of the given scale, reasoning that this might be the way to identify matches within the different faces. A screenshot showing the data gathered from this attempt is shown below:

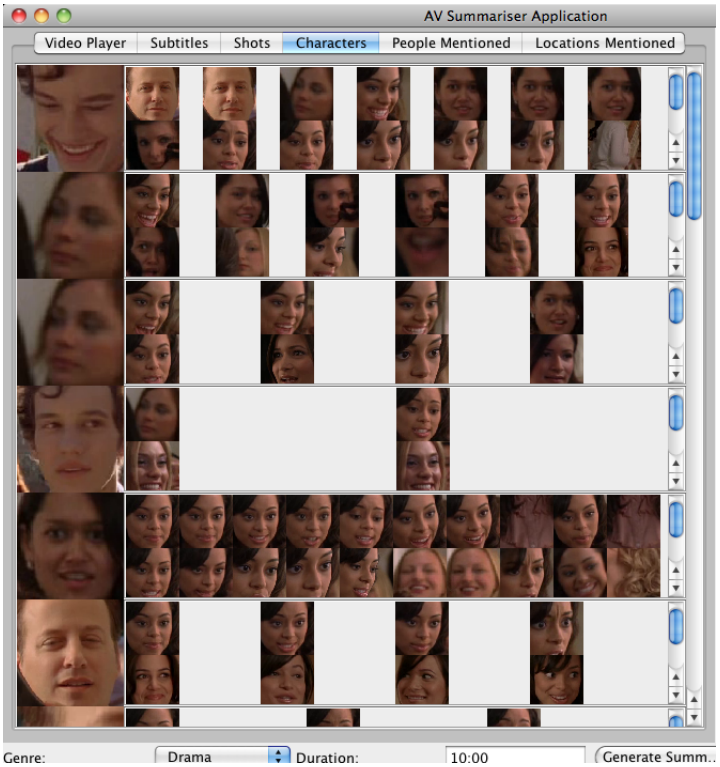

Figure 25: Histogram Comparison

As shown above, this theory did not work well in practice and therefore the search for a better matching solution continued.

Pixel Comparison of the center and boundary areas of the detected faces: This technique looked at the center square of a detected face, and the pixels in the center of each edge to try and find similarities. A screenshot showing the data gathered from this attempt is shown below:

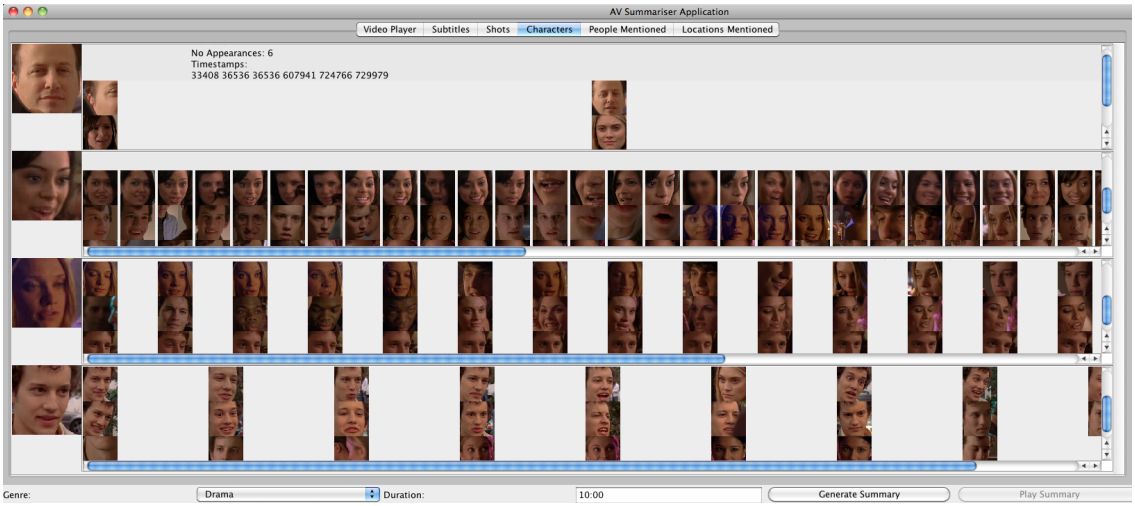

Figure 26: Pixel Comparison

As with the previous attempt, it can be seen from the image above that this theory also didn't work well in practice.

SIFT Comparison of Image Keypoints: This technique used an aligner to line up the images and then extracted and compared the image keypoints. It was initially designed to draw the similar points between images, however in this method the number of matches were logged and if two images had over three matches (the logic being that such points as eyes, nose and mouth would be the type of keypoints) then a positive match was logged. Two screenshots showing some of the best examples of this technique in action are shown below:

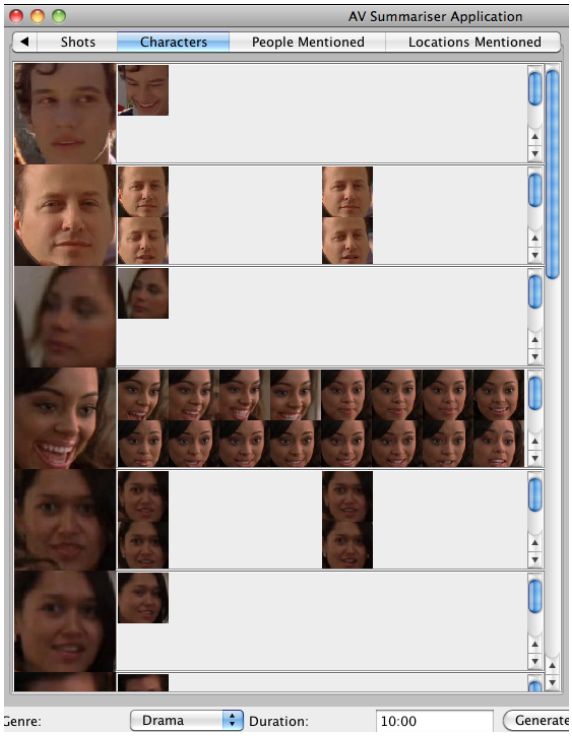

Figure 27: SIFT Comparison

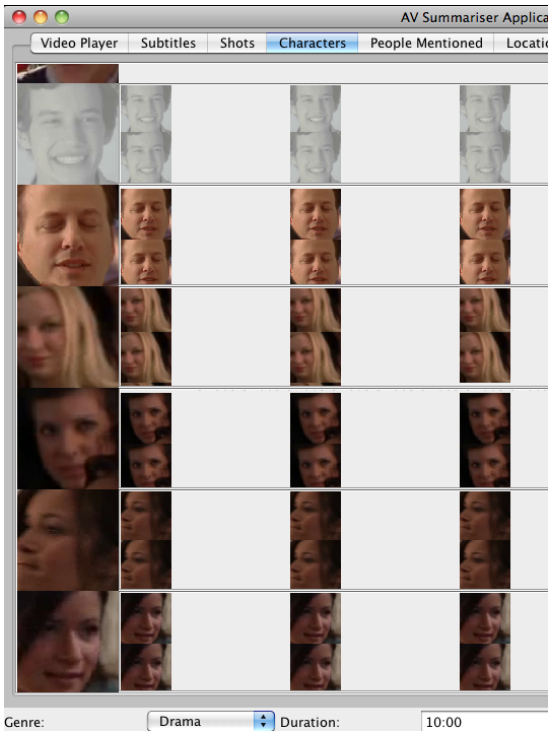

Figure 28: SIFT Comparison

In conclusion to this section, the fourth technique as demonstrated above was chosen as the final clustering technique as even though it's good results are highly dependant on a well lit video where the faces can easily be extracted, it is still by far the most successful method attempted.

Face Matching

The final step in this section was to match the characters to the actors playing them. The TVDB link as discussed previously (see Section 6.2) provides Objects with images of all the main characters of each series complete with actor names. For each character that contained enough matches, their images were added to a naive bayes recogniser and that was trained to recognise the different characters. Once this had been done, each series main character was tried against this to find the best match, if a match was positive then that character would be renamed by their actor name. This section could not be laboriously tested as it involved finishing the video in terms of facial recognition which meant that with the memory problems (discussed in a following section) it could only be tested on very short video clips. Nonetheless some experimentation was performed and below is a screenshot showing this:

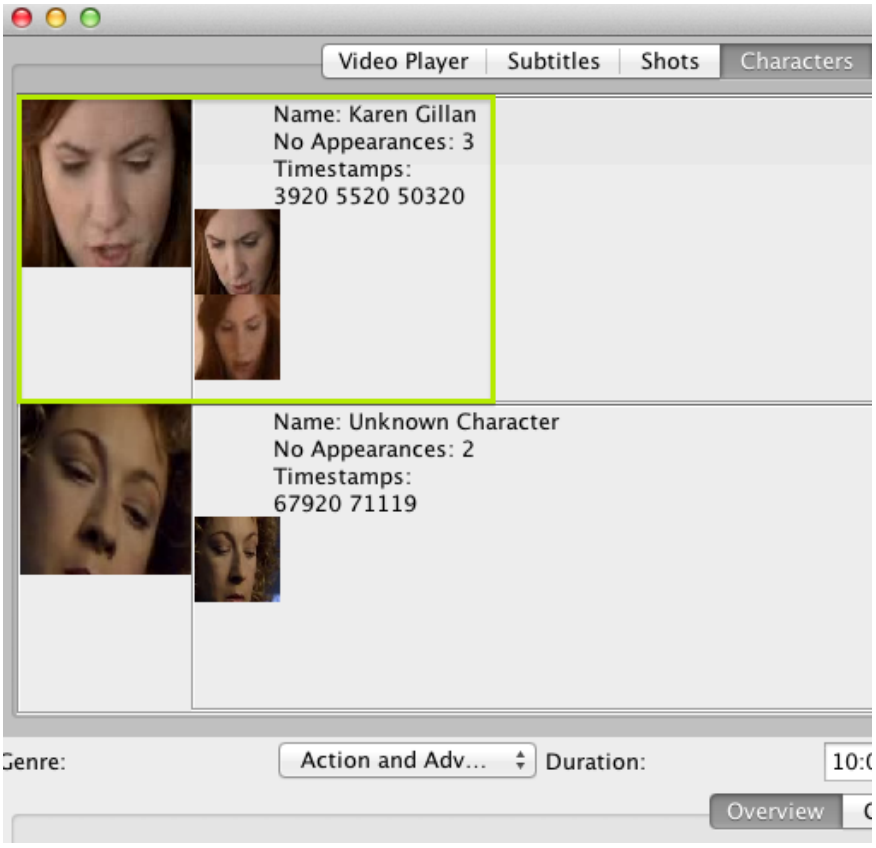

Figure 29: Pixel Comparison

As you can see from the picture above, it has had problems with identifying non-faces still so only one of the characters has enough images to be matched with an actor. However, the actor in this case has been correctly identified.

6.5.2 Class Diagram

The structure of the classes used within the package are shown in the class diagram below and detailed in further detail below.

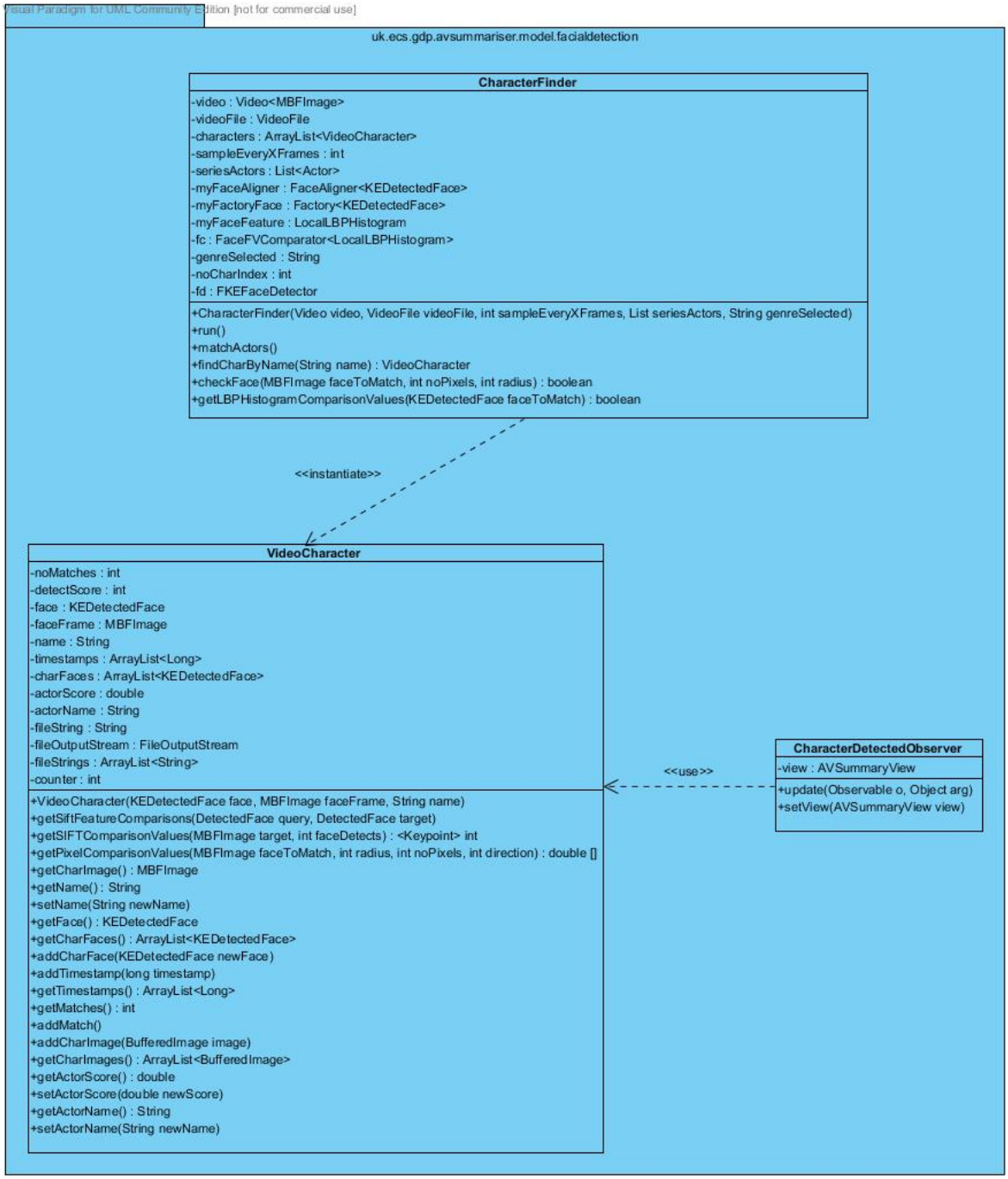

Figure 30: Facial Detection Package Class Diagram

6.5.3 Process Diagram

A diagram of how the different modules chain together to produce the facial detection and recognition is given in the following Figure:

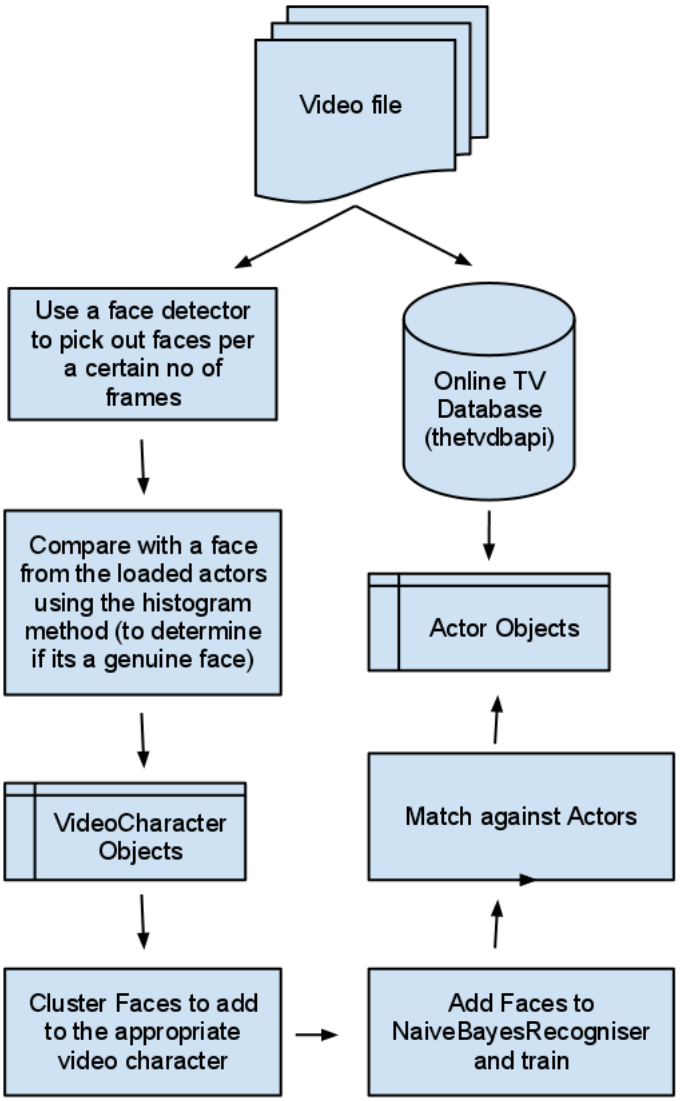

Figure 31: Facial DetectionProcess Diagram

6.5.4 VideoCharacter

A class which represents each individual character found in the video. It stores their image, timestamps and all the matching images that correspond to those timestamps. Due to memory problems instead of storing each matching image, these images are serialised and stored in a object output stream with references to the temporary files so that they can be read in at the appropriate time. Each VideoCharacter is displayed as an image in the Character panel and when clicked on expands to display the images of the character's appearances in the video and the timestamp links that jump to that area of the video.

6.5.5 CharacterFinder

A class which is responsible for locating, saving and matching the characters. It works in the following way:

- **Stage 1** - Lists all the identified faces in each frame that is looked at
- **Stage 2** - Performs the advanced face detection analysis on them to assess likliehood of being a face
- **Stage 3** - Checks to see if there are any existing characters that they are likely to match, if so that face is added to that character and its match value is incremented, if not a new character is created with that image
- **Stage 4** - Once it has clustered the characters, it puts them into a face recogniser and trains it
- **Stage 5** - That face recogniser then runs all the main characters produced by the TV Database (see Section 6.2) to see if any of those characters are a likely match and if so renames them accordingly.

6.5.6 CharacterDetectedObserver

This class implements the Observer interface and works with the Character Finder in Section 6.5.5 to draw the images of the detected characters in the character panel of the user interface. When the Character Finder wants to display a character it just has to notify the observer and that will call the appropriate methods in the GUI to draw their picture and show the corresponding information such as timestamps.

6.5.7 User Walkthrough

Despite the facial recognition section not being added into the final summary production, characters can still be found within the team's system. The two screenshots below demonstrate the functionality and how it may be used:

Step 1: Add a video, select it and add the series information (as shown in Section 6.2). Select "Find Characters" and move to the Character Tab. Once a face is displayed, select it to see corresponding matches and timestamps.

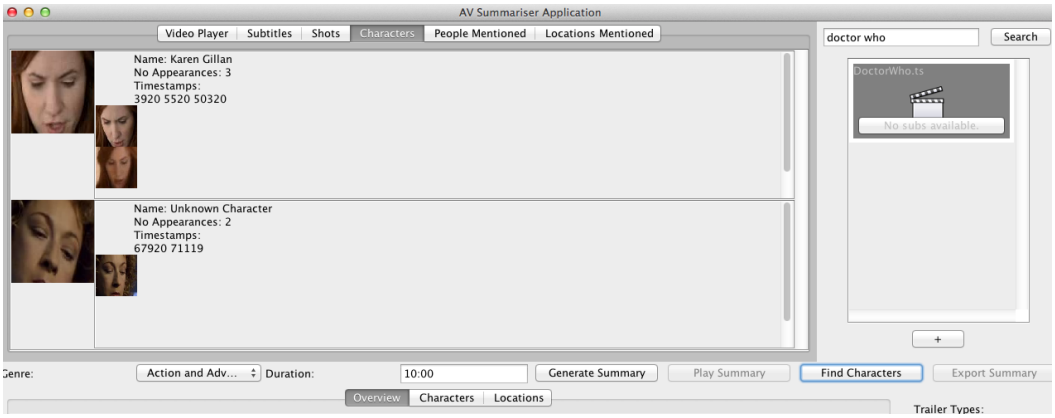

Figure 32: Selecting a video and series and viewing the faces as they are detected

Step 2: Wait for it to finish, and then click on the faces once again to see if they have been matched with an actor.

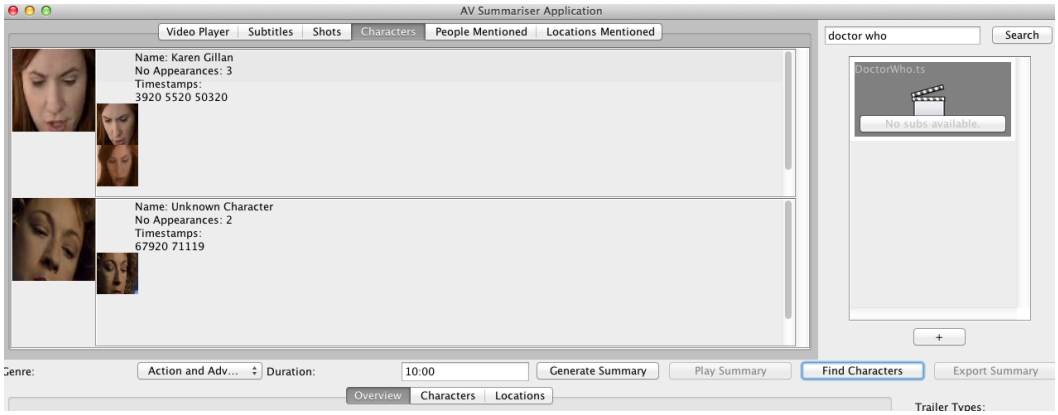

Figure 33: Viewing all the detected faces and their matched actors

6.6 Video Segmentation

6.6.1 Introduction

A key goal of the project was to enable segmenting a video in a way which appreciated how the video was made up. Some of the modules of the system will give a specific point in time (a timestamp) which represents something interesting, however this is not as useful when considering how the information should be passed back to the user.

A video is made up of shots which are then grouped into scenes. A scene is a series of shots in one location. An editor will use a variety of different shots, depending on the impressions that they wish to convey to the viewer. The lengths of shots or the density of shots can be used in a particular sequence to make inferences about what is happening in the video. For example, many varied short shots may represent an action sequence, or lengthy shots may be long shots where the film maker is setting the scene and showing little detail.

Furthermore, looking at the video shot data alongside the subtitle information, the shots may correlate with the different actors who speak. This could suggest a director is using over shoulder shots in a conversation. This information, particularly when combined with other data about the video, can be a powerful tool in summarising video content.

6.6.2 Software Implementation

The OpenIMAJ library on which the team is basing the project, has a video shot detector. This uses histogram differences to find the changes in shots.

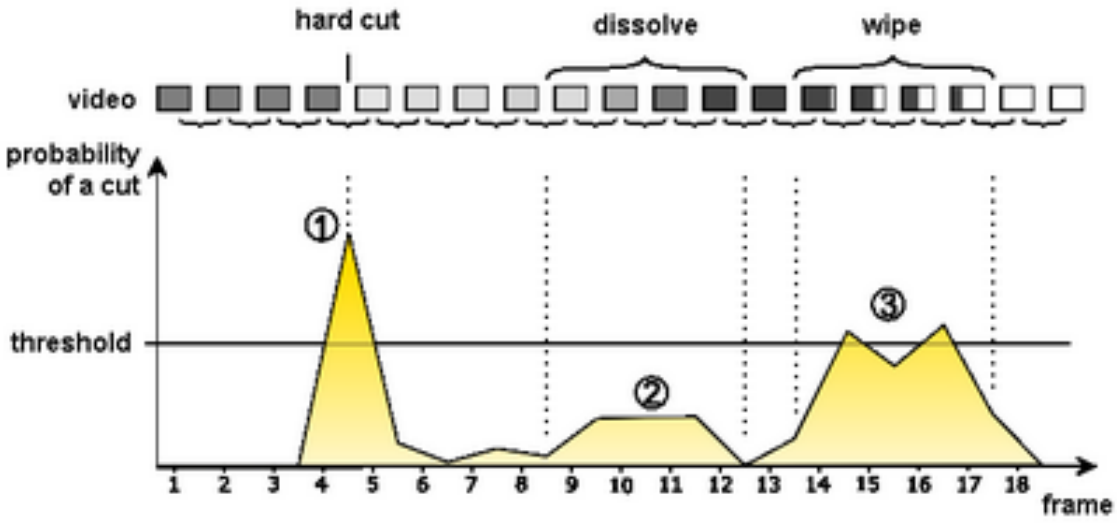

Figure 34: Shot Cut detection

It is down to the team to set the threshold or the sensitivity of the detector. This was done by simply watching the video and counting the frames, then checking to see if the shot processing returned a similar number. Shot detection is a hard problem for a computer to solve, it is relatively easy to fool the system into thinking that there is a shot change. For example in the screenshot below a man in a tunnel can be seen. The tunnel is unstable, so dirt is falling in front of the camera, which is detected as a large color change between frames so that a new shot is detected. This particular example could be addressed by reducing the sensitivity of the detector, but this is not always the case.

Figure 35: Incorrect cut in shot detection - threshold too low

Some events such as lighting in the video could not be ignored, so a few errors are likely to exist. On the whole however, errors are quite minimal and spread across a series the results are not noticeable. A listener is then attached to the video to fire an event when a shot change is detected. This provides a key frame which is scaled down to a thumbnail and keep in order to represent the shot to the user.

In the application, when the user clicks on a video, shot detection is started and the frame thumbnail representing each shot is output to the GUI, with a count of the number of shots and the time of the shot. This is for demonstration/informational purposes. The shot detection is threaded and shot detection can take place on multiple videos at any one time.

When the user chooses to generate a summary the system will ensure that the shots have been detected for all loaded videos. The data is then made available to the video summary builder.

6.6.3 User Walkthrough

Below are two screenshots indicating how to initiate shot detection:

Step 1: Add video and select it.

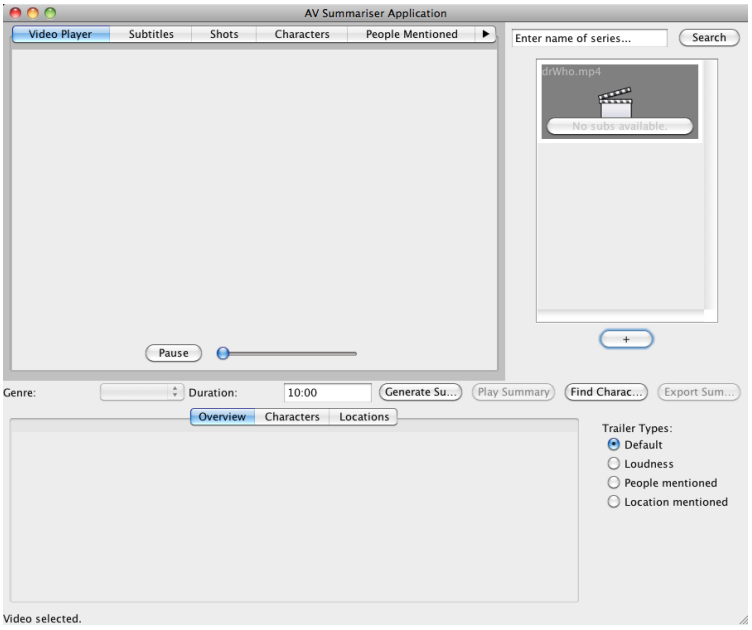

Figure 36: Adding a video and selecting it

Step 2: Press "Generate Summary" and the different shots will appear in the Shot Tab.

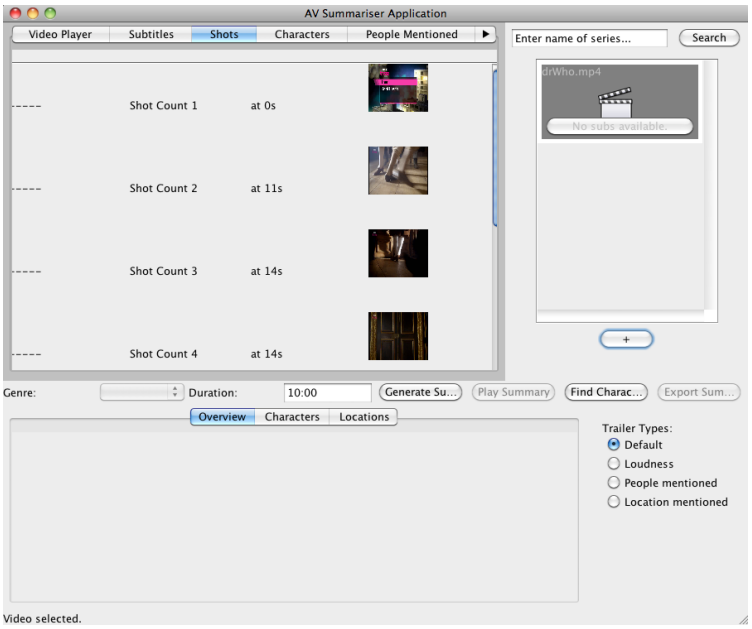

Figure 37: Viewing the Shot Detection Output

6.7 Summarisation

Package: uk.ecs.gdp.avsummariser.model.summary

This package contains all the Java code that produces the summary using all the data produced from the previous modules. This package has the following purposes:

- Using all the information produced by the system (excluding Frequency Detection and Face Detection which has been explained previously) and the user's metrics such as: Genre selected, Trailer duration, Trailer type.To produce a Summary object containing all the merged data and a trailer.
- Using a produced Summary object export all the data produced into three files which are:
  - A trailer for the television series.
  - A XML file called "summary_xml" which contains all the summary information such as metrics used, Series object information, Actor object information and all people mentioned with timestamps.
  - A XML file called "data_xml" which for each video file loaded contains:
    * The subtitle file associated with that video file.
    * All the data that the different modules produced for that video and subtitle file.

6.7.1 Process Diagram

A diagram of how the different modules chain together to produce the summary is given below:

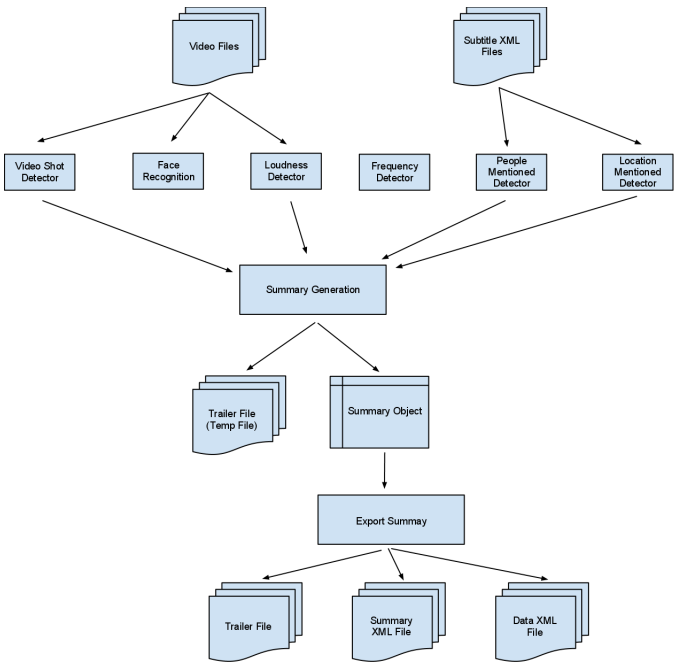

Figure 38: Summary Process Diagram

6.7.2 Class Diagram

The structure of the classes used within this package are shown in the class diagram below and detailed in further detail below.

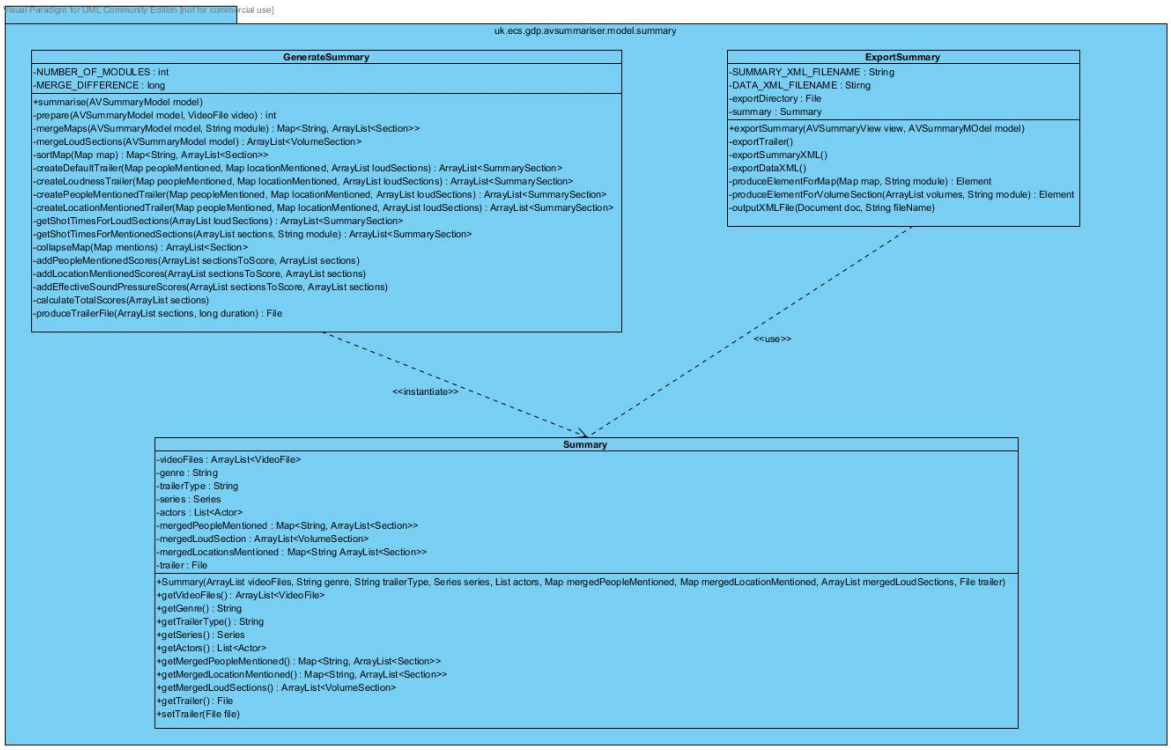

Figure 39: Summary Package Class Diagram

6.7.3 Summary

This class is a container object for all data that is gathered and produced for all the files that are loaded into the system

6.7.4 GenerateSummary

A static class which produces the Summary Object and video summary. How this is done depends on what metrics the user selects, which is discussed in the following stages:

Stage 1: Firstly, the system checks whether the different modules which are currently used have been ran and completed for each video file input. These various modules are:

- Video shot detection.
- Person name detection.
- Location name detection.
- Loudness detection.

If not then they are ran and the system waits until all modules have completed for all video files.

Stage 2: Next the following sets of data for each video are merged together:

- People names mentioned.
- Location names mentioned.
- Loud sections.

into a container for each.

Stage 3: Now the system chooses what video shots to use in the trailer. This depends on the trailer type the user selected currently these are the options:

- Default.
- Loudness.
- People Mentioned.
- Locations mentioned.

Which option the user selects becomes the priority in choosing video shots. This is how generally the video shots are chosen with further details underneath about what is chosen for each trailer type.

Stage 3A: An initial set of video shots is selected. This is done using either an ArrayList of Section or VolumeSection Objects. These are mapped to the VideoShots which contain them and are around three seconds in length.

- Default: VolumeSection Objects for loud sections used.
- Loudness: VolumeSection Objects for loud sections used.
- People Mentioned: Section Objects for people mentioned used.
- Locations Mentioned: Section Objects for locations mentioned used.

Stage 3B: Due to this rounding there may be repeats, overlapping or unable to be merged with the VideoShots Objects. These conditions are then checked for and the VideoShots are merged based on these conditions and the scores are adjusted as appropiate. This is done to produce an ArrayList of SummarySection Objects (these contain scores for each type of module used) and at the moment will contain only one type of score.

- Default : ESP scores.
- Loudness : ESP scores.
- People Mentioned : People mentioned scores.
- Locations Mentioned : Location's mentioned scores.

Stage 3C: Other score types are added to the SummarySection Object. For People and Location scores the scores are calculated as follows by doing the following for each Section Object and adding the individual scores together:

- If the start and end time of the Section Object is fully contained in this Summary-Section then add a one.

- Else if it starts and doesnt finish in the Section or vice versa then add 0.5.
- Else if not contained add zero.

Whereas for ESP scores it is the averge of the ESP score for the SummarySection Object plus the ESP of the current VolumeSection Object multiplied by x (Where x is either 1, 0.5 or 0).

- Default: People mentioned and Location's mentioned scores.
- Loudness: People mentioned and Location's mentioned scores.
- People Mentioned: ESP and Location's mentioned scores.
- Locations Mentioned: ESP and People mentioned scores.

Stage 3D: The total score is calculated for all SummarySection Objects (people names mentioned score plus Location names mentioned score).

Stage 3E: The SummarySection Object list is sorted in descending order according the trailer type chosen.

- Default: Total score then ESP score.
- Loudness: ESP score then Total score.
- People Mentioned: People score then Total score.
- Locations Mentioned: Location score then Total score.

Stage 4: Using the priority sorted SummarySection ArrayList produced, the Summary-Section Objects to go in the trailer are chosen while the total time is less than or equal to the target duration the user wanted.

Stage 5: SummarySection Objects are sorted back into video filename order and then time order, before creating a summary by splitting up the separate videos as required for the desired shots and are finally concatenated together to produce a summary.

Stage 6: Finally a Summary Object is produced containing the summary produced, all important merged summary data and individual data produced for each video.

6.7.5 ExportSummary

A static class presents a file chooser to the user to choose where to export the Summary object and all its content to. Once selected the trailer file is moved from the TEMP directory to where the user selected and two XML files are produced using JDOM[2] containing the information detailed earlier.

6.7.6 User Walkthrough

Below are six screenshots indicating how to generate a summary and the information that becomes available once it has been created:

Step 1: Load in videos, and subtitles (optional) and select "Generate Summary":

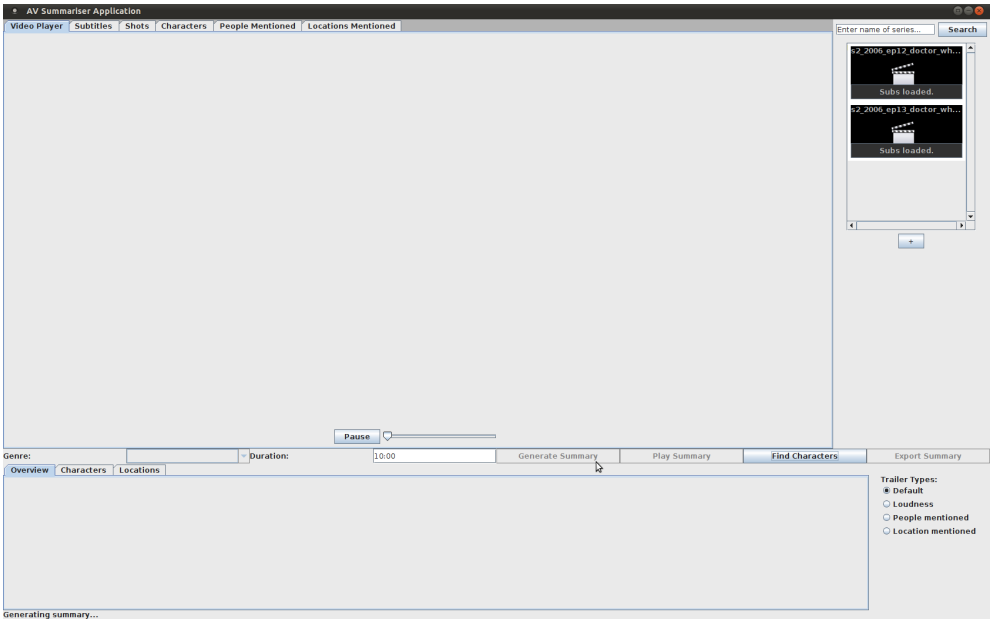

Figure 40: Pressing Generate Summary

Step 2: Observe the Overview Tab to view the information loaded in from the TV Database [9]:

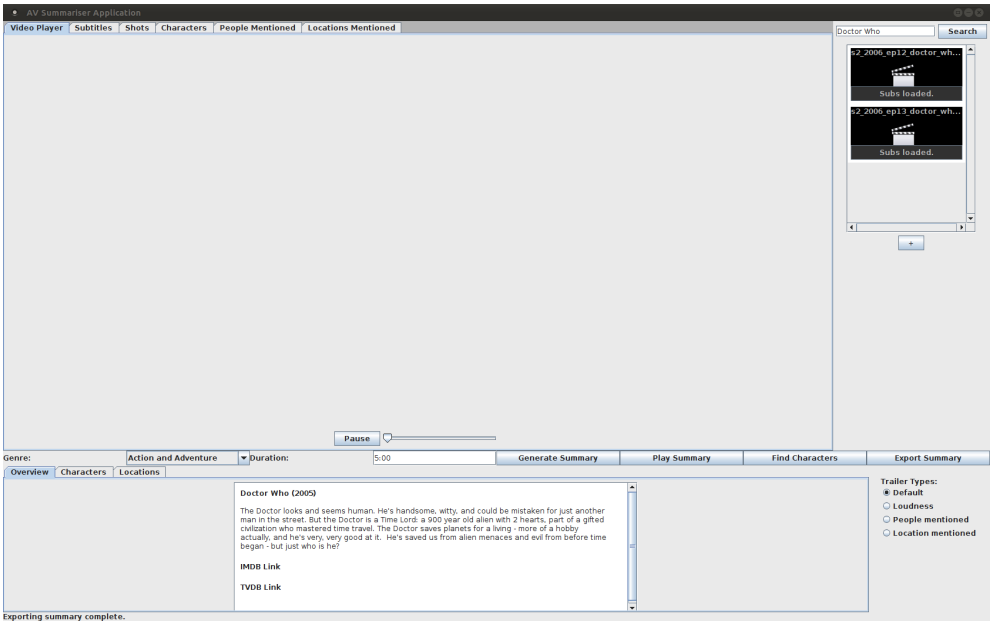

Figure 41: The Overview Tab

Step 3: Observe the Character Tab to view the information generated by the summary.

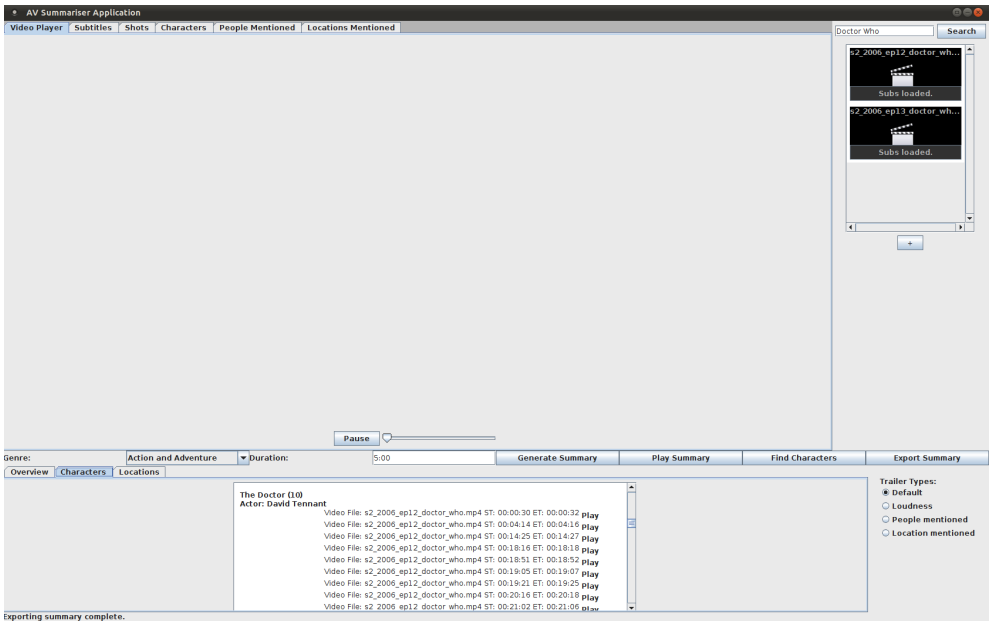

Figure 42: The Character Tab

Step 4: Observe the Location Tab to view the information generated by the summary.

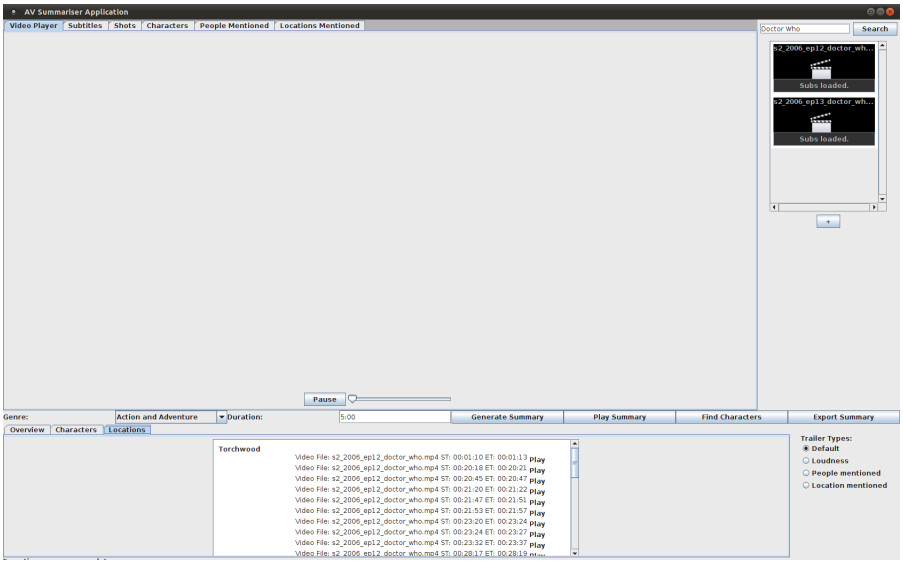

Figure 43: The Locations Tab

Step 5: Click on the "Play" button of a location timestamp to seek to that scene in the video.

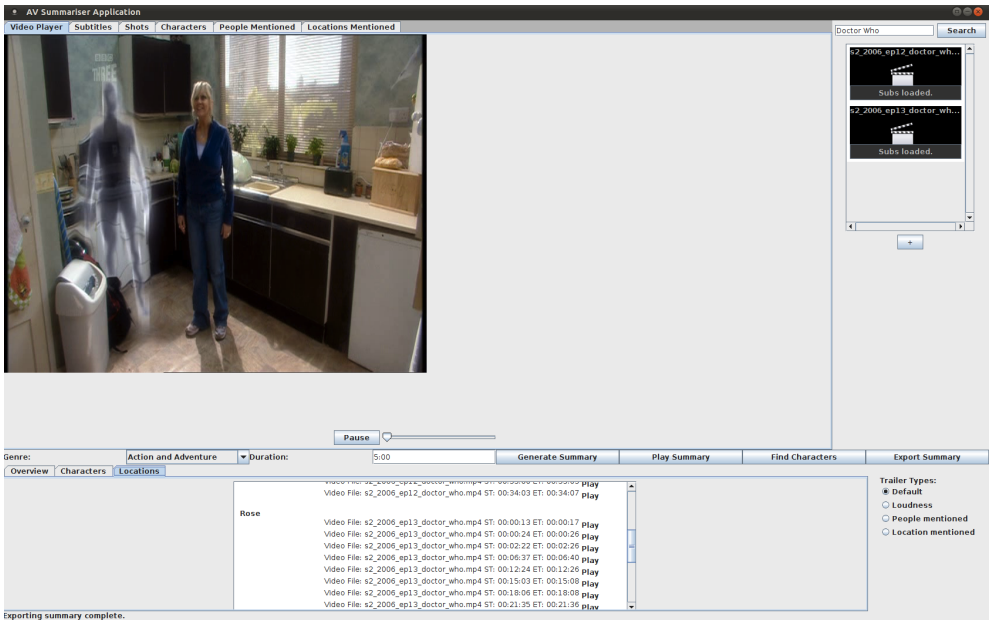

Figure 44: Using the Locations data to skip to that scene in the video

Step 6: Click on the "Export Summary" button and select where to export it to.

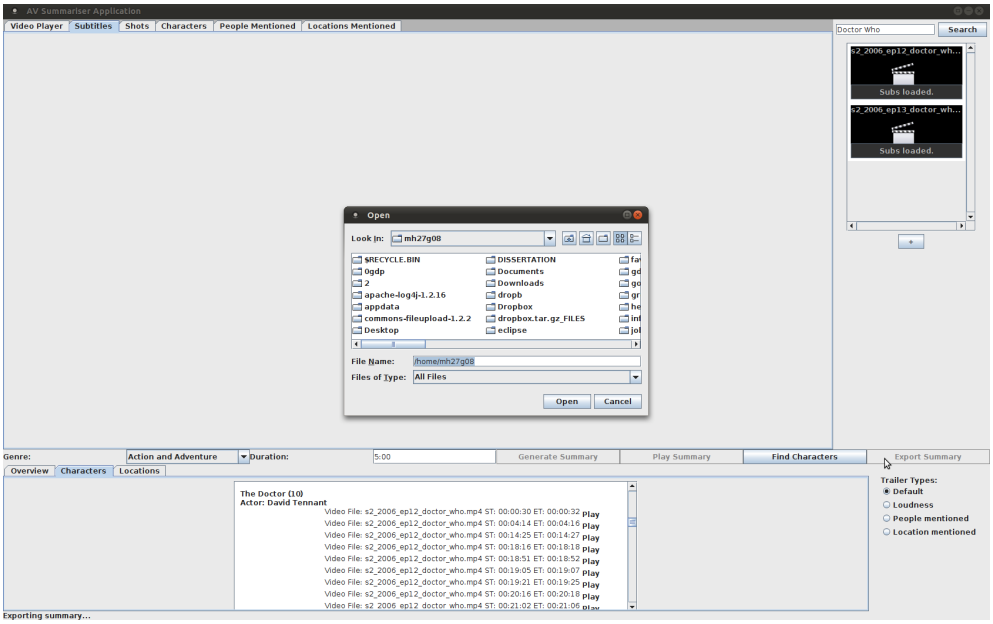

Figure 45: Exporting the Summary

6.8 User Interface

Packages:

- **uk.ecs.gdp.avsummariser.view**
- **uk.ecs.gdp.avsummariser.view.viewbrowser**
- **uk.ecs.gdp.avsummariser.view.pot**

The packages listed above contain all the View code which constructs the user interface; a screenshot of how the user interface looks is given below.

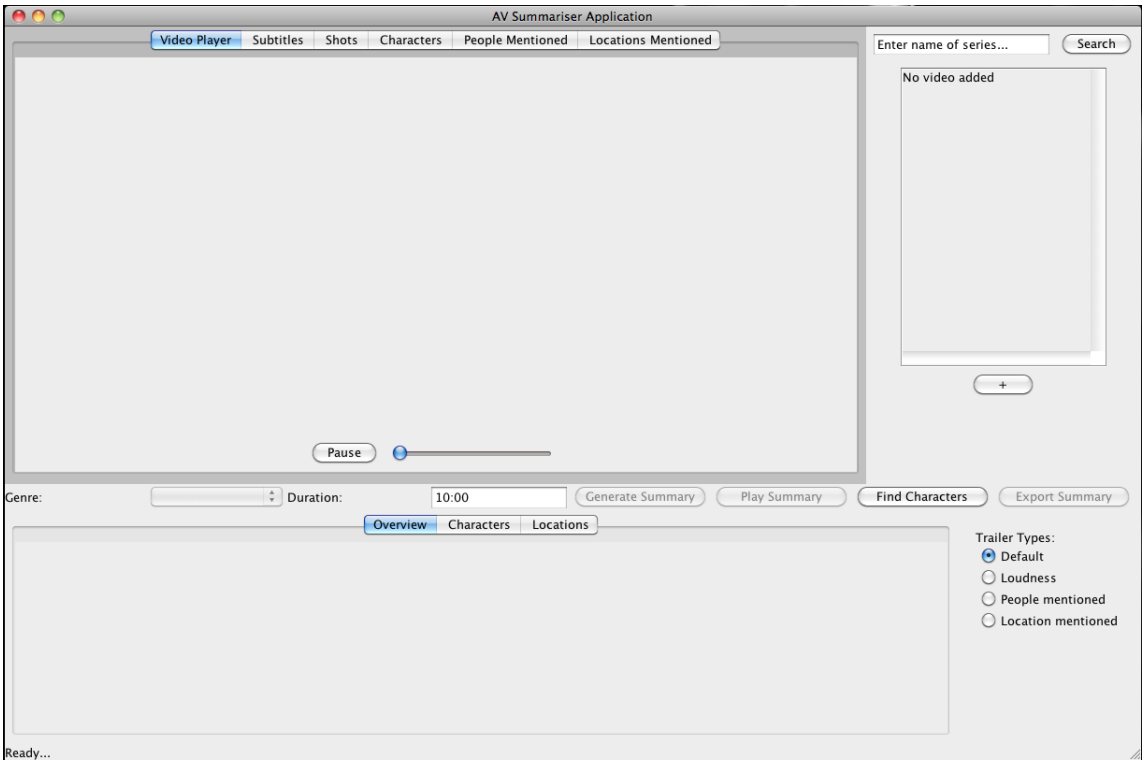

Figure 46: User Interface Screenshot

All the classes within these packages are either subclasses of JFrame, JTabbedPane or JPanel and come together to allow the user to view all output of system and control how the processing is done by entering/selecting summary metrics such as trailer target duration, the television series they wish to use, etc.

7 Code & UI Testing

In accordance with our previously decided testing strategy (see Section 2.9) these following sections detail those tests the team have performed upon the project as it has been developed and once it was completed. These tests comprise of unit tests, system tests and user goal testing. A complete list of these tests can be found in Appendix D.1.

7.1 Unit Testing

These tests ensure that the smallest modules of code work as intended. Shown here are some of those tests carried out by the team as the system was being developed. Most of these tests are over conditional sections of code such as if checks and for loops, which are essential to the author's intended functionality of the code. All of these test locations are prefixed by: uk.ecs.gdp.avsummariser.

Below is a table detailing ten of the most significant unit tests. The rest can be found in Appendix D in Section D.1.

Test	Test location	Test description	Expected result	Actual result	Success
1	model.audio. FrequencyDetector	For loop, does it exit when it should	Exit after ten iterations	Exits after ten iterations	Y
2	model.audio. VolumeDetector	In the method mergeLoudSectionsInVideo will the if check for the current selection being null work as intended	Enter an input of null and the if loop should be entered	If loop entered	Y
3	model.subtitles. PersonNameFinder	In the method findPersonNamesInSubtitlesUsingTVDB are all the Strings in the ArrayList looked at through the for loop	Should loop through the for loop for each String in the ArrayList, so enter ten Strings and the for loop shall loop ten times	Enters the for loop ten times	Y
4	model.summary. ExportSummary	In the method exportSummary there is an if check which checks to ensure that all a file extension is acceptable	All permitted file extensions should pass	Only permitted file extensions pass the check	Y

5	model.section.comparator.LocationMentionedComparator	In the method compare the if and else if check area was examined to ensure the correct region was entered for the entered input	First score is bigger than the second, should enter the first region	First region is entered	Y
6	model.section.comparator.LocationMentionedComparator	In the method compare the if and else if check area was examined to ensure the correct region was entered for the entered input	First score is smaller than the second, should enter the first region	First region is entered	Y
7	model.tvdb.SeriesSearch	In the method searchSeries enter the if region only when the user has entered a string	Enter nothing	Does not enter the if region, enters the else region instead	Y
8	model.subtitles.LocationNameFinder	In the method getAllLocationNames we test to see if the for loop iterates through all of the Strings in the ArrayList	Enter ten Strings and count that the for loop iterates ten times	The for loop iterated ten times	Y
9	model.subtitles.NameFinder	In the method setUpSentenceAndTokenModels we test to see whether the IOException try catch section works as intended	When an IOException occurs a stack trace should be printed out	A stack trace appears when an IOException occurs	Y
10	model.subtitles.NameFinder	In the method setUpSentenceAndTokenModels we test to see whether the FileNotFoundException try catch section works as intended	When a FileNotFoundException occurs a stack trace should be printed out	A stack trace appears when a FileNotFoundException occurs	Y

7.2 System Testing

These tests examine how the system as a whole works. These following tests are a subset of those already performed upon the team's project. Others tests may be found in Appendix D in Section D.2

Test	Test	Input	Expected Outcome	Actual Outcome
1	Load Video (Multiple video files)	MP4 video files	Loaded Successfully	Y
2	Select video (Non selected previously)	Video file without subs	Update display to show video selected and processing for video begins.	Y
3	Output tabs update correctly (Whilst not viewing)	Video file with subs	Output Tabs updated Successfully whilst viewing.	Y
4	Generate summary (Using metrics: genre, trailer type and duration)	Video file + Subs. Top Gear series selected.	Summary generated Successfully using metrics	Y
5	Generate summary (Multiple videos with mix of Sub files and none, TVDB info)	2 Video files. 1 + subs. 1 without TVDB info.	Summary generated Successfully.	Y
6	Play summary (Then play another video/summary)	1 Video file with subs. TVDB info loaded.	Video plays Successfully.	Y
7	Export summary (1 video without Subs, no TVDB info)	Video file without Subs. No TVDB info loaded.	Export summary Successfully. However due to development of this feature on Windows, on different operating systems, export directory plus file name to be exported always had Windows "\" file path seperator.	N
8	RETEST OF 7		Export summary Successfully. Null pointer exception thrown due to Map for people & location being empty per video.	N
9	RETEST OF 7		Export summary Successfully.	Y
10	Export summary again after complete (Checking Summary object in model has updated path)	Video file without Subs. No TVDB info.	Export summary Successfully and moved trailer succesfully.	Y

7.3 User's goals

These tests intend to look at how well the user's goals have been met by the team's finished system. Due to time constraints upon the team there was no time to prepare for a suite of user acceptance tests or for the testing to be carried out, as the team had initially planned. These tests aim to simulate in some ways how the user might interact with the system and to see whether it meets their expectations.

Test	Test description	Expected result	Actual result	Success
1	The ability to enter a range of different audiovisual formats	A number of different formats may be entered such as .ts and .mp4	Only .mp4 is accepted	N
2	Detection of shots within a video	The same number of shots should be detected by the system as a trained editor would find within a video, a team member found 347 in a short section of video	The system finds approximately the same number of shots as the team members do, the number it found was 361. This is mostly like down to the teams inexperience with video editing	Y
3	Detecting the points at which people appear in a video	Through facedetection and recognition the system should be able to find and identify faces in a video selection and record the moments these faces appear	Each detected character has all the points in which they appeared on screen recorded and skippable to	Y
4	Detecting those points at which characters are mentioned in the subtitles	Should record the time stamps of each occasion in which a character is mentioned in the subtitles	Each character has a list of those moments in which they were mentioned in the subtitles	Y
5	Detecting the points at which locations are mentioned in the subtitles	A list of points when a location is mentioned in the subtitles should be created	An accurate list of those moments in which locations are mentioned is created	Y
6	Detecting loud portions of audio	Audio levels should be recorded and loud portions can be used	Loud portions can be used as they are recorded down using their timestamps	Y
7	Detecting music in the audio	The system should be able to read in some audio and locate those sections which contain music	The system can not separate music from other audio	N

8	Key shot identification	Shots which are key to the video should be selected in some manner	Key shots are highlighted to allow the viewer to know these shots are deemed important in some manner	Y
9	Identify the main characters	Those characters which would qualify as main should be shown by the system	Those characters which appear frequently enough in a video are shown by the system to be main characters	Y
10	Identify the main locations	Those locations which are regarded to be main should be shown	Those locations which appear frequently enough to be regarded as main locations are shown	Y
11	Identify music	A given piece of audio should be able to be identified as a named track by a certain band	Audio samples of original music may be identified successfully as to the name of the song and the musicians who played it	Y
12	Summarise the input files based on user selected metrics	A summary or "trailer" should be produced for the entered TV shows, such a trailer should be concise, but also entertaining and placing an emphasise on the users chosen metric	A summary is created for any input audiovisual data and these summaries vary dependent upon the metrics chosen by the user	Y
13	Exporting a summary	A XML document should be produced to allow the summary details to be viewed else where and for the summary to be reproduced	A XML file is produced for a summary when desired	Y
14	Use OpenIMAJ	Using OpenIMAJ instead of recreating functionality	OpenIMAJ is used where possible	N

8 Evaluation

The team evaluates how they believe the team, the project management and the finalised project matches up to the project's goals 1.3.

8.1 Evaluation of the Final Product

Each of the different modules of the final product are evaluated below:

8.1.1 Evaluation of Person and Location Name Detection in Subtitles

This section evaluates the name and location detection which is performed per subtitle file in the system. This evaluation will be done as described in Section 2.10 using the follow TV programmes subtitles:

- Doctor Who 2005 Series 2 Episode 13 - Doomsday
- Top Gear Series 9 Episode 3 - US Special

These two programmes cover different genre types, the use of character and actor names as well as covering both fictional and non fictional names. This enables extensive testing of these detectors. An overview of these episodes is given in Appendix E in sections E.2 and E.4.

Prior to presenting the results, a few important points need to be made: firstly, the current system doesnt look to see if the same name is mentioned more than once in the same Subtitle object i.e. a line of speech, otherwise it would lead to repeated sections of the video which was deemed unnecessary. In addition, in order to count where shortened versions of names are used the system has been built to match instances such as "Amy" with "Amy Pond" in terms of name count. However this has led to wrong detections being made for example if the word Pond is mentioned it could refer to Amy Pond or Melody Pond and so both of these will count as a mention towards the respective characters.

The tables on the following pages present the results and the time taken to perform manual detection, natural language processing (NLP) for person name and location detection and also, using the list of actors/characters obtained from the Internet Television database.

Doctor Who 2005 Series 2 Episode 13 - Doomsday (Duration: 45 minutes)

Manual Person Name Detection

Name	Manual - Count	Manual - Main Character
Daleks	31	Yes
Cybermen	30	Yes
The Doctor	19	Yes
Rose Tyler	19	Yes
Mickey	10	Yes
Time Lord	10	
Jackie Tyler	6	Yes
Pete Tyler	6	
God	5	
Mum	5	Yes (Refer to Jackie Tyler)
Queen	4	
The Emperor	4	
Jacks	4	Yes (Refers to Jackie Tyler)
Dad	4	
Cyberleader One	3	
Dalek Thay	2	
Jake	2	
Stephen Hawking	1	
Chrissie	1	
Harriet Jones	1	
Jacqueline Andrea Suzette Tyler	1	Yes (Refers to Jackie Tyler)
Micketty-Mick-Mickey	1	Yes (Refers to Mickey)
Dalek Sec	1	
DalekJast	1	
Dalek Caan	1	
Mutt	1	
Jeff	1	
Tylers	1	
Total: 28	**Total: 175**	

Manual Location Detection

Name	**Manual - Count**	**Manual - Main Location**
Genesis Ark	14	Yes
Void	12	Yes
Torchwood Institute	10	Yes
Hell	5	
Planet	5	
Earth	4	Yes
Country	4	
Tardis	3	Yes
Sphere Chamber	3	
Petes World	2	
Universe	2	
Norway	2	
Darlig Ulv Stranden	2	
Staircase	2	
Room with the Sphere	1	
Parallel Earth	1	
Parallel Torchwood	1	
Arcadia	1	
Stairwell	1	
London	1	
Bergen	1	
Bad Wolf Bay	1	Yes
Total: 21	**Total: 64**	

OpenNLP Person Name Detection

Name	OpenNLP - Count	OpenNLP - Main Character
Daleks	31	Yes
Rose Tyler	22	Yes
Doctor	19	
Jackie Tyler	13	
Pete Tyler	11	
Mr Mickey	10	
Jacqueline Andrea Suzette Tyler	8	
Ah	6	
Lord	5	
Mum	5	
Sorry	3	
Wait	2	
Stephen Hawking	1	
Harriet Jones	1	
Jeff	1	
Total: 21	**Total: 138**	

OpenNLP Location Name Detection

Name	**OpenNLP - Count**	**OpenNLP - Main Location**
Rose	19	Yes
Ark	14	
Torchwood	11	
Earth	5	
Place	2	
Norway	2	
Nice	1	
Arcadia	1	
London	1	
Hope	1	
Total: 10	**Total: 57**	

TVDB Person Name Detection

Name	**TVDB - Count**	**TVDB - Main Character**
Rose Tyler	22	Yes
The Doctor (9)	19	Yes
The Doctor (10)	19	Yes
The Doctor (11)	19	Yes
Martha Jones	1	Yes
Total: 5	**Total: 180**	

Top Gear Series 9 Episode 3 - US Special (Duration: 60 minutes)

Manual Person Detection

TVDB Person Name Detection

Name	**Manual - Count**	**Manual - Main Character**
James	17	Yes (Refers to James May)
Richard Hammond	17	Yes
Jeremy	10	Yes (Refers to Jeremy Clarkson)
God	10	
Brokeback	5	Yes (Refers to Richard Hammond)
May	4	Yes (Refers to James May)
Big Stig	4	Yes (Refers to The Stig)
Katrina	4	
The Stig	3	Yes
Bernard Hopkins	3	
John Higgins	3	
Captain Slow	2	Yes (Refers to James May)
Murderer	2	Yes (Refers to Jeremy Clarkson)
Adolfs	2	
Andy	2	
Robert the Persian	2	
Carlos Tevez	2	
Joe Calazge	2	
Neil Robertson	2	
Nigel Bond	2	
Jezza	1	Yes (Refers to Jeremy Clarkson)
Mr Clarkson	1	Yes (Refers to Jeremy Clarkson)
Fat Stig	1	
Mr Big Mac	1	
Kojack	1	
John Wayne	1	
Holy Spirit	1	
Jesus Christ	1	
Burt Reynolds	1	
Gordon Ramsay	1	
George Bush	1	
Hillary	1	
Tom Hanks	1	
Total: 33	**Total: 111**	

Manual Location Name Detection

Name	**Count**	**Main Location**
America	8	Yes
Miami	6	Yes
Alabama	4	Yes
79th Street	3	
Florida	3	Yes
New Orleans	3	Yes
The South	3	Yes
Bagdad	2	
Earth	2	
Manchester	2	
Blackburn	2	
Bolton	2	
Arsenal	2	
Liverpool	2	
Aberdeen	2	
Las Vegas	2	
US	2	
Crucible	2	
Sheffield	2	
Bangkok	2	
England	1	
Silicon Valley	1	
Britain	1	
United States	1	
Jai-Alai Staduim	1	
Sunshine State	1	
Moroso Motorsport Park	1	
Brokeback Mountain	1	
Comfort Inn	1	
Best Western	1	
Red Lobster	1	
Deep South	1	
Belguim	1	
Louisiana	1	
Total: 34	**Total : 70**	

OpenNLP Person Name Detection

Name	**OpenNLP - Count**	**OpenNLP - Main Character**
James	16	Yes
Richard Hammond	13	Yes
Jeremy	9	Yes
Miami	6	
Ah	4	
John Higgins	4	
John Wayne	3	
Carlos Tevez	3	
Joe Calzaghe	3	
Neil Robertson	3	
Nigel Bond	3	
Robert	2	
Mercedes	2	
Sorry	2	
Andy	2	
Hi	2	
Tom Hanks	2	
Anyway	1	
Mac	1	
Lord	1	
Burt Reynolds	1	
After	1	
Gordon Ramsay	1	
Sadly	1	
Tonight	1	
George Bush	1	
Total: 27	**Total: 88**	

OpenNLP Location Name Detection

Name	**OpenNLP - Count**	**OpenNLP - Main Location**
America	7	Yes
Miami	6	Yes
United States	3	
New Orleans	3	
Alabama	3	
Las Vegas	3	
Florida	2	
Hi	2	
Manchester	2	
Crucible	2	
Sheffield	2	
Bangkok	2	
Silicon Valley	1	
Cougar	1	
Britain	1	
Nice	1	
Belgium	1	
Louisiana	1	
Total: 18	**Total: 43**	

TVDB Person Name Detection

Name	**TVDB - Count**	**TVDB - Main Character**
James May	19	Yes
Richard Hammond	13	Yes
Jeremy Clarkson	10	Yes
The Stig	5	Yes
Total: 4	**Total: 47**	

Data Analysis of these Techniques
Firstly, looking at the time taken for each of these processes which are shown below:

Person	**Doctor Who**	**Top Gear**
Manual	34 minutes	33 minutes
Natural Lanaguage Processing	5 seconds (0.2%)	5 seconds (0.3%)
TVDB	5 seconds (0.2%)	5 seconds (0.3%)

Location	**Doctor Who**	**Top Gear**
Manual	16 minutes	25 minutes
Natural Lanaguage Processing	5 seconds (0.5%)	5 seconds (0.3%)

The system performs this detection in a constant 5 seconds which is on average 0.3% of the performing manual name detection. Therefore, providing significant speed up in performing this processing. More importantly is how effective the name detection works, looking at person name detection first and the number of unique names the system produces.

Person	**Doctor Who**	**Top Gear**	**Average**
Manual	28	33	
Natural Lanaguage Processing	15 (54%)	27 (81%)	62%
TVDB	5 (18%)	4 (12%)	18%

Location	**Doctor Who**	**Top Gear**	**Average**
Manual	22	34	
Natural Lanaguage Processing	10 (45%)	18 (53%)	50%

On average NLP picks up 68% of the unique person names; comparing the two programmes the results are better for Top Gear which may be due to the use of non fictional names in Doctor Who such as Cybermen. In comparison, TVDB on average picks up 15% of the names mentioned which is due to the fact that the information on the database is only for the main characters that appear in multiple episodes/series for a television programme and therefore, wont pick up any names for non - main characters. For locations, NLP produces closer results for the two different programme types around 49% which as previously mentioned maybe due to the non-fictional nature of Doctor Who.

Looking at the counts per name:

Person	**Doctor Who (The Doctor)**	**Top Gear (Richard Hammond)**	**Average**
Manual	19	17	
Natural Lanaguage Processing	19 (100%)	13 (76%)	88%
TVDB	19 (100%)	13 (76%)	88%

Location	**Doctor Who (Genesis Ark)**	**Top Gear (America)**	**Average**
Manual	14	8	
Natural Lanaguage Processing	14 (100%)	7 (88%)	89%

For person names the different techniques produce the same number of counts once a name has been correctly identified so that on average for both these processes for a unique name it will find 88% of the occurrences of that name. This is very similar to location

names although however the 100% occurrence matching for these particular examples isnt normally the case for example for Rose Tyler it detects 115% of the occurrences which is incorrect and the reason behind this is discussed below.

Finally, looking into whether the system correctly identifies a main name. For person names, NLP and TVDB do identify the main characters correctly however for NLP it doesnt detect all the main characters unlike the TVDB technique which identifies all the main characters due to the information on the database. On the other hand for location NLP it identifies the main locations correctly for the Top Gear example, but is incorrect for this Doctor Who episode in saying that Rose is a location and furthermore a main location.

As a whole there are a few issues with using NLP and TVDB information to detect names and select the main ones. Firstly as mentioned the counts per name are off due to repeat character names in the same subtitle line not being looked for which would decrease the count but also these counts are affected by the matching of shortened names to the correct person i.e. Tyler to Rose Tyler, which leads to counts being added to the wrong names because the context of who the name refers to isnt detected i.e. whether Tyler refers to Rose or Jackie. Furthermore, both of these techniques cant detect and manually possibly couldnt either (if someone hadnt watched the episodes in question before) when alternative names are used to refer to a character for example in Top Gear, Jeremy Clarkson is referred to as Jezza and Murderer. Finally, there are incorrect detections in both techniques; for NLP Ah, Hello, Sorry are detected as Person names and Rose are for locations. This also applies to TVDB as in Doctor Who for example it detects Doctor Who as three different characters due to the different generations.

In conclusion as a whole the name and location detection constructed in this system drastically speed up the process of name detection in a video. Although the techniques produce varying results when finding all the unique names; once a name is detected the system will find almost all occurrences (88%) and identify the correct main names reliably, especially in using TVDB. However, there are issues which have been discussed and could be improved upon such as adjusting how checks are made and attempting to detect the context in which names are used to improve counts. As well as performing validation and restriction to remove false positives. This future work is discussed in Section 10.

8.1.2 Volume Detection Evaluation

The process of evaluating the Volume Detector has been done as specified in Section 2.10, through the use of manual inspection compared to the system production. The team has used the following TV programmes:

- Doctor Who (2005) Series 2 Episode 13 - Doomsday
- Top Gear Series 9 Episode 3 - US Special

and content of which are in appendix E in sections E.2 and E.4 respectively. The volumes of these shows were recorded as they sound to the listener, using a rough scale of 1 - 10 from silence to partially loud.

Using the system the graph feature in the system, a graph was drawn for each episode containing all the volume data for that programmes audio stream. The blue points are the volumes which the system identifies as loud points and the green points are all volumes. Below are the graphs for the aforementioned episodes.

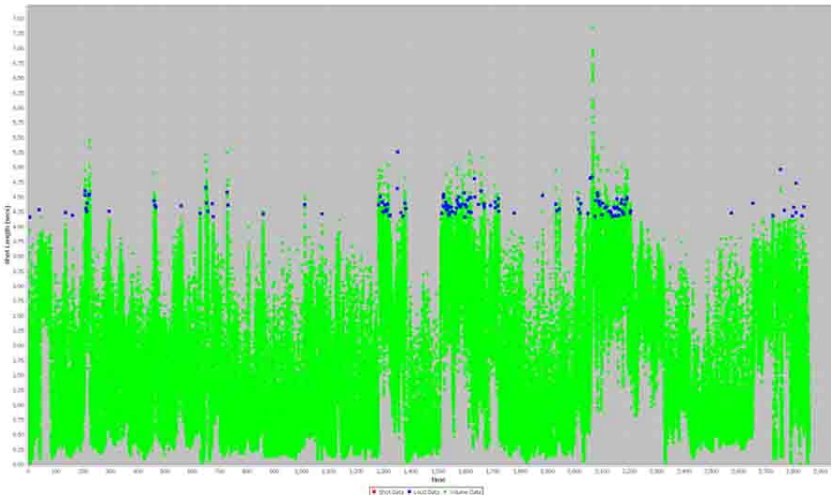

Figure 47: Doctor Who Loudness Graph

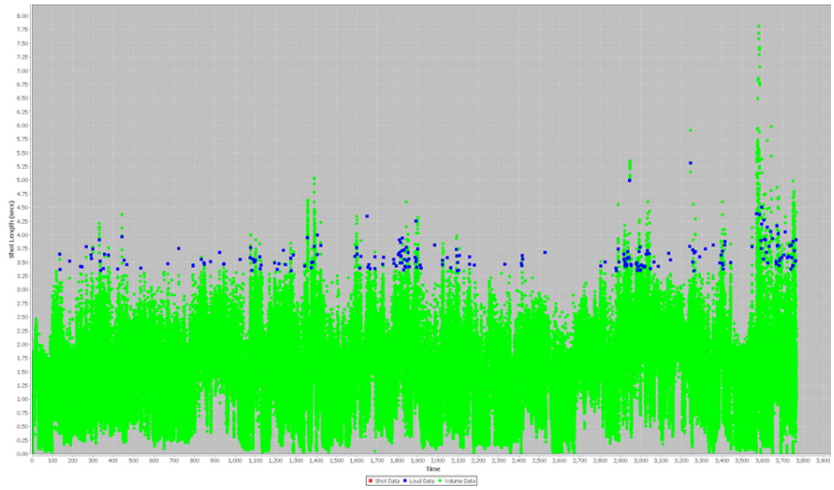

Figure 48: Top Gear Loudness Graph

These graphs were compared to the volumes recorded by the team baring in mind the volumes recorded by the team was subjective and not too accurate due to human error.

Areas noted as loud in our analysis matched the loud points plotted on the graph, likewise for quiet areas. However inaccuracies may be put down to the viewers inexperience.The volume detector performs as desired in detecting the loud sections and appears to be fairly accurate.

In terms of speed, the volume detection process took 10 minutes to complete comparing this to manual inspection of the data which took approximatly 75 minutes. Therefore our process takes 13% of the time to do it manually.

Volume detector functions correctly but unfortunately takes longer to process compared to other modules such as Name and Location Detection in Subtitles (see Section 6.3), therefore an area of future work for this module would be to improve its module would be to improve its performance.

8.1.3 Face Recognition

This element of the system remains unfinished as several problems still exist that the team did not have time to solve. Therefore this Section has been evaluated in terms of the different techniques used as opposed to its integration into the final product.

Facial Recognition Techniques: As detailed in the Facial Recognition Section (6.5) this section was broken down into three main components.

- Facial Detection
- Facial Clustering
- Facial Matching

In order to evaluate the facial detection technique upon which the other two elements build, a minute of footage was gone through manually and the faces were identified. These results were then compared to those produced by the facial recognition code which uses the functions provided by the OpenImaj team [8]. **NB: These evaluations are being made purely on the final techniques as detailed in the facial recognition section and only the genuine faces are being judged as the aforementioned section looks at the issue of false positives**.

Manual Inspection of Facial Recognition

Character Description	**Total Appearances**
Man in Wig	25
Guard 1	18
Guard 2	18
Girl at painting	10
Doctor	7
Rory	16
Amy	16

Final Facial Detection Technique

Character Description	**Total Appearances**	**Appearance Accuracy**
Man in Wig	16	(16/25) 64%
Guard 1	0	(0/18) 0%
Guard 2	0	(0/18) 0%
Girl at painting	0	(0/10) 0%
Doctor	0	(0/7) 0%
Rory	2	(2/16) 13%
Amy	12	(12/16) 75%

The table above shows that some of the characters were very well identified and others were not. There are however understandable reasons for why both the guards and the doctor were not identified. The guards both appeared in the scene behind the man in the wig and were smaller and less clear to see. A similar issue occured with Rory and Amy as they both appeared in the same set of frames, although Rory did manage to log two detections. The doctor's head was the only visible part of him but he was at the bottom of the screen

and rotated by about 90 degrees making him much harder to identify. Overall three of the characters were well identified and matched and those that appeared as sideline characters to a scene or weren't clearly displayed had less successful results.

Final Facial Clustering Technique

Character Description	**Faces Identified**	**Cluster Accuracy**
Man in Wig	10 in one set, 6 in another	(10/16) 63%
Rory	2	(2/2) 100%
Amy	11	(11/12) 92%

The table above shows the four characters that had their faces detected compared for clustering accuracy. Generally the clustering above produced good results with the images being clustered correctly with the only problems of the man in the wig being identified as two different characters, and some additional false positives that were matched with these faces.

Final Facial Matching Technique

Character Description	**Actor Match Success**
Rory	Not enough images to match
Amy	Successfully Matched

The table above shows the two characters that were detected and had the potential for matches as both are main characters within the series. Karen Gillan was successfully matched with Amy Pond but Rory was unable to be matched as there were not enough instances of his face to train the recogniser.

In conclusion to this, each element works well as far as building on the previous ones goes. However, it is difficult to determine how this would work running over a large set of videos as due to the memory problems of not being able to fully face detect a video this could not be tested further. However, each section has produced some successfull results and suggests that if the memory problems were solved and the issue of false positives was further tackled then these methods could be effectively combined to produce a significantly better facial recognition and detection system. Due to the problems this section was not encorporated into the summarisation element of the final product, however it can still be run individually through the system.

8.1.4 Evaluating Video Shot Detection.

Finding the breaks in video is a relatively straightforward thing for a person to do manually, but it is very time consuming, to do this accurately someone would have to watch the video back in normal time. If the user is only interested in counting the shots then a video can be processed at the speed the video is watched. However this only works for relatively short videos as the attention levels of the viewer have to be particularly high.

If the viewer is interested in recording the time point of each start of shot, the record keeping is more difficult. Writing down a time point on paper would involve looking away from the video which means that the user is likely to miss the next shot. This means the video will probably need to be paused periodically, depending on the video cut rate, this can easily make the time to find shots around double the length of the video.

In our system the shot detection for a 45 minute video, shot detection took 10 minutes 30 seconds. Our system shows a frame representing a particular shot in the GUI, it also presets the time that the shots appeared to the user. The shot time information is also then made available in the summary export and is plotted onto the graph. Again this would be a particularly time consuming task, and its speed of completion would depend on the skill of the viewer. This means that our application can complete the shot detection stage at around 8.5 times the speed of a human. This figure is likely to improve in our systems favour as the length of video to process increases due to the high level of concentration needed to work through material.

The developers spent some time adjusting the sensitivity of the shot detector, this was done by watching a section of video and counting the shots, then allowing the system to do the same. We achieved a good level of accuracy and the system tended to differ on very fast camera movements, which occasionally happened in our Doctor Who test video. Some special effects such as flashes would also fool the system. This was not a particularly large issue for the system, due to the infrequency of errors. In addition, sequences of shots are combined as metrics are combined, and therefore the chances of impact on the final video summary are quite low. With more time to develop this area, additional testing could be done on different types of video. However as previously mentioned, manually finding and recording shots is too time consuming to do at any length in a project of this size and scope.

8.1.5 Video Summary Evaluation

Our system produces a video summary based on the various modules of our system. This is based on:

- Person detection
- Location detection
- Loudness detection
- Shot detection

As mentioned in Section 6.5 and 6.4.7, the team were not able to integrate the face detection and recognition or music detection at this stage, however the video summary output is a good representation of the loaded videos as demonstrated below.

Summary Shot Distribution

Using three 45 min Doctor Who episodes, the team tested summary generation to check that a variety of shots from different episodes were included in the the video summary as well as checking that the duration parameter that the user sets restricts the maximum length of the video summary. The results of this testing are shown below:

- **Test 1:** Doctor Who (2005) Series 6 Episodes 1 - 3 (Episode length: 45 minutes).
- **Target time:** 10:00, actual 9:33.
- **Efficiency:** The time to summarise these videos was 16 minutes then a further 7 minutes to build the video summary. It takes less than a minute for someone to load in the videos and subtitles and perform a search of the open TV database. Therefore the total time is less than 24 minutes.

Outlined in the table below is a comparison of manual and automated processing by the application for these episodes:

Action (3 Episodes)	**Manual (minutes)**	**AVSummariser**
Person Finding	34 x 3 = 102	0:05 x 3 = 0:15
Location Finding	16 x 3 = 48	0:05 x 3 = 0:15
Shot Detection (Recording times)	45 x 3 = 135	10:30 x 3 = 31:30
Loudness detection	45 x 3 = 135	10.00 x 3 = 30.00
total	420	Multicore machine total: 16min

The shot distribution for two summary lengths are displayed below and are ranked in interest order i.e. most important first.

10 minute summary:

- Episode 1
- Episode 1
- Episode 1
- Episode 1
- Episode 3
- Episode 1
- Episode 1
- Episode 1

15 minute summary:

- Episode 1
- Episode 2

- Episode 1
- Episode 1
- Episode 1
- Episode 3
- Episode 3
- Episode 1
- Episode 2
- Episode 1
- Episode 1
- Episode 1

As can be seen from the two tests above, the interesting points of the three Doctor Who episodes tend to fall in episode one as this is the introduction to the series with many major events. However, the system clearly uses shots from all episodes and produces summaries that are either equal to or slightly less than the target length in all tests.

8.1.6 Video Summary Content Analysis

The video summary produced has also been evaluated to check that the most important points in a episode have been included. The television programmes which have been used are as follows:

- Top Gear Series 9 Episode 3 - US Special
- Top Gear Series 14 Episode 6 - Bolivia Special
- Doctor Who (2005) Series 2 Episode 12 - Army of Ghosts
- Doctor Who (2005) Series 2 Episode 13 - Doomsday

Appendix E contains the manual inspection of these episodes as well as the comparison of which points made it into the summary. Each test has been done by viewing each episode and documenting the interesting features. Using this data the output video summary could be watched and compared with the results noted alongside the original observations. Note this was done using the default trailer type.

Limitations of this approach are that it is subjective about what are the important points in an episode. This is a difficult problem to avoid which is why the team used a number of different videos however to improve test results this process could be done on varying television programmes and using a greater number of test subjects. But due to the time restriction of this project this wasnt possible.

As a whole the system picks up a good number of important points; the breakdown per episode is below:

Episode	No. of Interesting Points	No. of matches in Summary	Percentage
Top Gear Series 9 Episode 3 - US Special	17	2	12%
Top Gear Series 14 Episode 6 - Bolivia Special	14	3	21%
Doctor Who (2005) Series 2 Episode 12 - Army of Ghosts	24	6	25
Doctor Who (2005) Series 2 Episode 13 - Doomsday	8	2	25%

Table 10: Interest Point Matches

On average 21% of important points are identified which demonstrates that the system at this time didnt perform well for this experiment, which may be due to the particular trailer type being used or the teams opinion of the most interesting points. Therefore further testing is needed and there are areas which can be improved to increase the identification percentage are detailed in Section refsec:FutureWork. For example r- running the test on Top Gear Series 9 Episode 3 - US Special using the People mentioned trailer type, it produced a 24% match rate.

There are output videos which the system has produced in Appendix F are available on the CD.

8.1.7 Non-Functional Evaluation

The first non-functional requirement (NF1) was that the new system should be able to handle the BBCs file formats available from their research archive. The team encountered an issue at the video concatenation stage with the .ts file format. This was worked around by using the unix ffmpeg utility to convert the video to mp4. This is a commonly used utility which the software depends on through the Xuggle library so this was not judged to be a major issue by the team.

The software accomplishes the objective to export the data in a standard format. (NF2)The data is output in XML format and the video in MP4. This is compatible with all popular operating systems. The application has a status bar at the bottom of the GUI, which informs the user of progress at the various stages, most importantly through summarising stages which can take a significant period of time. The third requirement NF3: processing the broadcast data faster than a user could watch the video. This point has been covered in the other evaluation sections. The application is relatively efficient as it is multi-threaded (NF5), and the team recognised the significant impact that this had on performance. The application that the team have produced processes videos with separate threads for the various modules, this means that the application runs much faster on a modern machine with multiple cores than something that is a few years old. Finally, as already mentioned earlier in the project we have built the system upon the OpenIMAJ library.

8.2 Evaluation of the Team Dynamics

The team was formed the previous semester (Spring semester 2011) through shared friendships and the belief that each individual would work well with the others. The teams previous history working together through solo and group courseworks allowed each team member to know the strengths and weaknesses of the others before the project began. Such prior knowledge allowed the team to allocate job roles based upon their previous experience with one another as shown in 2.2. These job roles though quite vague, did prove to match each individuals tasks throughout the project.

To ensure the team never had serious discontent between team members it was decided at the offset of the project to have regular meetings within the team where issues could be aired and problems vented within the team throughout the week to ensure they would not become serious issues between team members. As shown in Section 2.5.2 the team met each Monday and Friday in a formal manner to discuss issues that had arisen over the weekend and during the course of the week. Such regular meetings meant that there were no issues between team members that were not resolved swiftly.

In addition to regular meetings, communication between these meeting times was also decided to be vital to the team. As may be seen in Section 2.4.1 the team had decided upon always being contactable through a number of methods, dependent upon time of day and their location. Each team member shared with the others their contact details and throughout the project each team member was capable of contacting the others through one or the other of the prescribed means with a generally swift response. A more concrete agreement about what means to contact each other through may have made the team more effective since different individuals often contacted one another through different mediums dependent upon their preferences so it was found that quite a few software tools and pieces of hardware had to be checked often to ensure that no communication from other team members was missed. Google chat was also under utilised as an instant messenger (IM) form of communication since it is a browser based form of IM. A better form of IM may have been hotmail or skype with a seperate piece of software which logs the user in when they connect to the web without their assistance since often members would be online, but not signed into Google talk.

The final core part of the team dynamics was pair programming. As with the others pair programming was a planned part of the team's works from the onset of the project, as shown in Section 2.6.3 which coupled with peer review (Section 2.6.2) was intended to create a shared knowledge base of the projects source code. Pair programming was under utilised the teams believes, though some pair programming was engaged upon not as much as was planned for was completed. Unfortunately what the team was trying to prevent from occuring did happen, where only one individual has knowledge of a section of code. Such specialisms did occur throughout the project often causing team members to be unsure of one anothers work and what was present else where in the code. Fortunately through the frequent meetings and communication channels these specialisms were not allowed to become an issue, but the coding of the project could have been better managed.

8.3 Evaluation of the Project Management

The planning and management of the project was all decided upon during the inital two weeks of the project. During these two initial weeks the team chose methods of development and planned their projected tasks for the whole project. In Section 2.4.2 and Section 2.3the team details their decisions upon the tools to use for project management and the method they chose for their software model, which was the iterative waterfall model. The chose of software model directly effects the ways in which the team is managed. For instance an agile method such as Scrums would result in a more individualistic approach to team management within a scrum, but then a team based approach to time management at the start of each scrum. The choice of the iterative waterfall model meant for the team that it was only once a task was completed was it possible to be reviewed and then potentially worked further upon as was deemed necessary. Such a model meant that the projects management was often not aware of the all the team members status upon their various tasks until the time for completion was reached, at which point their states were reviewed and often extended. That meant often problems with a task would not become visable to the team until it was at the point of becoming a problem. Preferably a model where team members were constantly updated of one anothers progress would have been used so that others could have become involved with one anothers work as assistance before the tasks became a team wide issue. As discussed in Section 8.2 the team did not complete as many tasks as desired using pair programming which exacerbated the issue of team members being unaware of one anothers current states.

Team management breakdown into two more specific areas, time management and resource management. The following sections evaluate how well these were done within the team.

8.3.1 Time Management

This section details how well the team progressed during this project and how effectively we managed our time and whether we achieved our project milestones. This evaluation will be done by comparing the initial gantt chart we produced at the beginning of the project and the current version; these are shown in Appendix B.

A table below compares the start and end time of each of our project stages:

Stage	**Initial Gantt Chart**	**Final Gantt Chart**
Planning		
Started	3/10/11	3/10/11
Completed	14/10/11	19/10/11
Design		
Started	5/10/11	5/10/11
Completed	19/10/11	21/10/11
Implementation		
Started	14/10/11	14/10/11
Completed	18/11/11	8/12/11
Verification		
Started	19/10/11	16/10/11
Completed	25/11/11	13/12/11

As shown in the table above the project schedule changed significantly between the initial and current gantt charts; the reasons why the project progress got delayed is discussed below.

Firstly, is that the team found particular tasks in this project difficult which is due to the lack of experience and knowledge the team has in this area, as identified in the risk analysis (see Section). Leading on from this point, the team underestimated the enormity of the tasks involved with this project in the Planning stage which is one of the reasons why some of the tasks discussed in this report are incomplete such as Face Detection and Music Detection.

Another reason is that the OpenIMAJ library [8] which the team used as part of the Implementation stage of the project was in development during the project and was presented at the ACM Multimedia 2011 [11] this led to the team having to overcome bugs and issues that were found whilst implementing and creating workarounds which then had to be re written once these issues were solved. Despite these problems, the library helped in completing the tasks involved within this project which the team probably wouldn't have completed without their help due to the teams lack of experience in this area as mentioned.

Thirdly, the team have had to deal with other module coursework clashes in particular COMP6017 Web Services which three members of this team are undertaking. Due to the difficulty and effort that was required for this piece of work, a team member had to allocate all their time to that coursework in order that it could be completed on time despite those team members working on this coursework from when it was received and balancing their time between this and that coursework. Therefore their work was rebalanced among the rest of the team members in order to keep progressing.

Finally, as the system was implemented and the different modules were integrated together it became apparent that the teams development environments were not suitable in terms of processing power and struggled in running all the integrated modules together to produce the final summary. Although this have an impact in the project progress as much as the other reasons stated.

In conclusion, the team has worked effectively to overcome these challenges in terms of planning ahead, re balancing work as necessary and communicating with the OpenIMAJ development team to solve any technical issues the team had during the Implementation stage of this project.

8.3.2 Resource Management

The sole real resource other than time that the team's project had were the team members themselves. As previously discussed in Section 2.1 each team member has their own personal strengths and weaknesses thus managing those tasks which team member, or recourse is deployed upon is critical to ensure effective management of these resources. For instance a resource which is designated the lead programmer through their strong programming experience should not be deployed upon documentation tasks when there are other programming tasks which they could be utilised more effectively on.

Such resource management was performed fairly effectively through selfelective task choosing from the team members. Each team member chose from the upcoming tasks those tasks which they felt they would be best at. Such selfselection of tasks did result in fairly effective resource deployment though occasionally larger or more challenging tasks were left to whoever was willing to or was voted suited for task to pick up rather than chosen by a team member.

A more effective method of resource management would be preferable in the future, but without a manager who knows the capabilities of each team members as intimately as

themselves the balance between what a resource can achieve well and what they enjoy would not be reached. As far as the team is aware no more effective and fair method could be found than the one they used, but as shown it did have its failings.

9 Conclusions

The goal of the teams project was to create a software tool designed to assist the BBC's archiving process by automatically extracting and summarising audiovisual data for the purpose of assisting in archiving and supporting content reuse.

The team accomplished this by producing a multiplatform application which brings together various video and text analysis technologies in order to extract meaningful and reusable metadata. This has been achieved by implementing modules that perform: Natural Language Processing on Subtitles, key interest points identification in video and audio features. Followed by presenting this information to the user. The results are produced by discovering: Person and Location names mentioned in Subtitles, high volume audio sections and querying an external data source. The final product also includes some facial detection and analysis functionality, however this has not been fully incorporated into the final system.

The team has investigated a great number of areas and has produced a system that is likely to be used in future projects. Future work that could be completed as part of this project is detailed in the following section.

10 Future Work

The scope of the project was very large since the team was given many potential features from their client in the early stages. As touched on in some previous sections of this report there were some requirements which the team would have worked on if there was more time available, as well as some new ideas which surfaced whilst the project has been developed.

The first stage of future work would be to perfect the existing parts of the project which due to time constraints were not fully implemented or may not be performing optimally. Additionally further testing may have helped to tweak the summary type settings to produce better video summaries. The following paragraphs discuss extensions for the system.

The team would like to expand the system to accept a greater number of video types. The system currently works with MP4 files, whilst it is easy to convert to this format externally, it would be more user friendly for the application to handle this.

The name detection process which is currently used within the system could be improved to increase the accuracy of detecting names and their occurrences within Subtitles. Improvement to this module could be to extend the occurrence name search process to look for repeats of the same name with a Subtitle speech line, which the system currently doesnt do because the team are only interested in when different names occur. Furthermore this module could perform context analysis of a sentence to find out which person is referred to with more accuracy i.e. Tyler could be for Rose Tyler or Jackie Tyler. Improvements could also help with removing false positives in analysis in particular where person names are detected as the main locations.

Another future area to investigate would be music detection since due to the limited time available and the lack of suitable libraries meant that this feature wasnt completed. This is something that the team would like to have worked on, but it is believed the module could represent a new project in itself. This would then be integrated with the existing Frequency Detector and Music Identification modules in the system.

Facial detection and recognition was not quite complete in the final system. One of the first issues to tackle would be memory issues, particularly perfecting the balance between the application storing too much data, but also having enough information available to make predictions. Also expanding SIFT comparison to work in the same way as it does for facial key points. Additional extensions that could also enhance the system would be to able to match every character to their corresponding actors, as opposed to just the main ones.

Currently the output video is very obviously computer generated. The developers have not yet invested time into shot transition and would like to look into this in order to improve the viewing experience. When processing particular videos there is some unexpected audio noise, which could not be investigated in the time available for the project, but is an important issue to address. A further addition to the module which has been identified during development is ensuring a good range of shot length in the final output. This would mean analysing the shot and video segment density and making adjustments before the final video summary is built. Additionally the developers would like to investigate alternative techniques to rank the shots for the video summary.

An area mentioned by our client was to investigate how video which had not been broadcast could be analysed to see whether additional programmes can be made with the excess footage. This was not something that was in the team's final list of requirements for a

number of reasons. In order to do this the system would need some sample footage of a program where this use case applies. It would also involve processing a lot more data, and therefore increase the time needed to develop and test this type of functionality. The team would also need to learn more about the type of content and editorial decisions in order to make better judgements. Overall this would mean more interaction with the BBC, which is difficult to organise and execute in such a short project.

A final addition to the product would be to extend the natural language processing used to identify the main topics. That would likely make the textual summary more informative. In order to do this well there would need to be some comparison within an episode and also across a series. Potentially the system could use the TVDB to assist with this, possibly looking at titles of episodes also. The way a topic would be discovered and considered accurate is less well defined than names and locations for example. Therefore it would take some time to implement and test, particularly assessing its accuracy is difficult as even a group of video viewers can struggle to agree on key themes.

References

[1] Echonest - http://the.echonest.com/.

[2] Jdom library - http://www.jdom.org.

[3] Jen api - http://code.google.com/p/jen-api/.

[4] Jfreechart - http://www.jfree.org/jfreechart/.

[5] Json simple - http://code.google.com/p/json-simple/.

[6] Maven - http://maven.apache.org/.

[7] Open nlp library - http://incubator.apache.org/opennlp/.

[8] Openimaj - http://sourceforge.net/p/openimaj/home/openimaj/.

[9] The television database - http://thetvdb.com/.

[10] The television database library - http://code.google.com/p/javatvdbapi/.

[11] Acm multimedia - http://www.acmmm11.org/content-awards-recognitions.html, 2011.

[12] L. Ballan, M. Bertini, A. Del Bimbo, and W. Nunziati. Soccer players identification based on visual local features. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 258–265, New York, NY, USA, 2007. ACM.

[13] Vítězslav Beran, Michal Hradis, Pavel Zemcik, Adam Herout, and Ivo Řezníček. Video summarization at brno university of technology. In *Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, TVS '08, pages 31–34, New York, NY, USA, 2008. ACM.

[14] A.B Bondi. Characteristics of Scalability and their Impact on Performance. In *Proceedings of the 2nd International Workshop on Software and Performance*, pages 195 – 203, 2000.

[15] A Cockburn and L Williams. The costs and benefits of pair programming. In *Proceedings of the First International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2000)*, 2000.

[16] Saman Cooray, Hyowon Lee, and Noel O'Connor. A user-centric system for home movie summarisation. In Kuo-Tien Lee, Wen-Hsiang Tsai, Hong-Yuan Liao, Tsuhan Chen, Jun-Wei Hsieh, and Chien-Cheng Tseng, editors, *Advances in Multimedia Modeling*, volume 6523 of *Lecture Notes in Computer Science*, pages 424–434. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-17832-0_40.

[17] M. Everingham, J Sivic, and A. Zisserman. "Hello! My name is... Buffy" - Automatic Naming of Characters in TV video.

[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison Wesley, 1st edition, January 1995.

[19] Zhu Liu, D. Gibbon, and B. Shahraray. Web-based real time content processing and monitoring service for digital tv broadcast. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2010 IEEE International Symposium on*, pages 1 –6, march 2010.

[20] Zhu Liu, Eric Zavesky, Behzad Shahraray, David Gibbon, and Andrea Basso. Brief and high-interest video summary generation: evaluating the at&t labs rushes summarizations. In *Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, TVS '08, pages 21–25, New York, NY, USA, 2008. ACM.

[21] Daniel Mcennis, Cory Mckay, and Ichiro Fujinaga. Jaudio: A feature extraction library. In *International Conference on Music Information Retrieval*, 2005.

[22] Cory Mckay and Ichiro Fujinaga. jmir: Tools for automatic music classification. In *Proceedings of the International Computer Music Conference*, 2009.

[23] H Mills, M Dyer, and R Linger. Cleanroom software engineering. *IEEE Software*, 4(5):19–25, September 1987.

[24] Arthur G. Money and Harry Agius. Elvis: Entertainment-led video summaries. *ACM Trans. Multimedia Comput. Commun. Appl.*, 6:17:1–17:30, August 2010.

[25] Paul Over, Alan F. Smeaton, and George Awad. The trecvid 2008 bbc rushes summarization evaluation. In *Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, TVS '08, pages 1–20, New York, NY, USA, 2008. ACM.

[26] F. Patin. Beat Detection Algorithms. Gamedev.net, 2003.

[27] W. Royce. Managing the development of large software systems. In *Proceedings of IEEE WESCON*, volume 26, pages 1–9, August 1970.

[28] M. Sano, H. Sumiyoshi, M. Shibata, and N. Yagi. Generating metadata from acoustic and speech data in live broadcasting. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 2, pages ii/1145 – ii/1148 Vol. 2, march 2005.

[29] Alan F. Smeaton, Bart Lehane, Noel E. O'Connor, Conor Brady, and Gary Craig. Automatically selecting shots for action movie trailers. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, MIR '06, pages 231–238, New York, NY, USA, 2006. ACM.

[30] I Sommerville. *Software Engineering*. Addison Wesley, Pearson Education, 8th edition, June 2007.

[31] Yoshiharu Suga, Naoko Kosugi, and Masashi Morimoto. Real-time background music monitoring based on content-based retrieval. In *Proceedings of the 12th annual ACM international conference on Multimedia*, MULTIMEDIA '04, pages 120–127, New York, NY, USA, 2004. ACM.

[32] Victor Vald&3233;s and José M. Martínez. Introducing risplayer: real-time interactive generation of personalized video summaries. In *Proceedings of the 2010 ACM workshop on Social, adaptive and personalized multimedia interaction and access*, SAPMIA '10, pages 9–14, New York, NY, USA, 2010. ACM.

[33] Víctor Valdés and José M. Martínez. Binary tree based on-line video summarization. In *Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, TVS '08, pages 134–138, New York, NY, USA, 2008. ACM.

[34] L Williams and Kessler R. Pair Programming Illuminated, 2003.

Appendices

A Agreed Project Specification

Title: Summarising Audiovisual Collections
Supervisor: Paul Lewis
Team Members: Michael Harris, Jonathan Harrison, Sami Kanza, Jenny Lantair
Customer: Matthew Addis at IT Innovation

Project Specification:

Background
The BBC has an audio visual archive, stretching back 50 years, which stores media broadcasted by the BBC for the purpose of preserving content and for compliance reasons. This archive is currently catalogued by hand which is very time consuming, for example it takes 10 hours to process 1 hour of material.

Problem
The problem is that there is limited metadata attached to this media therefore searching and summarising their content is extremely difficult. We have been given the task of producing Audio/Visual summaries of a collection; whereby a AV summary is a complete subset of a collection which represents the key elements of its content. Ideally, the AV summaries produced will be defined by a users metrics. These summaries could then be used applications such as Archive Monetisation, Rights Assessment and Archive Content Selection.

Goals and Deliverables

The goals of this project are:

- To build a AV summary of a collection using metrics such as:
  - Main characters.
  - Non key / guest characters.
  - Key scenes.
- To support AV data in MPEG2 format and DVB subtitle data in W3C Timed Text Markup Language.
- To be able to process material faster than the current method of cataloging.
- If possible, construct a modular system to allow integration into existing Open Source tools.

The deliverables of this project are:

- A summarisation engine with a basic interface which allows users to produce an AV summary of a collection based on their parameters.

B Early Gantt Charts

B.1 Initial Gantt Chart

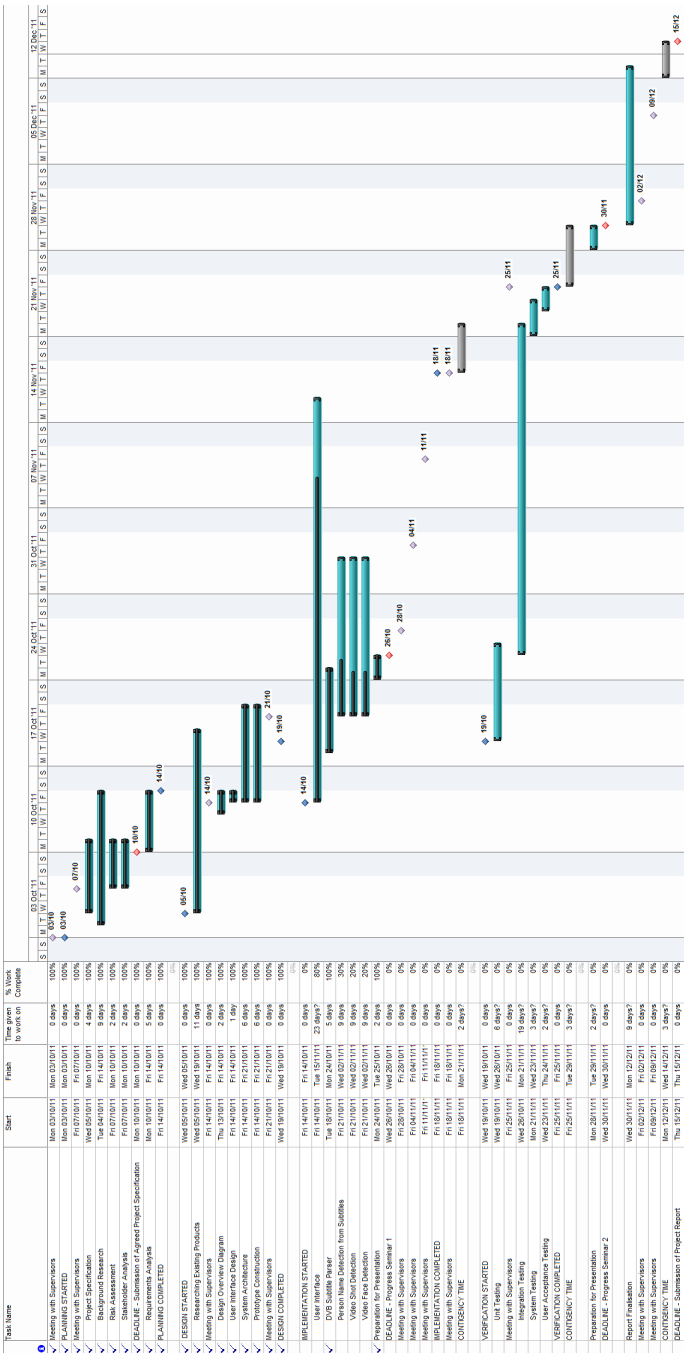

Figure 49: Initial Gantt Chart

B.2 Final Gantt Chart

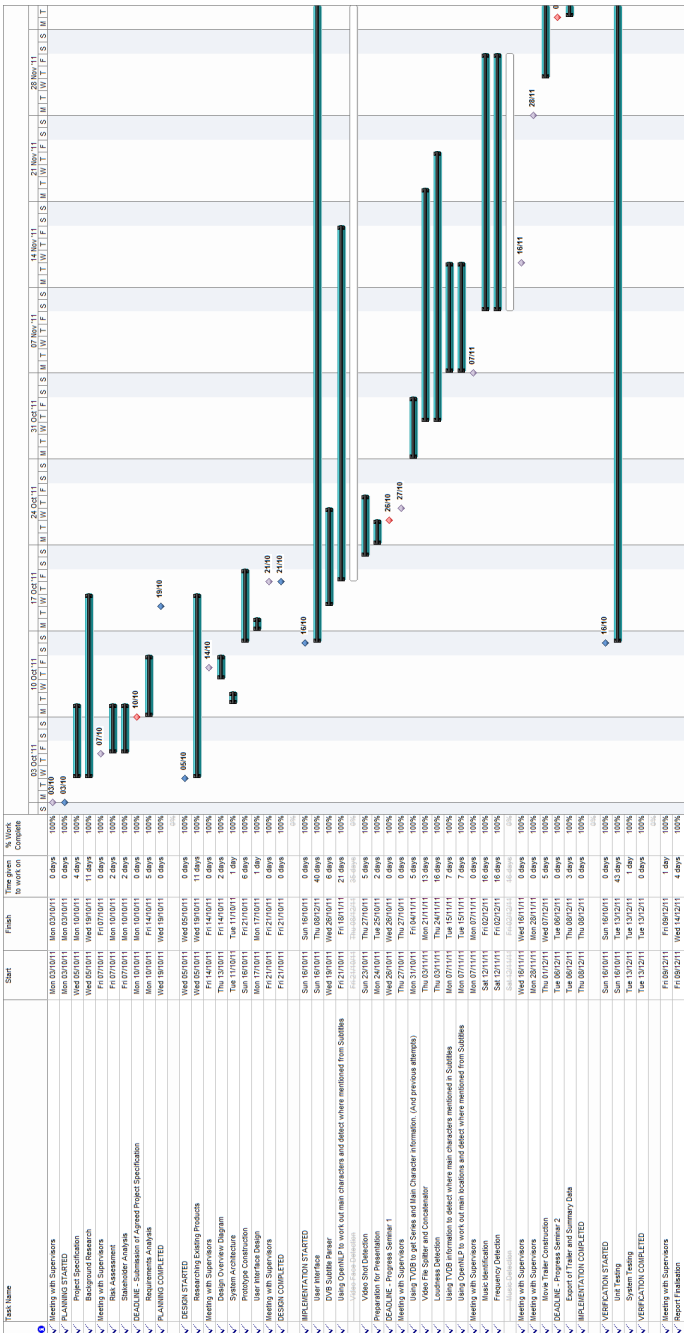

Figure 50: Final Gantt Chart

C Use Case Diagrams

This diagram shows the general functionality of our system where the primary actor is the Archiver.

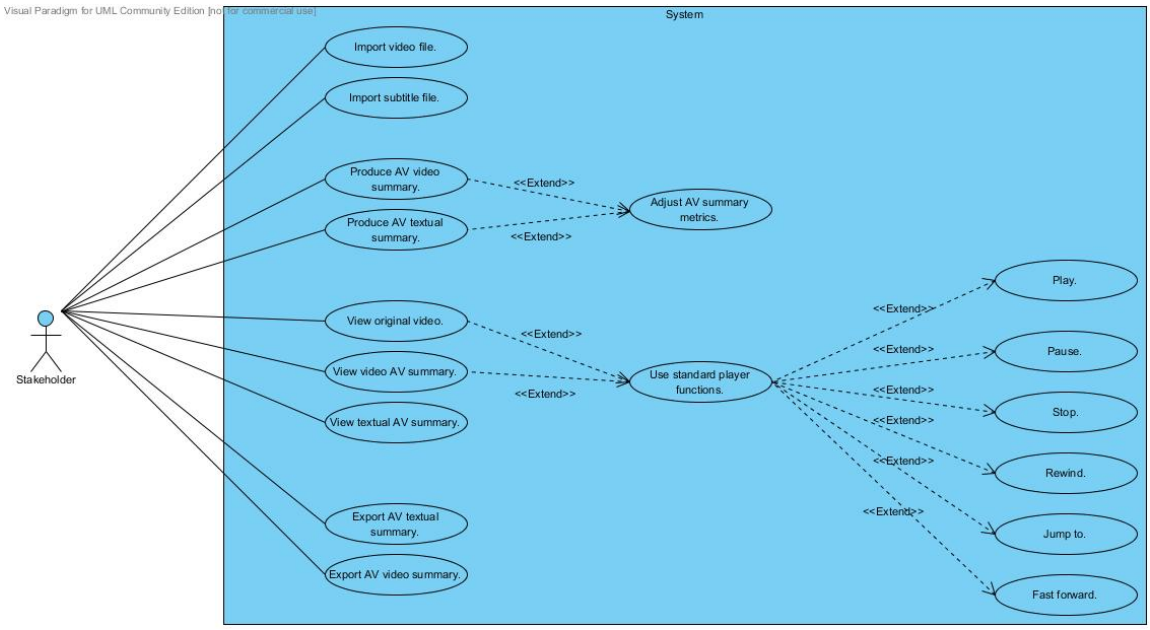

Figure 51: General Use Case Diagram

This diagram depicts the senarios of the Archive Monetisation, Rights Assessment and Archive Content Selection problems described in section 1.2:

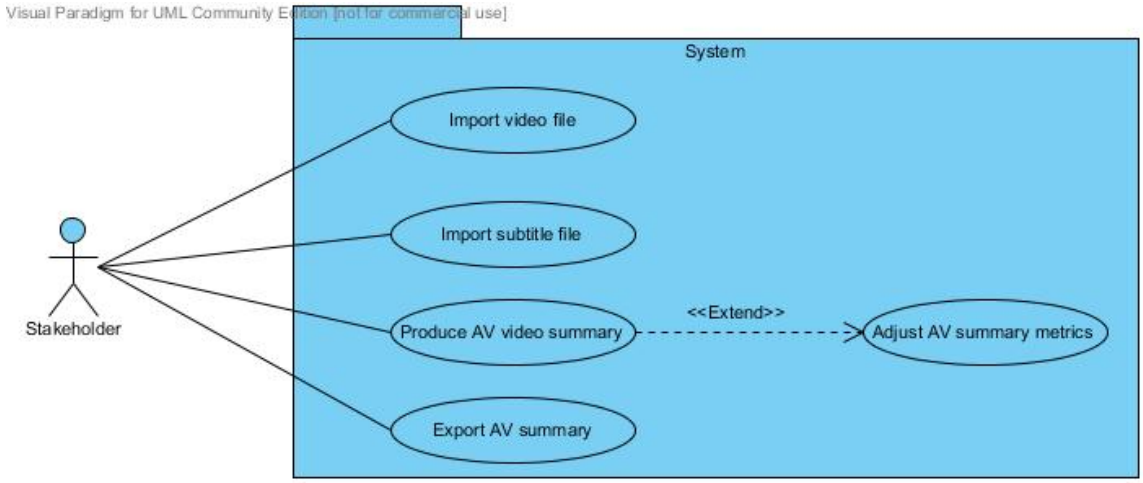

Figure 52: Archive Monetisation Diagram

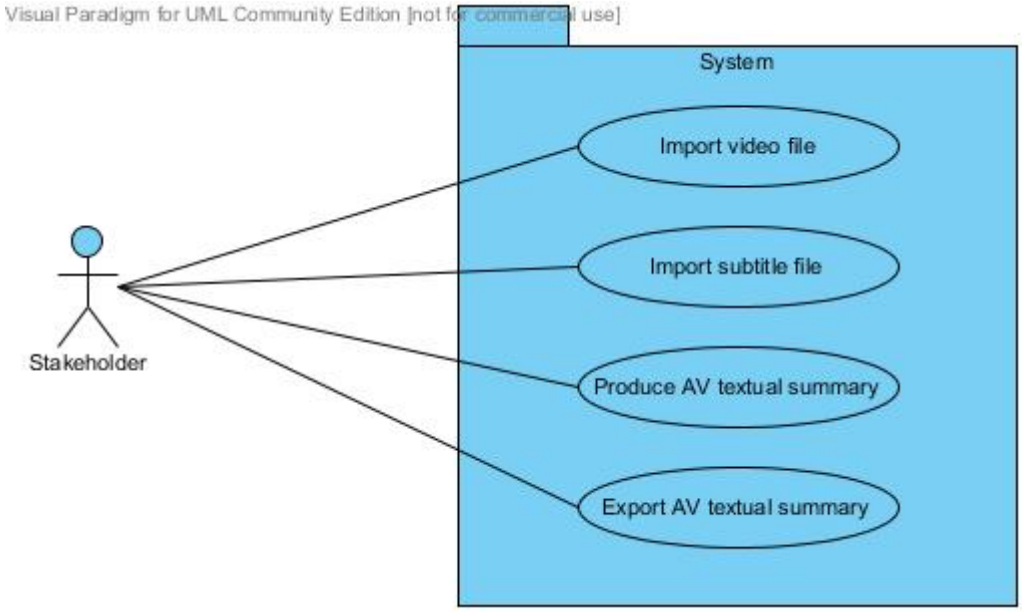

Figure 53: Rights Assessment Diagram

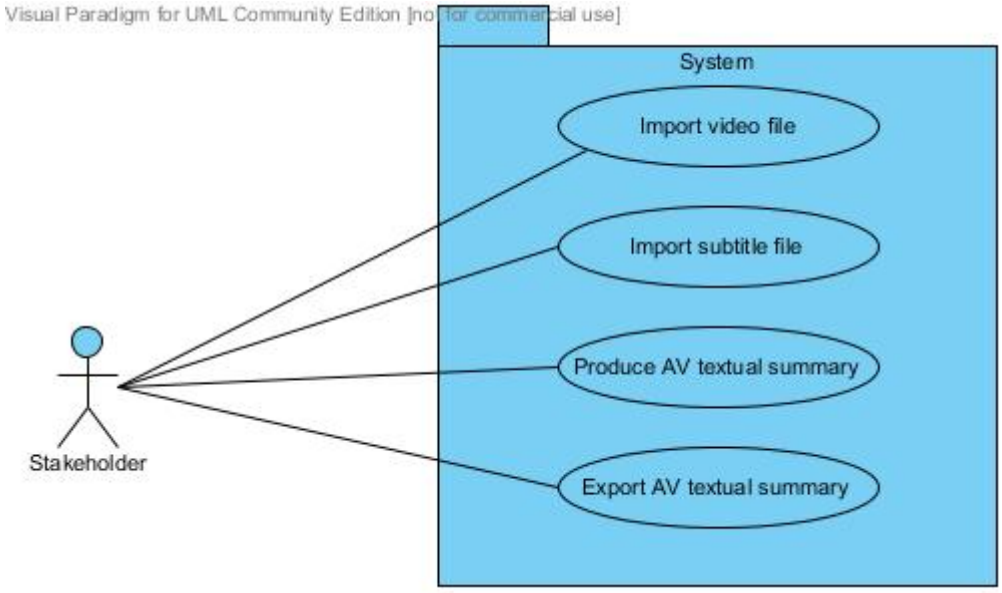

Figure 54: Archive Content Selection Diagram

D Testing Appendix

D.1 Unit Tests

This section details the rest of the unit tests.

Test	Test location	Test description	Expected result	Actual result	Success
11	model.threads. LocationName FinderThread	In the method run there is a while loop which should only iterate when the name finder process is currently operating	Should iterate through the while loop for whole time that the name finder process is active	Iterates whilst the aforementioned process is active	Y
12	model.tvdb. SeriesSearch	In the method chosenSeries the for loop should iterate the same number of times as there are genres for a series of TV chosen	If there are five genres the for loop should iterate five times	Iterates five times	Y
13	model.threads. PersonName FinderThread	In the method run there is a while check used to see whether a process is use, when in use this while loop should be entered	Process is in use so enter while loop	while loop is entered	Y
14	model.time. TimeUtils	In the method getTimeIn-String there is a for loop which should iterate twice	It should iterate twice	Iterates twice	Y
15	model.viedotools. GetSubsetAdapter	In the method start there is a for loop used to find the number of video streams in a file, the for loop should iterate the same number of times as there are valid video streams	Eight video streams should cause the for loop to iterate eight times	Iterates eight times	Y

16	view. PersonNames MentionedDetail Panel	In the method update there is an if check to see whether the UI should be updated	If the scroll-Pane does not equal null then it should be removed and the UI updated	ScrollPane is removed and the UI is updated	Y
17	view.videobrowser. VideoBrowser	In the method VideoBrowser there is a check to ensure that only accepted file formats are displayed in the file chooser	Should only display .mp4 files	Displays .mp4 only	Y
18	model.videotools. VideoFileHelper	In the method concatFiles there is an if check to see if any videos are present	If there are no videos then the if region should be entered	When no files were entered the if region was entered	Y
19	view. SeriesSearchPanel	In the method seriesSearch-Panel there is an if check to see whether the user has pressed the enter key	When the entere key is pressed the if region should be entered	Upon press-ing enter the if region is entere	Y
20	view.plot. ScatterPlotWindow	In the method addSeries there is a check to see whether a series is present	If a series is present then the if region should be entered	When there are no series present the if region is not entered	Y
21	model.music. MusicIdentifier	In the method identifyMusic does the if check only enter the area when all the variables are null & All variable equal null	Should enter the if section	If section en-tered	Y
22	model.music. MusicIdentifier	In the method identifyMusic we test to see whether the try and catch section work correctly when an exception occurs	When an ex-ception occurs a stack trace should appear	Stack trace appears once an exception is detected	Y

D.2 System Tests

Test	Test	Input	Expected Outcome	Actual Outcome
11	Load Video (Single file)	MP4 video file	Loaded Successfully	Y
12	Load Video (Any other file type)	AVI video file	Cannot be loaded into system	Y
13	Load Video (Unable to move focus to underlying GUI)		Unable to move focus	Y
14	Load Subtitles (Valid subtitle file)	Valid DVB subtitle file	Loaded Successfully	Y
15	Load Subtitles (Other XML file)	Non DVB XML file	UnSuccessfully loaded and output message	Y
16	Load Subtitles (Any other file type)	TXT file	Cannot be loaded into system	Y
17	Load Subtitles (Unable to move focus to underlying GUI)		Unable to move focus.	Y
18	Load Subtitles (Can't select multiple files)	Multiple DVB subtitle files	Can only select on file.	Y
19	TVDB Search (Performed when press return)	"Doctor Who"	Searched performed.	Y
20	TVDB Search (Performed when click Search)	"Doctor Who"	Searched performed.	Y
21	TVDB Search (No matching programme name)	"azazazaza"	Searched performed and exits as no results produced. Displaying message in GUI.	Y
22	TVDB Search (Results displayed and unable to move focus to underlying GUI)	"Doctor Who"	Search performed and results display. Unable to move focus to underlying window.	Y
23	TVDB Search (Results displayed and chose Cancel)	"Doctor Who"	Searched performed and cancelled succesfully. Displaying message in GUI.	Y
24	TVDB Search (Results displayed and chose Ok)	"Doctor Who"	Searched performed and completes Successfully. Displaying message in GUI.	Y

25	TVDB Search (Complete and then do search again)	"Two and a Half Men"	Searched performed and completes Successfully overwriting previous data. Displaying message in GUI.	Y
26	Select video (Another already selected)	Video file with subtitles	Update display to show video selected and deselect other. Processing for selected video begins. Subtitle processing displayed in output tabs.	Y
27	Select video (When a video is playing)	Two video files	Update display to show video selected and deselect other. Processing for selected video begins. And video continues playing.	Y
28	Play video (Non already playing)	Video file	Video begins playing in Player Tab from beginning.	Y
29	Play video (Another video is already playing)	Two video files	Video begins playing in Player Tab from beginning.	Y
30	Pause video	Video file	Video pauses correctly.	Y
31	Pause video (Paused previously)	Video file	Video resumes playing.	Y
32	Seek in video (Whilst video not playing)	Video file.	Nothing happens.	Y
33	Seek in video (Slider updates whilst video playing)	Video file.	Slider position updates as video plays	Y
34	Seek in video (Whilst video playing)	Video file.	Video jumps to position in video.	Y
35	Remove video (Whilst video is processing)	Video file	Video is removed from system. Processing which is running completes and doesn't cause issues.	Y
36	Remove video (Whilst video is processing)	Video file with Subtitles	Video is removed from system. Processing which is running completes and doesn't cause issues.	Y
37	View graph of video data.	Video file with Subtitles.	Graph is displayed showing all data produced for that video.	Y

38	Output tabs update correctly (Whilst viewing)	Video file with subtitles	Output Tabs updated Successfully whilst viewing.	Y
39	Generate summary (No videos loaded)		Unable to select Generate Summary	Y
40	Generate summary (Using Default Trailer Type)	Video file with Subtitles. Top Gear series selected	Summary generated Successfully.	Y
41	Generate summary (Using Loudness Trailer Type)	Video file with Subtitles. Top Gear series selected	Summary generated Successfully.	Y
42	Generate summary (Using People Mentioned Trailer Type)	Video file with Subtitles. Top Gear series selected	Summary generated Successfully.	Y
43	Generate summary (Using People Mentioned Trailer Type)	Video file without Subtitles. Top Gear series selected	Summary generated Successfully.	Y
44	Generate summary (Using Location Mentioned Trailer Type)	Video file with Subtitles. Top Gear series selected	Summary generated Successfully.	Y
45	Generate summary (Using Location Mentioned Trailer Type)	Video file without Subtitles. Top Gear series selected	Summary generated Successfully .	Y
46	Generate summary (One video without Subtitles, no TVDB information)	Video file without Subtitles. No TVDB information loaded.	Summary generated Successfully.	Y
47	Generate summary (One video without Subtitles, TVDB information)	Video file without Subtitles. Top Gear series selected	Summary generated Successfully.	Y
48	Generate summary (One video with subtitles, no TVDB information)	Video file with Subtitles. No TVDB information.	Summary generated Successfully.	Y
49	Generate summary (One video with subtitles, TVDB information)	Video file with Subtitles. Top Gear series selected	Summary generated Successfully.	Y
50	Generate summary (Multiple videos with mix of Subtitles files and none, no TVDB information)	Two Video files. One with subtitle and one without. No TVDB information loaded.	Summary generated Successfully.	Y
51	Generate summary (After previously selected a video)	Video file with subtitle file. TVDB information loaded.	Summary generated Successfully.	Y

52	Generate summary (Select video whilst processing)	Video file with subtitle file. TVDB information loaded.	Summary generated Successfully.	Y
53	Generate summary (Add video whilst processing)	Two Video file with subtitle files. TVDB information loaded.	Summary generated Successfully.	Y
54	Generate summary (Remove video whilst processing)	Two Video file with subtitle files. TVDB information loaded.	Summary generated Successfully.	Y
55	Generate summary (Change genre whilst processing)	One Video file with subtitle file. TVDB information loaded.	Summary generated Successfully.	Y
56	Generate summary (Change duration while processing)	One Video file with subtitle file. TVDB information loaded.	Summary generated Successfully.	Y
57	Generate summary (Change trailer type whilst processing)	One Video file with subtitle file. TVDB information loaded.	Summary generated Successfully.	Y
58	Generate summary (Change series whilst processing)	One Video file with subtitle file. TVDB information loaded.	Summary generated Successfully.	Y
59	Generate summary (Generate summary again after completion)	One Video file with subtitle file. TVDB information loaded.	Summary generated Successfully.	Y
60	Play summary (No summary produced)		Play Summary button not available.	Y
61	Play summary (Whilst generating summary)	One Video file with subtitle file. TVDB information loaded.	Play Summary button not available.	Y
62	Export summary (No summary produced)		Export Summary button not available.	Y
63	Export summary (Whilst generating summary)		Export Summary button not available.	Y
64	Export summary (One video without Subtitles, TVDB information)	Video file without Subtitles. Top Gear series selected	Export summary Successfully.	Y
65	Export summary (One video with subtitles, no TVDB information)	Video file with Subtitles. No TVDB information.	Export summary Successfully.	Y
66	Export summary (One video with subtitles, TVDB information)	Video file with Subtitles. Top Gear series selected	Export summary Successfully.	Y

67	Export summary (Multiple videos with mix of Subtitles files and none, no TVDB information)	Two Video files. One with subtitle and one without. No TVDB information loaded.	Export summary Successfully.	Y
68	Export summary (Multiple videos with mix of Subtitles files and none, TVDB information)	Two Video files. One with subtitle and one without. TVDB information loaded.	Export summary Successfully.	Y

E TV Testing Data

A number of different genre of BBC produced TV shows have been reviewed. Each show has had its volume levels through out the broadcast noted and any especially interesting points made a note of. These recorded volume levels have been used to test the accuracy of the systems audio detectors and the points of interest within a show have been used to see whether what the viewer finds interesting ends up within the systems summarisation.

Due to the nature of sound, audio volume is rather subjective, so the measurements used for these tests is purely a simple scale. The scale is from 0 - 10 , where 0 is silence and 10 is painfully loud, 5 the midpoint would be normal talking level and the other points are inbetween these levels. These volumes may be compared to those recorded by the system, for instance a point where the audio reaches what the viewer would call maximum or 10 we compare that time point to the one recorded by the system and see if that is the loudest the system recorded. A silent point recorded by the user as a 0 would be expected to be shown as very little or no sound by the system.

E.1 Top Gear Series 14 Episode 6 - Bolivia Special

Time (min:sec)	**Audio Description**	**Volume (0-10)**	**Interesting points**	**Matches Summary**
00:007 - 00:22	Intro music	9	N	
00:22 - 00:30	Clapping	8	N	
01:00 - 01:22	Shots of the jungle with music	7	Y	
02:08 - 02:12	Music	7	N	
03:18 - 03:23	Music	7	N	
03:39 - 04:00	Music	8	N	Y
04:36 - 04:50	Music	7	N	
05:16 - 05:20	Laughter	6	N	Y
05:52	Laughter	6	N	
07:08 - 07:19	Music	7	N	
07:38 - 07:45	Good shots	4	Y	
08:24 - 08:34	Clarkson sinking	6	N	
09:27	Jungle shots	2	N	Y
10:20 - 10:30	Engineering noises	8	N	
11:44	Crash	8	N	
12:40 - 13:00	Background noises	9	N	
13:30 - 13:38	Crunching	7	N	
13:40 - 13:50	Shouting	7	N	Y
14:10 - 14:30	Music	7	N	
15:00 - 15:50	Music	7	N	Y
16:28 - 17:17	Music	7	N	
17:17 - 17:19	Bees	5	N	
17:54 - 18:18	Music	7	N	
18:32 - 18:38	Music	7	N	
18:48 - 18:55	Screaming	7	N	
18:54 - 19:09	Screaming	7	N	
20:20 - 20:22	Sand	5	N	
20:24 - 20:36	Screaming	7	N	
21:01 - 21:30	Music	7	N	

21:32 - 21:54	Music	8	N	Y
21:56 - 21:57	Crash	8	N	
22:12 - 22:14	Sliding noises	7	N	
22:21	Laughter	7	N	
22:58 - 23:04	Talking	6	N	
23:05 - 23:14	Engine	8	N	
23:14 - 23:16	Sliding noises	7	N	
23:25 - 23:55	Engine	8	N	
24:00 - 24:10	Water noises	5	N	Y
24:45	Crunch	7	N	
24:47 - 25:00	Engine	8	N	
25:30 - 25:35	Engine	9	N	
25:50 - 26:03	Winches	7	N	
26:14 - 26:27	Chainsaw	9	N	
26:27 - 26:40	Music and Engineering noises	8	N	
27:20 - 27:26	Background noises	7	N	
27:40 - 28:03	Bridge view	5	Y	
28:15 - 28:41	Music	7	N	
28:46 - 29:16	Music	8	N	
29:20 - 29:40	Circular saw	8	N	
29:41 - 29:55	Screaming	7	N	
30:48 - 31:12	Music	7	N	
31:13 - 31:40	Rain	6	N	Y
31:41 - 32:10	Rain	7	N	
32:24	James	7	N	
32:50	Deep water	7	N	
33:00	Swimming car	7	N	
33:20 - 34:00	Water noises	7	N	
34:20 - 34:40	Shouting	7	N	
35:10 - 35:28	Driving	7	N	
35:30 - 35:47	Driving	7	N	
35:50 - 36:00	Driving and a rattling noise	8	N	
36:01 - 36:17	Music	7	N	
37:09 - 37:20	Total wipeout joke	6	N	Y
37:21 - 37:32	Music	7	N	
37:40 - 38:09	Music	7	N	
38:21	The most dangerous road	5	Y	
38:30	Shot of the dangerous road	5	Y	
39:21 - 39:23	Engineering noises	8	N	
39:50	Change	7	N	
40:01	Scary shot	7	Y	
40:16 - 40:43	Music	7	N	
40:45	Honking	8	N	
40:46 - 41:17	Driving	7	N	
41:19	Honkinh	8	N	
42:08 - 42:10	Machete moment	6	Y	
42:25 - 43:04	Music	7	N	
42:48	Honking	8	N	

43:19	Honking	8	N	
43:25	Honking	8	N	
43:31 - 44:00	Music	7	N	
44:15 - 44:40	Music	6	N	
44:50 - 45:10	Engineering noises	8	N	
45:17	Waterfall	5	Y	
46:01	Falling bricks	5	N	
46:53	No lights	5	N	
47:35 - 47:45	Dead collegue crosses	5	Y	
48:20 - 48:35	Music	7	N	
49:49 - 49:59	Fixing music	8	N	
50:02 - 50:10	Music	7	N	
50:30 - 50:36	Car jumper	6	N	
51:00 - 51:06	Shouting	7	N	
51:40	Laughter	7	N	
51:45	Transition noise	7	N	
51:56 - 52:05	Sky	4	Y	Y
52:26	Transition noise	7	N	
52:30 - 52:40	Music	7	N	
52:46	Caramelised cocaine	5	Y	
53:28	Mountain view	4	N	Y
53:54	Height above ground	5	N	
54:37	Transition noise	7	N	
55:10 - 56:05	Music	7	N	
55:28	Driving into a well	6	N	
55:41	James	6	N	
56:26 - 57:15	Music	7	N	
57:23	Transition noise	7	N	
57:25 - 57:30	Music	7	N	
57:55	Transition noise	7	N	
58:04 - 58:20	Music	7	N	
58:40	Volcano	4	Y	Y
58:38 - 59:00	Music and engine noises	8	N	
59:17	Talking	4	N	
59:34	Crunch	7	N	
59:39 - 59:43	Crunching	7	N	
60:00 - 60:20	Music	7	N	
61:40	Viagra jokes	6	Y	
61:50 - 62:35	Music	7	N	
63:43 - 64:13	Music	7	N	
65:00 - 65:38	Music	7	N	
65:38 - 65:50	Music	6	N	
66:00	“Air!”	5	N	
66:41 - 67:20	Music	7	N	
68:10 - 68:35	Music	7	N	
68:45 - 69:12	Music	7	N	
69:20 - 70:16	Music	7	N	Y
70:40 - 71:17	Music	8	N	
71:18 - 71:45	Music	7	N	

72:00 - 72:15	Crashing	8	N	
73:03	Sliding	8	N	
73:04 - 74:20	Music	7	N	Y
73:10	Nice view	5	Y	Y
74:25 - 74:36	Music	7	N	
75:46 - 75:56	Talking	7	N	
75:57 - end	Credits	8	N	

E.2 Top Gear Series 9 Episode 3 - US Special

Time (min:sec)	**Audio Description**	**Volume (0-10)**	**Interesting points**	**Matches Summary**
00:00 - 00:17	Intro music	8	N	
00:17 - 00:28	Clapping	9	N	
00:28 - 01:36	Talking	6	N	
01:36 - 02:06	Music	7	N	
02:22 - 02:28	Music	7	N	
03:37	Honking	9	N	
03:46 - 03:50	Honking	8	Y	
04:56	Background noises	7	N	Y
05:00 - 05:10	Background noises	8	N	
05:16 - 05:19	Honking	9	N	
05:45 - 06:23	Music	8	N	
06:24 - 07:00	Music	9	N	
07:01 - 07:24	Music	8	N	
07:25 - 07:30	Honking	8	N	
07:45 - 07:56	Background noises	7	N	
08:00	Accident	9	N	
08:48	Plane overhead	9	N	
08:57 - 09:00	Train passing	9	N	
09:01 - 09:12	Music	7	N	
09:31	Plane overhead	9	N	
09:32 - 09:52	Music	9	N	
09:59	Clunk!	8	N	Y
10:07	Clunk!	8	N	
10:47	Plane overhead	9	N	
10:57	Plane overhead	9	N	
11:04	Honking	8	N	
11:11 - 11:30	Music	8	N	
11:49	Train passing	9	N	
12:00	Plane overhead	9	N	
12:40	Train passing	9	N	
12:55	Snorting	7	N	
13:08 - 14:05	Music	7	N	
14:17 - 15:25	Music	7	N	
15:29 - 15:54	Music	7	N	Y
15:55 - 16:54	Music	7	N	
17:02 - 17:25	Music	7	Y	
18:00 - 18:03	Stig of America	7	Y	

18:40 - 18:47	Humerous fat jokes at the American Stig	7	Y	
19:27 - 19:28	Engineering noises	8	N	
19:34 - 20:22	Music	7	N	
20:43 - 20:47	Joke	6	Y	
21:12	Joke	7	Y	Y
21:53	Alligators	5	Y	
22:11	Hot dog drawing	6	N	
22:21	Being mean about Clarkson	5	N	
22:22 - 22:40	Loud braking	9	N	
22:58 - 23:14	Braking	9	N	
23:31 - 23:48	Music	7	N	
24:08	Driving	8	N	
24:41 - 24:50	Revving	8	Y	
25:14	Jumpstart	7	N	Y
25:20 - 25:27	Clarkson jokes	5	N	
25:20 - 25:40	Shouting	7	N	
25:45 - 25:55	Joking about Americans	5	N	
26:00	Colliding	7	N	
26:37 - 27:00	Music	7	N	
27:02	Radio call signs	6	Y	Y
27:24 - 27:27	Music	7	N	
28:15 - 28:40	Lorry driver and prostitute	6	Y	
28:58	Broken car	6	N	
29:36	Crash into James	8	N	
29:40 - 30:28	Music	7	N	
30:30 - 31:00	Music	7	N	Y
31:15 - 31:44	Music	7	N	
32:20 - 32:46	Laughter	7	N	
32:40	Car shower	8	N	
33:39 - 33:47	Music	7	N	
33:48 - 34:00	Shower car	7	N	
34:45 - 35:00	Roadkill	6	N	Y
35:10 - 35:20	Tortoise	6	N	
35:47	Vegetarian James	5	Y	
36:00 - 36:16	Background noises	6	N	
36:50	Crispy duck/squirrel	5	N	
37:37 - 37:47	Cow on the car	6	Y	
37:56	Cow car	7	N	
38:30	Laughter	7	N	
39:26 - 40:10	Music	7	N	
40:12 - 40:30	Music	8	N	
40:50	Cow smell	6	N	
41:02	Cow juices	6	N	
41:07 - 42:06	Music	4	N	
42:18	Baghdhad	5	N	
42:32	Hotter in hell sign	5	N	
43:20	Getting you killed	5	N	

42:47 - 43:47	Talking	6	Y	
44:02	Talking	6	N	
44:30 - 45:10	Music	7	N	
45:30	“Shot their own sign”	5	Y	
46:00 - 46:34	Squeeking	7	N	
46:40 - 47:05	Music	7	N	
47:35 - 48:00	Local hillbilly lady	5	N	
48:33	Red necks arrive	7	Y	
48:40 - 49:00	Rock throwing	7	Y	
49:40 - 50:00	Car cleaning	7	N	
50:12	“Could have killed us”	5	Y	
50:21	Jokes	5	N	
50:25 - 51:30	Thunderstorm	7	N	
51:30 - 51:54	Music	6	N	
51:55 - 52:19	Music	7	N	
52:25 - 52:36	Music	7	N	
52:37 - 54:08	Driving noises	7	N	
54:09 - 56:18	New orleans	5	N	Y
56:19 - 56:58	Music	7	N	Y
57:26 - 57:38	Clapping	8	N	
57:39 - 57:45	Laughter	8	N	
58:25	Laughter	8	N	
58:44	Laughter	8	N	
58:53	“Don’t go to America”	6	N	
59:00 - 59:08	Clapping	8	N	
59:08 - end	Credits	8	N	

E.3 Doctor Who (2005) Series 2 Episode 12 - Army of Ghosts

Time (min:sec)	Audio Description	Volume (0-10)	Interesting points	Matches Summary
00:14	Sad music	4	N	
00:36 - 00:40	The TARDIS noise	8	N	
01:00	Panaramic view of the beach	5	N	
01:20 - 02:00	Intro music	9	N	
01:58 - 02:04	The TARDIS noise	8	N	
02:15 - 02:34	Shouting	7	N	Y
03:34 - 04:00	Music	7	N	
03:47 - 03:54	Ghost appearance	8	Y	Y
04:02	Torchwood music	8	N	
04:17	Ghosts	7	Y	Y
05:00	Ghost watch team clapping	8	N	
05:27	Ghost watch news	6	N	
05:45	Ghost love	7	Y	
05:50	Fake ghost advert (Ecto shine)	6	N	
06:07	Japanese girls squeeling	8	Y	
06:27	Eastenders	7	N	
06:40 - 06:50	Ghosts everywhere	6	Y	
07:20	“They’re not ghosts”	6	N	
07:37	Loud excited talking	6	N	
08:20	The sphere	3	N	
08:33	Good shot of the spere	3	Y	
08:40	“Like it’s staring”	5	Y	
09:00	Sphere is repulsive	5	N	
10:50	Gareth shouting	7	N	
11:02	Worried voice	6	Y	
11:06	Shot through the dust screens	5	Y	Y
11:10	Music	7	N	
11:30	Shouting	7	N	
11:40	“Anyone gone through here..”	5	Y	
11:47	Screaming and machinary noises	9	Y	
11:58	Ghostbusters song	9	Y	Y
12:11	Triangulation	6	N	Y
12:30	“I think it’s horrific”	5	Y	Y
12:35 - 13:08	Science shouting	7	N	
13:20	Powering up	6	N	
13:30	Cyberman head	6	N	
13:40 - 14:00	Triangulation start	6	N	
13:54	Loud talking	6	N	
14:00	Rose is better	5	N	
14:28 - 14:45	Talking with her Mum	6	N	Y
14:51 - 14:58	Background noises	8	N	
15:00	“And into ghost shift”	7	Y	

15:15	See cyberman eyes and mechanical noises	8	Y	
15:20	Vies of ghosts through the monitors	5	Y	
15:41	Alarms	9	N	
16:05	"Offline"	7	N	
16:30	Locating the doctor	6	N	
16:50	"The dr."	7	Y	
16:55	"It's him"	7	Y	
17:03	Shouting	7	N	
17:17	"He's coming"	7	Y	
17:24	"It's him"	7	N	
17:45	Shouting	7	N	
18:00	Soliders	8	Y	Y
18:28	Safety catches off	8	Y	
18:42	Clapping	8	N	Y
18:51	Hooray	8	N	Y
19:32	Joke	5	N	
20:00	Shouting	7	N	
20:20	Music	9	N	
21:00	Torchwood	6	N	
22:05	Side effects	6	Y	
22:12	TARDIS taken	7	N	
22:22 - 22:34	Rose peeks out the TARDIS	8	N	
23:20	"Doctor Who enemy of Britain"	7	N	
23:40	"You're a prisioner"	7	N	
23:57 - 24:04	The sphere	8	N	
24:18	Talking	5	N	
24:50 - 25:11	Explanation of a void ship	6	N	
25:14 - 25:40	Space between dimensions	6	N	
26:00	Talking	7	N	
26:24	Shouting	8	N	
26:30	Chainsaws and screaming	9	N	
28:00	Hole in the world	6	N	
28:30	Canary wharf view	5	Y	
29:30 - 29:43	World splintered and glass shattering	6	N	
30:24 - 20:35	Ghost shift countdown	8	N	
31:03	Creepy music	5	N	
31:21	Background noises	9	N	
32:20	Music	8	N	
32:37	Talking	6	N	
33:08	Talking	6	N	
33:16	Mickey's back	5	Y	
34:15	Shouting	8	N	
35:00	Sphere activates	6	N	
35:16	Earpiece controls them	5	N	

35:40	It's in their brain	5	N	
36:00	Sphere is active	6	N	
36:20 - 36:50	Shouting	9	N	
37:15	The cyberman arrive	8	N	
37:27	Advanced gaurd	6	N	Y
37:36	Shooting	9	N	
37:40	Mickey	7	N	
38:05	Shooting the scientists	9	N	
38:31 - 38:51	100% Ghostwatch	9	N	
39:00	Cybermen come through	9	N	
39:11	Screaming	9	N	
39:30 - 40:00	Talking	6	N	
40:02 - 40:50	Screaming	9	N	
40:37	Victory	6	N	
40:48	Sphere	6	N	
40:58	Sphere opens	7	N	
41:11	Mickey	8	N	Y
41:28	Music	9	N	
41:46	Daleks	8	N	
41:53	Exterminate	9	N	
41:56 - end	Credits	8	N	

E.4 Doctor Who (2005) Series 2 Episode 13 - Doomsday

Time (min:sec)	Audio Description	Volume (0-10)	Interesting points	Matches Summary
00:10 - 00:19	"My name is Rose.."	5	N	
00:32	Cybermen	5	N	
00:36	Daleks	5	N	
00:39	".. the story of how I died"	5	N	
00:44 - 01:20	Intro music	8	N	
01:20 -01:24	Exterminate	8	N	
01:55	"You'll be necessary"	8	N	
02:04	"Commence awakening"	8	N	
02:40 - 03:24	Cybermen broadcast	8	Y	
03:25 - 03:45	Rioting	9	Y	
03:36	Explosions	9	N	
03:49	Exploded houses	8	N	
03:55	Canary wharf on fire	8	Y	
04:20 - 04:57	Screaming	8	N	
05:00	Cybermen investigate	6	N	
05:10 - 05:21	Cybermen talking	6	N	
05:30 - 05:57	Stomping	7	N	
06:08 - 06:35	Identify	7	Y	Y
07:00 - 07:50	"Upgrade the universe"	7	N	Y
07:52 - 08:10	Daleks declare war on Cybermen	7	N	Y
08:20	Dalek talking	7	Y	
09:00 - 09:25	Shooting	9	N	
09:40 - 10:28	Screaming and chainsaws	9	N	
10:40 - 11:00	Shooting	9	N	
11:20 - 11:37	Jackie escapes	7	N	
12:20	Shouting	7	N	
12:38	Talking	6	N	
14:00 - 14:20	Dalek talking	7	N	Y
15:45 - 16:00	Shouting	7	N	
16:40	Shouting	7	N	
16:50 - 17:22	Shouting	7	N	
17:58	Handprint	7	Y	Y
18:40 - 19:00	Noises	8	N	
19:12	Dalek talking	7	N	
20:40 -21:20	Noisy	8	N	
21:20 - 21:30	Explosions	9	N	
21:30 - 22:10	Rose's dad arrives	8	N	Y
22:00	Genesis gets touched	8	N	Y
22:10 - 22:30	Daleks	7	N	
22:40 - 23:00	Music and stomping	8	N	
23:12	"Pete"	7	N	
24:20	Jokes	6	N	
25:13	Hugs	6	N	
25:20 - 25:37	Exterminate and shooting	9	N	
26:00 - 26:18	Stomping	8	N	

26:19 - 26:50	Shouting	8	N	
27:01	“Elevate”	7	N	
27:29	Nice view of the Genesis ark	5	Y	
27:50	Daleks come out of the ark	5	Y	
28:02	“It’s a prision ship!”	6	N	
28:20 - 28:24	Stomping	8	N	
28:31 - 28:38	Daleks flying	7	N	
29:24	“I can see”	7	N	
29:44	“Void stuff”	7	N	
30:34	“Pete’s world”	7	N	
30:37	Goodbyes	6	N	
31:45 - 32:10	Sad Rose	6	N	
32:20 - 33:00	Shouting	7	N	
33:10	Music	7	N	
33:30 - 33:45	Cybermen shooting	8	N	
34:00	“Exterminate”	7	N	
34:20	Portal	9	N	
34:48	Aliens getting sucked back through	9	N	
35:39	Lever struggle	9	N	
36:19	Lets go	9	N	
36:25	Shouting	9	N	
36:33	Rose is caught	9	N	Y
36:47	System closed	9	N	
36:50 - 38:45	Sad music	5	N	
38:45	Calling Rose	5	N	
39:00 - 40:27	Sad music	6	N	
41:00	Sad music	6	N	
41:50	Crying	7	N	
43:45	“I love you”	7	N	
44:12	“Rose Tyler I ..”	6	N	
44:40	Music crescendo	8	N	
45:30 - 45:42	“What!”	8	N	
45:42 - end	Credits	8	N	

F CD Contents

- source.zip
- Top Gear America Summary
  - top-gear-usa-summary.mp4
  - top-gear-usa-person-mentioned-type-summary.mp4
  - top-gear-usa-graph.png
- Top Gear Bolivia Summary
  - top-gear-bolivia-summary-10-min.mp4
  - top-gear-bolivia-summary-5-min.mp4
  - top-gear-bolivia-graph.png
  - data_xml.xml
  - summary_xml.xml
- Doctor Who Episodes 1 - 3 Summary
  - doctor-who-summary-ep1-3.mp4
  - doctor-who-ep1-graph.png
  - doctor-who-ep2-graph.png
  - doctor-who-ep3-graph.png