Automatic Camerawork in virtual talk show production

Author: Mikael Lang
Company: NHK Japan Broadcasting Corporation
Supervisor at NHK: Dr. Masaki Hayashi
Supervisor at KTH: Mr. Björn Hedin
Examiner: Prof. Nils Enlund at NADA/Media, KTH
Sammanfattning

Automatiserat kameraarbete i virtuella talkshowproduktioner

NHK Japan Broadcasting Corporation har utvecklat ett script-språk kallat TVML (TV program Making Language) för att kunna producera kompletta TV program på PC. TVML är ett textbaserat språk som beskriver TV-program. Ett PC-program kallat TVML Player tolkar TVML-scripten och översätter dem till TV-program medelst realtidsdatorgrafik, syntetiserade röster och andra multimediafunktioner.

Idén med TVML är att slutanvändaren har en TVML Player installerad i sin PC, eller i framtiden i sin TV. Tanken är då att TV-stationen endast sänder ett TVML-script till slutanvändaren och att TV-programmet genereras på slutanvändarens visuella enhet.

NHK vill använda TVML för att automatiskt i realtid generera TV-program från dialogscript endast innehållande information om vad karaktärerna i TV-programmet säger.

Mitt arbete har varit att utveckla ett nytt system som från dialogscript automatiskt genererar kameraarbete för TV ”talk shows”. Systemet använder en kunskapsbaserad algoritm för att bestämma vad kamerorna skall inkludera och tillåter användarinteraktion och multipla karaktärer. Programvaran har testats på forskare på NHK och resultaten har varit lovande.

Abstract

NHK Japan Broadcasting Corporation has developed a language named TVML (TV program Making Language) for making complete TV programs on PCs. It is a text-based language to describe TV programs. A PC-program called the TVML-player interprets the TVML-script and transforms it to a TV-show in real time with real-time computer graphics, synthesized voices and other multimedia functions.

The idea with TVML is that the user has a TVML player installed in his/her computer or in the future in the TV set. The broadcaster then only sends a TVML script and the TV show is generated in the viewer’s viewing unit using the TVML player.

NHK want to use TVML for automatic TV program production. The idea is to have speech lines as input and transform them to TV, automatically in real time.

My work has been to develop a software that automatically generates camera work for TV talk shows. The software uses a knowledge-based algorithm to decide what the cameras will include and allows multiple characters and user interaction. The performance of my software was tested on research engineers at NHK and the result was quite good.
Preface

The realization of my master’s thesis would not have been possible without the help from Professor Funakebo, the grants from Sweden Japan Foundation, the grants from Japan Precision Measurement Technology Foundation, my great supervisor at NHK Dr. Hayashi and all my good fellow-workers at NHK. I would herby like to take the opportunity to thank you all for giving me the opportunity to write my master’s thesis at NHK and for giving me one of the best six month in my life. I would also like to thank the Sunaga family for helping and supporting me under my six month in Japan.
Table of content

1. Introduction .................................................................................................................. 1
   1.1. Structure of this report .......................................................................................... 1
   1.2. Introduction .......................................................................................................... 1
2. Method ............................................................................................................................. 2
   2.1. Problem definition ............................................................................................... 2
   2.2. Goal definition ...................................................................................................... 2
   2.3. Procedure ............................................................................................................. 2
3. Background ..................................................................................................................... 5
      3.1.2. The TVML Player [1] .................................................................................. 8
      3.1.3. Usage of TVML today .................................................................................. 8
      3.1.4. The TVML Player’s external control mode [2] ............................................ 9
   3.2. Introduction of automatic camera-works ............................................................... 9
      3.2.1. The idea/purpose for automatic TV program production ......................... 10
   3.3. Previous work in automatic camerawork ............................................................ 10
      3.3.1. Automatic generation of talk show from dialog using TVML [4] ............. 10
      3.3.2. The system .................................................................................................... 10
      3.3.3. Problems with this algorithm ...................................................................... 13
      3.3.4. System for automatic TV program generation using dialog transcription [7]. 13
      3.3.5. The system .................................................................................................... 13
      3.3.6. Choice of camera focus .............................................................................. 13
      3.3.7. Decision when to switch camera ................................................................. 14
4. Development .................................................................................................................. 16
   4.1. Ideas behind the new system ................................................................................. 16
      4.1.1. How is a real TV show produced ................................................................. 16
      4.1.2. How can a real producer be simulated ....................................................... 16
      4.1.3. Differences from previous systems .............................................................. 17
   4.2. The system basics .................................................................................................. 17
      4.2.1. Character and camera definition ................................................................. 17
      4.2.2. System introduction .................................................................................... 19
   4.3. Layer 1: The decision on when to switch shot ...................................................... 20
      4.3.1. Idea behind the algorithm .......................................................................... 20
      4.3.2. Camera switch by transition probability ..................................................... 21
      4.3.3. Camera switch by duration probability ....................................................... 22
   4.4. Layer 2: The decision which shot to use ............................................................... 22
      4.4.1. Idea behind the algorithm .......................................................................... 22
      4.4.2. Deciding who is a dominant speaker .......................................................... 23
      4.4.3. Comparing characters ................................................................................. 23
      4.4.4. Functions developed ................................................................................... 24
      4.4.5. Algorithm ..................................................................................................... 24
      4.4.6. Camera switch when speaker is not in shot .............................................. 25
      4.4.7. Camera switch when the character already is in the shot, the shot is a close shot and the reason for the switch is duration .................................................. 26
      4.4.8. Camera switch when the character already is in the shot, the shot is a multiple shot and the reason for the switch is duration ........................................... 26
4.4.9. Character already in the shot, the shot is a close shot and the reason for the switch is transition.................................................................27
4.4.10. Character already in the shot, the shot is a multiple shot and the reason for the switch is transition.................................................................27
4.4.11. Special situation when the program starts........................................28
4.5. The decision which camera to use ......................................................28
4.6. Gesture generation ...........................................................................29
4.7. Startup data.....................................................................................30
  4.7.1. Placing the characters.................................................................30
4.8. Using the system................................................................................31
4.9. Advanced settings.............................................................................34
4.10. Calibrating the system......................................................................35
  4.10.1. The interviews............................................................................35
  4.10.2. Changes made during the interviews...........................................35
4.11. Testing the system...........................................................................37
  4.11.1. The interviews............................................................................37
  4.11.2. Results.......................................................................................37
5. Summary..............................................................................................39
6. Conclusion ...........................................................................................40
7. List of literature....................................................................................41
8. Enclosures ...........................................................................................42
1. Introduction

The introduction starts with the structure of this thesis followed by a short introduction on the subject.

1.1. Structure of this report

This thesis starts with a short introduction followed by the problem definition, goal definition, and procedure. Thereafter comes an introduction of the TVML technology, followed by two short introductions on two by NHK previously developed systems for automatic TV program production. Then the main part begins, which is the description of the system that I have developed followed by the test results, a summary and the conclusion.

1.2. Introduction

NHK has developed a language named TVML (TV program Making Language) for making complete TV programs on PCs. It is a text-based language to describe TV programs. A PC-program called the TVML Player interprets the TVML-script and transforms it to a TV-show in real time with real-time computer graphics, synthesized voices and other multimedia functions. The idea with TVML is that the user has a TVML Player installed in his/her computer or in the future in his/her TV set. The broadcaster then only sends a TVML-script and the TV show is generated in the viewer’s viewing unit using the TVML Player.

Picture 1: A TVML script going in to the TVML Player and coming out as a TV show.

There is a mode in the TVML player called external control mode, when this mode is used it, is possible to control the TVML Player in real time with an external program, while the TV program is created. NHK have done some research on how one can use this mode for automatic TV program production. Automatic TV program production involves gesture generation and automatic camera switching generation. My work has been to develop a software that generates automatic camera work for TV talk shows, allowing multiple characters and user interaction. My system also uses a knowledge-based algorithm to decide which characters the camera will include in the picture.
2. Method

The method part starts with the problem definition followed by the goal definition and the procedure.

2.1. Problem definition

Previous systems developed by NHK use random values in combination with statistical data to decide what the camera will focus on. It means that there is no awareness from the software on what is going on in the show. It would therefore be interesting to try to develop a system that has a direct connection between the choice of camera shot and what is going on in the TV show.

NHK wants the user to be able to adjust the camera clipping frequency in real time. This would make it possible for older people or people with handicaps that today complain about the MTV like clipping style in many TV shows, to adjust the camera switching frequency in the TV shows as they are watching it.

There is also a wish to have a more flexible system, which allows multiple characters.

My work will be to develop a software application based on TVML that will convert text to TV. The application will have camera work generation but no gesture generation. The software will include the following features:

1. Number of people (making it possible to have one to infinity many characters.). TVML allows a maximum of 128 characters. However, today’s computer power limits the amount of characters to 10-15.
2. User interaction, to make it possible for the user to affect the camera clipping frequency in real time.
3. Using knowledge about the speaker as a basis for decisions. The two previous systems have used random values combined with statistical data to decide what will be included in a camera shots. I will try do base that decision on data that my software can extract from the text script.

2.2. Goal definition

The goal is to develop a software that converts text to TV talk shows automatically, in real time. The functionality that NHK wants me to include is:

1. The system should be able to handle multiple characters.
2. The user should be able to change the camera switching frequency manually in real-time.
3. The decision on what to include in each shot shall be based on data from the TV show while it is running.

My work will concentrate on the automatic generation of camerawork and it will not handle the generation of gestures.

2.3. Procedure

To construct a new software environment is a quite complicated process. I will here under describe my approach and why I have chosen this approach.
First step was to analyze previous work done in the field of automatic TV program production. I analyzed two research projects made by NHK. The first one was done in 1999 by Dr. Hayashi [4] and the second one made by Ariyasu-san [7] was released 2,5 month after I started with my project.

After analyzing Dr. Hayashi’s work and interviewing the people working with Ariyasu-san’s project, I started to plan my work and draw up some guidelines for how my software should work according to the goals set.

I divided the development process in two steps. In the first step I developed the new software environment in such a way that it satisfied goal one and two. I did that because I thought that I would learn the most about interaction with the TVML Player by starting as soon as possible with the development, and I couldn’t start with the development of the camera switching algorithm before I had constructed the software environment, because then I didn’t know which parameters that I could use for the camera switching decisions. When I developed the software environment I made sure that it collected all data that I thought could be needed for the camera switching algorithm.

The second step was to develop the camera switching algorithm and to do that I needed knowledge about how the choice of camera shot is conducted in a real TV show. I had access to an analysis of 30 hours debate shows from Ariyasu-san’s [7] project. After discussing with her and doing some basic statistical investigation, I came to the conclusion that it would be very hard to extract any data of relevance from these shows. The reason was that however I tried I could not find any tendencies that could be used as rules. The data contained too many parameters and combination of parameters. Neither did I have the possibility to analyze the TV shows by myself, because they were all in Japanese and it would have been very hard for me to get a proper understanding of them. To collect material of my own was also out of question because that it would take too long time to gather and analyze the data. Me and my supervisor came to the conclusion that there might not be any perfect TV programs, there might be a lot of camera shots that are made because that there was no better possibility for the moment, because the cameramen were already in a certain position etc. Each TV show also gets affected by the producer’s personal style, which makes it hard to draw some general rules. I therefore chose a passive approach, I would try to make up rules based on the information that the TVML player have access to and then try them to see if they would generate a show that looked natural or not.

When I thought that the shows generated by my software looked good, I started my calibration step where I interviewed six TV technology researchers about their thoughts about the shows generated and the software’s functionality. After each interview, I remodeled my software to match the comments from the interview targets.

After the calibration session, I started the testing session to get an opinion on the software. I let six other TV technology researchers watch a show produced with my software and then I asked them about their impression about and thoughts of the show.

The reason why I chose this method as a confirmation that my system works, is that the TVML Player itself have some imperfection and it would be hard for an non-professional person to see which errors are generated from my software and which errors are generated from the TVML Player it self. When I started with the development, I could only go out from
what I knew and what I wanted to achieve. The calibration step then worked as a fast method, to get rid of the most obvious errors and to calibrate the hard coded values in my software. The second interview part worked as a confirmation of my work and a judgement of my software, giving guidelines for further research projects.
3. **Background**

This chapter gives a brief introduction on the TVML technology and two previously developed systems for automatic TV program production.

3.1. **Introduction of TVML [6]**

To generate a TV program with a computer requires some kind of intermediate expressions that the TV-program producer can give to the computer to indicate tasks like directing the performance of the actor or to direct the lights etc. TVML is a text-based language designed to do just that. The TVML language is easy to understand for humans and it instructs the computer what to do. This is achieved by using highly abstracted text like “zoom-in” or “talk”. Video, audio and hypertext are then automatically generated from the TVML-script by a software called the TVML Player.

```
character: talk( name= BOB, text= HI)
character: talk( name= MARY, text= Hello)
camera: closeup( name=A, what=BOB)
```

*Picture 2: The flow of TVML. The picture shows how a TVML script gets transformed to a TV show.*

The TVML language fits best to produce TV programs that have a standard format and is some kind of information presentation show like news, weather news, presentation of documentaries etc. TV shows like dramas etc is not a target for TVML. In table 1, you can see a brief description of the different technologies used to transform a TVML-script to a TV-show.
To give an example of how the transformation of a TVML-script to TV-show works, will I here describe how a “studio presentation of a movie with one host” would work.

The studio shots of the anchor is CG (Computer Graphics), the anchors voice is text synthesized to speech and a CG setup camera captures it all. The movie is a movie file that is played back. Titles and captions are generated by using the layout description of HTML to display text information. The audio is produced by playing back an audio file or as above mentioned “text to speech synthesize”. To do this we need two types of information, script data written in TVML and various forms of reference data. The division of reference data and script data can be seen here under.

**Reference data**
- CG characters
- Voices used by the voice synthesizer
- The studio environment (background, tables, chairs etc)
- Lights
- Cameras

**Script data**
- The utterances from the characters
- Directions to which reference data to use

### 3.1.1. The TVML Language [6]

The specifications for the TVML language has been developed with reference to the structure of program production scripts used in real TV program productions. The TVML language is event driven and consist of different event classes (see table 2) that have commands that requires parameters. For example to make the character “Mike” say “Hi mate” you would write:

\[
\text{Character:talk(name=Mike, text="Hi mate." )}
\]

In this case the character is the event type and talk the command and the items inside the parentheses are the parameters. So the standard format of an event is:
**Event type: Command name(parameters…..)**

In table 2 you can see some examples of event types and their commands.

<table>
<thead>
<tr>
<th>Function</th>
<th>Event type</th>
<th>Examples of commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG character</td>
<td>character:</td>
<td>talk(...), walk(...), look(...), sit(...), etc...</td>
</tr>
<tr>
<td>CG camera</td>
<td>camera:</td>
<td>closeup(...), twoshot(...), etc...</td>
</tr>
<tr>
<td>CG studioset</td>
<td>set:</td>
<td>openmodel(...), change(...), etc...</td>
</tr>
<tr>
<td>CG prop</td>
<td>prop:</td>
<td>position(...), openimageplate(...), etc...</td>
</tr>
<tr>
<td>CG lighting</td>
<td>light:</td>
<td>model(...), switch(...), etc...</td>
</tr>
<tr>
<td>Motionpicture</td>
<td>movie:</td>
<td>play(...), etc...</td>
</tr>
<tr>
<td>Title</td>
<td>title:</td>
<td>display(....), etc...</td>
</tr>
<tr>
<td>Superimposing</td>
<td>super:</td>
<td>on(....), etc...</td>
</tr>
<tr>
<td>Sound</td>
<td>sound:</td>
<td>play(...), etc...</td>
</tr>
<tr>
<td>Naration</td>
<td>neration:</td>
<td>talk(...), etc...</td>
</tr>
</tbody>
</table>

Table 2: The table shows examples of different TVML functions, event types and commands.

Each event command has several parameters, the user sets as many as he or she wishes. The parameters that the user doesn’t set will use preset values. It all depends on how much control the user desires over the event. For example, if the user wants the character “MIKE” to say “What are you doing BOB” in a certain speed and with an exaggerated character gesture, the event description would be written as follows.

**Character: talk (name="MIKE", text=" What are you doing BOB", rate=5.0, emotion=exite)**

When the TVML Player processes a TVML script, it processes one line at the time and does not go on to the next line until the previous event has been executed. However, there is a possibility to define events in time. There are two types of events, action-events that takes some time to perform for example for a character to sit (the time will then be the time that it takes for the character to go from the state that he/she is in to go to the state sit). The other one is state-events that simply specify state change like superimposing a text. To make it possible to have action events occur simultaneously, there is a parameter “wait” that can be set to “yes” or “no”. If it is set to “yes”, the action event waits until the next action event occurs and they will then take action at the same time.

**Character: bow (name=BOB, wait=yes)**
**Character: bow (name=MARY)**

In this case BOB and MARY will bow simultaneously, but without the parameter “wait” set to “yes”, MARY would wait to bow until BOB was finished bowing. There is also a method to define absolute time values in TVML. For example if the user wanted to start the playback of a motion picture file from frame 100 to frame 200 and superimpose text after 1.5 seconds it would look like this.

**Movie:playfile(filename=test.mov, from=100, to=200, wait=no)**
**Wait(time=1.5)**
**Super: on(type=text, text="This is a text movie.")**
**Movie:wait_playfile()**
This was a brief introduction about the TVML language to learn more about the TVML language please see Enclosure 1

3.1.2. The TVML Player [1]

The TVML Player is software that reads and converts TVML script to video and audio in real-time. The hardware platform for the TVML player is a Windows PC or any graphic workstation from SGI. In a Windows PC, a Microsoft speech API is used for voice synthetication. The mouth of the character is opened in direct proportion to the magnitude of the voice level to achieve lip-syncing. SGI workstations require an external voice-synthesizer (hardware) that is attached to the serial port of the machine. The TVML Player supports AVI movie files, Quick-Time movie files and SGI movie files, it also supports WAV and AIFF audio files, and TIFF files for still pictures. It also supports OpenInventor and VRML 1.0 for the modeling data format in computer-generated characters, sets, and props. The TVML Player features a straightforward user interface with buttons for playback, stop, pause etc., enabling selections of TVML script files and immediate playback. Se Enclosure 2 for a complete TVML script.

![TVML Player](image)

Picture 3: The picture shows an example of how a TVML show looks.

3.1.3. Usage of TVML today

TVML itself has so far had very limited usage. The reason is that NHK started the development of the TVML with some private companies. TVML is sold but is quite expensive and has mostly been used in universities for teaching purposes. There are however plans from NHK to make TVML an open source software, to allow further development. NHK’s broadcasting unit is planning to use TVML to produce broadcasting material in a near future. NHK have also developed a new system called “TV4U” which uses parts from the TVML technology. TV4U converts web information to HDTV shows in real time.
3.1.4. The TVML Player’s external control mode [2]

The basic operations of the TVML Player is to playback a TVML script and convert it to video, audio and hyper-text, a pure interpretation job. Read one line, syntactically parse it and execute it, then wait with the next line until the event is over. To allow TV programs with interaction the TVML Player is equipped with an external control mode, which allows external computer programs to control the TVML Player. This means that it is possible to insert any TVML-command while the TVML-Player is playing a script. If you boot the TVML Player in “external control mode”, a shared memory is created that both the TVML Player and the external application will have access to. The shared memory works as the communication link between the two softwares. The external software can then at any time send a script line to the TVML Player, and that script line will be executed as soon the last event is over. The external application can control everything about the TVML Player including the GUI (Graphic User Interface) and it can also inform itself about the status of the TVML Player by demanding status flags.

![Diagram of TVML Player in external control mode](image)

*Picture 4: The picture shows how the TVML Player communicates with external applications.*

3.2. Introduction of automatic camera-works

NHK has done some research about how one can use TVML as platform for automatic TV program production. The TVML player provides an external control function enabling external computer programs to control the TVML Player in real-time. This makes it possible to change the TV program in real-time (while the TV program is generated in the TVML Player). There are several external programs involved in the automatic TV program production for example the “Camera Switching Generator” and the “Gesture Generator”. The “Camera Switching Generator” manages all the camera switching and the “Gesture Generator” manages all the gesture generation of the characters, all done in real-time. While using the TVML player and the external programs for automatic TV program production, the only thing the TV program producer has to do is to define the characters, the computer graphics environment, the cameras and the dialog between the characters. All the gestures and camera switching is taken care of by the external programs in real-time.
3.2.1. The idea/purpose for automatic TV program production

The idea with the automatic TV program production is that it can be used to simply convert text to TV. It could for example be used to convert web pages to TV shows, or internet chat sites to TV shows. In the future an automatic TV program production system could be integrated in the TV-set and be used to present local weather news and other personally defined contents. The gain is that it limits the workload that the broadcasting station has to put in to a TV production.

3.3. Previous work in automatic camerawork

Using TVML, it is relatively easy to produce a scene in which computer generated characters are talking on the basis of input dialog. But to make a TV show from these dialogs demands gesture generation and camera work. There have been two research projects at NHK about automatic TV program production both taking a similar approach but using different data to base their algorithms on. I will here give a brief introduction on both of them.

3.3.1. Automatic generation of talk show from dialog using TVML [4]

This was the first research project that NHK conducted on automatic camera work production and it was mainly done by Dr.Hayashi. The system was designed for a two person TV show and the statistical data was taken from a Japanese famous talk show called “Tetsuko’s Room”. The concept behind the system is to use statistical data from a real TV show to decide when it’s going to be a camera switch and what the camera will focus on. The decisions are then made by using statistical data combined with the random number generator of the computer.

3.3.2. The system

The flow of this system is shown in picture 6. First the dialog with the Host and Guest is loaded in to the software. Thereafter the length of each speaking interval is calculated, based on the dialog inputted, using speech-synthesizing equipment. Next, the probability for the
camera-work is decided, by using data from statistical survey from real TV shows together with random numbers and the pre-calculated speaking times. From the generated camera work and the previously inputted dialog a TVML script is created. The TVML script can then be played back on the TVML player.

Dr. Hayashi also developed an “online” system besides the “off-line system” described above. In the “online” system, it creates camera work switching triggers in real-time based on the same algorithm, using real-time measured speaking time.

The collection of statistical data was made for different situations, depending on what type of camera shot there was for the moment, if there was a change in speaker or not, and if it was the host or the guest that was speaking. The type of camera shots used in this system was:
- Host Close Up
- Guest Close Up
- Two person shot
- Overhead shot

The statistical survey on these camera shots determined the following for both guest and host:
- Duration probability for the previous camera shot when there is a change of speaker.
- Duration probability for the previous camera shot when there isn’t a change of speaker.

Picture 6: Schematic picture of how the automatic camera work is generated with the TVML player
Transition Probability for a specific type of camera shot when there is a change of speaker.
Transition Probability for a specific type of camera shot when there isn’t a change of speaker.

When the system is running, it uses the statistical data for different situations to determine the camera work. In the process of generating camera work, the need for camera work and the type of camera work are determined by duration probability and transition probability for specific camera shots, respectively, based on appropriate random numbers.

The system is built up by two algorithms, a transition and duration part, each one using two different data sets, one for the situation where there is a change of speaker and one for the situation where there is no change of speaker. The duration probability algorithm is called on a regular time basis and uses the data set most proper for the moment. If the duration probability algorithm decides there is going to be a camera switch it calls the transition probability algorithm that decides which camera to use.

If there is no change of speaker, the camera switching generator uses the duration probability algorithm with the data set for “no change of speaker” to decide if there is going to be a camera switch or not. If the decision is that it is going to be a camera switch the CSG calls the transition probability algorithm and uses the data set for “no change of speaker”, to decide which camera to use. If the camera work is not needed, the previous camerawork continues.

If there is a change of speaker, the system first uses the duration probability algorithm and the data set for “change of speaker”, to decide if there is going to be a camera switch. If the decision is that it will be a camera switch the transition probability algorithm decides which camera shot to use, using the data set for “change of speaker”.

*Picture 7: Schematic picture of the algorithm developed by Dr. Hayashi*
3.3.3. Problems with this algorithm
This algorithm was the first one used to test if automatic camera work would work. It is limited to two characters and a very limited amount of cameras. The camera switches doesn’t have too much correlation with what is going on in the shows, the camera switches are only based on the time since last camera switch and change of speaker. Dr.Hayashi has described the following areas as areas to continue to work on.

The statistical data for this project was based on a famous Japanese two man show called “Tetsku Room”. Dr.Hayashi recommends further research to generate more general statistical data.

The algorithm needs to be expanded to be able to handle more characters, at least three.

This study only concerns a limited variety of camera shots. These are close-up, two-shot and overhead shot. In real TV programs there are more options.

Counter measure must be implemented for situations where a character “chims in” and annoying camera work is produced. For example when a character says a short word like “No”, “Yes” or “Hmm” and the camera zooms in on the character that made the utterance.

3.3.4. System for automatic TV program generation using dialog transcription [7]
This was the second research project conducted at NHK for automatic camera work generation and it was released 2,5 month after I started my research. The system was developed by Ariyasu-san. The system was constructed for 1-8 characters and its decision algorithm is based on statistical data combined with random numbers together with rules based on real cameramen’s experiences. The system takes concerns about more factors than the first system developed. The system for example places the characters in the room according to how much and when they speak. It also takes in consideration the angle that the characters are shot in. This is to give the viewer a better image of the characters spatial position. This is done by using a pre defined studio environment for 8 different situations.

3.3.5. The system
The camera switching in this system is divided in two parts, the decision to switch camera and the decision on what the next camera is going to focus on. This system uses some very accurate statistical data that is the result analysis of 30 hours of debate programs.

3.3.6. Choice of camera focus
Ariyasu-san [7] has divided up the different possible shots in different situations based on how the TV viewer understands the camera clips. These are:

- Close shot on the speaker
- Several peoples shots including speaker
- Close shot on none speaker
- Several peoples shot excluding the speaker
- Dolly shot

It was then discovered that there is a big correlation between the different types of shots as you can see in table 3.
<table>
<thead>
<tr>
<th>after/before</th>
<th>speaker 1S</th>
<th>include speaker</th>
<th>dolly</th>
<th>include participant</th>
<th>participant 1 S</th>
</tr>
</thead>
<tbody>
<tr>
<td>speaker 1S</td>
<td>11%</td>
<td>85%</td>
<td>73%</td>
<td>77%</td>
<td>72%</td>
</tr>
<tr>
<td>include speaker</td>
<td>6%</td>
<td>14%</td>
<td>11%</td>
<td>14%</td>
<td>6%</td>
</tr>
<tr>
<td>dolly</td>
<td>8%</td>
<td>1%</td>
<td>1%</td>
<td>2%</td>
<td>1%</td>
</tr>
<tr>
<td>include participant</td>
<td>15%</td>
<td>2%</td>
<td>4%</td>
<td>3%</td>
<td>2%</td>
</tr>
<tr>
<td>participant 1 S</td>
<td>35%</td>
<td>8%</td>
<td>11%</td>
<td>4%</td>
<td>19%</td>
</tr>
</tbody>
</table>

Table 3: In the upper row you can see the previous camera shot, in the leftmost column you can see the next coming camera shot. The values are the probabilities to go from a camera shot in the top row to a camera shot in the leftmost column.

To decide which shot to use, the system uses a random number generator in combination with the data in table 3.

3.3.7. Decision when to switch camera

The decision when to switch camera uses a quite complicated algorithm that has been developed by thorough analyses. The algorithm takes in consideration several factors that have been analyzed by multiple linear regression and the correlation between these factors and the switching time is 0.83 (the contribution ratio is 69%). Each utterance in the input text is categorized based on these factors.

The factors that are considered are:
- Length of remark
- Picture effect, if the utterance is the first one for a specific character, her/his name will be superimposed on the screen and the utterance is categorized as superimposed.
- The previous camera shot (Kind of shot)
- Other factors, this system has a gesture module that allows the characters to show feelings.
Table 4: The table shows the variables used in Ariyasu-san’s algorithm.

<table>
<thead>
<tr>
<th>factor</th>
<th>category</th>
<th>category coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>length of remark</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X11 0s~30s</td>
<td>A1</td>
<td>-0.75432</td>
</tr>
<tr>
<td>X12 30s~60s</td>
<td>A2</td>
<td>-0.0639</td>
</tr>
<tr>
<td>X13 60s~90s</td>
<td>A3</td>
<td>0.393439</td>
</tr>
<tr>
<td>X14 90s~120s</td>
<td>A4</td>
<td>0.517815</td>
</tr>
<tr>
<td>X15 120s~180s</td>
<td>A5</td>
<td>0.409238</td>
</tr>
<tr>
<td>X16 180s over</td>
<td>A6</td>
<td>1.721694</td>
</tr>
<tr>
<td>picture effect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X21 super impose</td>
<td>B1</td>
<td>10.86068</td>
</tr>
<tr>
<td>X22 flip</td>
<td>B2</td>
<td>6.077252</td>
</tr>
<tr>
<td>X23 no effect</td>
<td>B3</td>
<td>-0.68935</td>
</tr>
<tr>
<td>kind of shot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X31 speaker S1</td>
<td>C1</td>
<td>2.219515</td>
</tr>
<tr>
<td>X32 include speaker 1</td>
<td>C2</td>
<td>-1.32513</td>
</tr>
<tr>
<td>X33 include speaker 2</td>
<td>C3</td>
<td>-2.06837</td>
</tr>
<tr>
<td>X34 dolly (speaker)</td>
<td>C4</td>
<td>2.320856</td>
</tr>
<tr>
<td>X35 participants</td>
<td>C5</td>
<td>-3.28907</td>
</tr>
<tr>
<td>X36 dolly (participants)</td>
<td>C6</td>
<td>0.209537</td>
</tr>
<tr>
<td>X37 follow (breaking)</td>
<td>C7</td>
<td>-3.94285</td>
</tr>
<tr>
<td>X38 follow (gesture)</td>
<td>C8</td>
<td>-2.81013</td>
</tr>
<tr>
<td>X39 follow (expression)</td>
<td>C9</td>
<td>-1.8129</td>
</tr>
<tr>
<td>X40 other</td>
<td>C0</td>
<td>-4.69</td>
</tr>
<tr>
<td>other factors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X41 gesture</td>
<td>D1</td>
<td>-2.8671</td>
</tr>
<tr>
<td>X42 quotation of name</td>
<td>D2</td>
<td>-2.8549</td>
</tr>
<tr>
<td>X43 dolly--&gt;speaker</td>
<td>D3</td>
<td>9.143352</td>
</tr>
<tr>
<td>X44 breaking</td>
<td>D4</td>
<td>-4.59026</td>
</tr>
<tr>
<td>X45 heated</td>
<td>D5</td>
<td>-7.75546</td>
</tr>
<tr>
<td>X46 dolly--&gt;whole</td>
<td>D6</td>
<td>9.245671</td>
</tr>
<tr>
<td>X47 program structure</td>
<td>D7</td>
<td>20.77709</td>
</tr>
<tr>
<td>X48 other</td>
<td>D8</td>
<td>-0.36744</td>
</tr>
<tr>
<td>constant term</td>
<td>E</td>
<td>9.88327</td>
</tr>
</tbody>
</table>

Each category is assigned one Xij variable shown in the table 4. When a category is assigned to an input utterance the corresponding Xij is set to 1. If Xij is not assigned, it will be set to 0. Each category has a coefficient as seen in the table 4. By using the described coefficients one can calculate the switching time \( t \) by the following formula.

\[
t = \sum_{i=1}^{6} A_i X_{1i} + \sum_{i=1}^{3} B_i X_{2i} + \sum_{i=1}^{10} C_i X_{3i} + \sum_{i=1}^{8} D_i X_{4i} + E (\text{seconds})
\]

This system has as previously mentioned a special system for placing each character, but because my system will not use that kind of technologies will I not go further in to that.
4. Development

This chapter describes the system that I developed and the test results gained after trying it on real people.

4.1. Ideas behind the new system

The goal of the Automatic TV program production is to produce a TV show that reminds as much as possible of a real TV show. My research is however limited to the camera work of a TV show.

4.1.1. How is a real TV show produced

For a regularly TV talk show there is about three cameramen. The work of the cameramen is quit machine like, they get instructions from the control room and executes the instructions. In the control room there are quite a lot of people, a video technician, an audio technician, a producer etc. If you look at the production of a TV show, you can see that the only actual creating person is the producer, he/she takes all the decisions, the rest of the crew executes his/her commands and some time catch up errors that he/she makes.

The next step is to understand why a TV producer makes the decisions that he/she makes. The most important goal for the TV producer is to create a TV show that is easy to follow and that gives a clear understanding of who is talking to whom, and what the spatial relationship between the characters are. The second goal is to make the TV show interesting. The TV producer makes a TV show interesting by switching cameras, size of picture and angle of cameras. In short words, he/she creates a variation by switching between the different camera shots. A TV show would be very easy to follow if you would fix a camera in one position to make a group shot all the time, but the show would be quite boring for the viewer, the variation of camera shots is needed.

The last step is to understand how a TV producer makes his/her decisions. The way a TV producer makes his/her decisions is a combination of rules, experience and understanding of the situation.

4.1.2. How can a real producer be simulated

To create automatic camera clipping, we first have to decide which functions are needed from the production crew. The most natural idea that comes to my mind is that you can look at a TV production as a producer controlling the cameras. And that is also the approach that I am going to take. It means that the system will contain an algorithm that will simulate a real producer’s decisions.

My system has to be able to make the same type of decisions as a TV producer, the drawback is however that my system have very little input in comparison to a real TV producer. A real TV producer has his/her 5 senses, an understanding for what is going on in the show and a pre-knowledge about the characters. The only information that my system will have is:

- Who is talking
- Who have been talking and for how long time
- Which camera is in use
Which camera have been in use and for how long time, and who was speaking at that time
The total time that the talk show have been on.

With these inputs, my system has to make the same decisions as a TV producer.

4.1.3. Differences from previous systems
Some differences between my software and the previous ones developed are that in Dr.Hayashi’s [4] software, it was only possible to have two characters with four pre defined camera shots.

Ariyasu-san’s software allowed 1-8 characters using one movable camera with pre defined camera positions, this gave very good shows but made it hard to change the studio environment.

My software allows 2-28 characters and 2-28 cameras which are all easy to define. This makes it easy to change the studio environment. My software also allows user interaction, which none of the previous systems allowed. It will also be very easy for the user to decide in which order the speaking characters will sit in. The main difference is however that my system is knowledge based, and therefore has a real connection between the camera switching and what is happening in the TV show.

4.2. The system basics
To allow multiple characters and cameras I developed a very flexible system that uses a naming convention together with binary numbers definition. Using binary numbers makes the system very fast and the naming convention is very easy to understand for humans. To allow maximum flexibility I also divided up cameras and camera shots. If a camera can make a shot on two persons sitting beside each other it means that the camera can make 3 different shots, a close-shot on each character and a two-shot including both characters. Defining the camera shots by themselves makes the system very flexible.

4.2.1. Character and camera definition
The system allows the characters to have a wide variety of formations and at the same time it makes it possible to have 2-28 characters and 2-28 cameras. The system is built up by using binary numbers and has a structure based on the order the characters sit in. They are defined by a capital letter A-Z, where A is the first character and B the second character etc. (going from left to right). Each of these letters relates to a binary number. For example A has the number 0001 and B 0010 and so on. To do this I’m using a 32 bit integer variable. The cameras are then named and defined by the characters it can make a good shot on. So if we had a 5 people show we would have the characters.

\[
\begin{align*}
A & : 0000 \ 0001 \\
B & : 0000 \ 0010 \\
C & : 0000 \ 0100 \\
D & : 0000 \ 1000 \\
E & : 0001 \ 0000 \\
\end{align*}
\]

The system must contain at least one camera that can make one shot on all characters and that camera would in this case be named AE and have the binary number 0001 0001. A camera that could make a god shot on the characters A to C would be named AC and have the binary
number 0000 0101. If a camera only can be used for one character, for example A, it would be named AA and have the binary number 0001. The system then creates all the possible camera shots by using the character and camera definitions. For example the camera AC which has the binary number 0000 0101 could make 6 different shots:

- **AA**: 0000 0001
- **BB**: 0000 0010
- **CC**: 0000 0100
- **AB**: 0000 0011
- **BC**: 0000 0110
- **AC**: 0000 0101

The system collects data about what is going on in the show during the whole show. The data that the system collects are:

- Who is speaking
- Previous speaker
- Camera in use
- Previous camera
- Which Shot is in use
- Previous shot
- Each characters speaking time
- The length of the last speech each character made
- Total time a camera has been in use
- Time since last camera action
- Time since the show started
- Attention value (is described in Layer 2)
The data is used in the decision process described later. But they are also used in the construction of camera shots while the show is running. For example, if the system decides to have a camera shot on the speaker and the previous speaker, and the speaker is A 0000 0001 and the previous speaker is C 0000 0100. The shot is then easily created by using the binary operator “&”, and the new shot is then 0000 0001&0000 0100=0000 0101 which is an AC shot. All the control functions in the software uses binary operations like the one above.

4.2.2. System introduction

The system is divided in two different layers, Layer 1 and Layer 2. Layer 1 is called each 1ms by the windows control system and handles the decision if there is going to be a camera switch. Layer 2 is called by Layer 1 if the decision is to have a camera switch and decides what to include in the camera shot. Several parts from the system then decides what type of switch it will be and which camera to use.

```
TRIGGER
Windows Timer
```

```
LAYER 1
Decides if there is going to be a camera switch or not
```

```
LAYER 2
Decides what to include in the camera shot
```

```
SYSTEM
Decides what type of switch and which camera to use
```

```
Executes decision
```

*Picture 8: Schematic picture of my system.*
4.3. **Layer 1: The decision on when to switch shot**

The demand from NHK was that the camera switching frequency should be user controlled. Ariyasu-san[7] developed a very good system for the camera switching frequency that would have been great to use. However by the time I got access to her system I found it too time demanding and “hard for the user to understand” to try to make her system user controlled because it is a very complicated system. Instead I developed a system that is very easy for users to understand and control, but it is not as scientific based as her system.

4.3.1. **Idea behind the algorithm**

When the user starts the program a default frequency settings file is loaded. If the user pushes the advanced settings button a new window appears where there will be possibilities to adjust the settings. Layer 1 is called every 1ms by the windows system timer, main reason for this high frequency is to have as fast response as possible when there is a change of speaker. Layer 1 contains of two different algorithms, Camera Switching by Transition Probability and Camera Switching by Duration Probability. When Layer 1 is called, it first checks if there is a change of speaker, if true, the camera switch by transition probability algorithm is called else the camera switch by duration probability is called.
4.3.2. Camera switch by transition probability

The camera switching by transition probability is very straightforward. The setting is a probability value between 0 and 100 and is set by the default file or the user in the advanced settings menu. If there is a change of speaker the random value generator will be called, and if the value is below the set value, a camera switch occurs and Layer 2 is called.
4.3.3. Camera switch by duration probability

The camera switch by duration probability consists of two parts. The first part checks if the speaker is in the shot. There is a maximum time limit for a shot not including the speaker, this time limit is loaded when you start the software and it is also possible to adjust it in the advanced settings menu.

The second part reminds of the camera switching by transition probability. The difference is that it needs three parameters, a minimum time value, a maximum time value and a minimum probability.

The minimum time value is the minimum time before a camera switch occurs. For example if you want at least 6 seconds between the camera clips, then the minimum value would be 6 seconds.

The maximum time value is the time when you will force a camera clip to occur, that would be if you didn’t want to have any shots longer than 15 second, the maximum time would then be set to 15 seconds.

The minimum probability is the probability for a camera switch at the minimum time. The probability values between the minimum and maximum time are calculated by the following formula, where “prob” is the probability for a camera switch. The variable “prob” will be a value between 0-100.

\[
prob = \frac{100 - \text{min}_{\text{prob}}}{\text{max}_{\text{time}} - \text{min}_{\text{time}}} \times \text{time} \_\text{since} \_\text{last} \_\text{switch} + \text{min}_{\text{prob}}
\]

This value is then compared with a random value and if the random value is below the probability for a camera switch the camera switch will occur and Layer 2 is called.

All the variables are preloaded when the user starts the program and it is possible to change them in the advanced settings menu.

4.4. Layer 2: The decision which shot to use

A regular talk show consists of a series of close shots with group shots and non-speaker shots inserted every now and then [7]. The question is when to insert group-shots and non-speaker-shots. I will here under describe how I have chosen to solve this problem.

4.4.1. Idea behind the algorithm

My starting point for this algorithm is based on an observation that I have made in real life. When a group of people is sitting and speaking around a table, each person gets different amount of visual attention from the other persons. One thing that will cause the other persons to give visual attention is if the speaker says something controversial, however today technology is not advanced enough to semantically analyze the content of the speech. Another thing that I also believe draws attention to the speaker is how much he/she he/she has spoken in relationship to the other persons. If one person speaks a lot, the other persons will start to
wander with their eyes and look at other things. If a person that hasn’t spoken so much before suddenly says something, the other persons will immediately give him/her visual attention, because he/she probably has something important to say. My idea is to try this theory in my software.

How would visual attention be translated to a TV-studio environment? My idea is that a close shot on the speaker gives the most visual attention to the speaker. For each character added to a camera shot, the less visual attention is given to the speaker. The least visual attention a character can get is when he/she is speaking and the camera is shooting a listener instead of the speaker. This means that a camera shot including the speaker and one non speaking person gives more visual attention to the speaker, than a camera shot including the speaker and two non speaking persons.

I have graded the different types of camera shots in the following way, starting with the most visual attention at the top of the list.

1. Close shot on the speaker
2. Shot on speaker including 1 non speaking characters
3. Shot on speaker including 2 non speaking characters
4. Shot on speaker including 3 non speaking characters
5. Shot on speaker including >3 non speaking characters
6. Shot not including the speaker

My idea is to insert group shots with many characters and non-speaking shots when a character that has spoken a lot is speaking. When a character that has not spoken a lot speaks, there will be a majority of close shots and group shots containing a small amount of characters.

4.4.2. Deciding who is a dominant speaker

To avoid that for example a long introduction will have to big influence on a long show I have developed a time measurement variable that I call “attention value”. Basically the attention value is the time that a character have been speaking and been shot by the camera at the same time, divided by the amount of characters included in the shot. If there is a non-speaker shot when a character is talking, no time will be added. The attention value is then used to decide if a character is a dominant or non-dominant speaker.

```
If( speaker in shot){
    Speaker->AddAttention_Value(time_since_last_camera_switch/
    amount_of_characters_in_shot)
}else{
    //No time added
}
```

4.4.3. Comparing characters

I tried two different methods to compare if a character has been speaking more or less than the other characters. The first method was to compare the speaker’s attention value with the average attention value for all characters. The second method was to compare the speaker’s attention value with the previous speaker’s attention value. The test was done in the middle of the development process and the testing was done on me and my supervisor.
The second method (to compare the speaker with the previous speaker) gave the best result, trying it on different types of TV shows. I believe that the reason for that is that a TV show is momentarily. What is important is the relationship between the characters talking for the moment.

Comparing the speaker with the average speaker gave strange results in the beginning of a show. For example if there is a five person show and two characters dominate the discussion for the first 15 minutes, both of them will be dominant speakers comparing them with the average. Because the average will be the average of all five characters attention value, it means that the total attention value will be divided by five. Comparing the speaker with previous speaker gives a better understanding of the relationship between the speakers.

4.4.4. Functions developed
To make the final design of the algorithm as flexible and easy as possible I developed several functions that defined different types of camera shots. These functions are source code functions but are important for the reader for further understanding.

CloseUp("SPEAKER");
Makes a close up on the speaker

CloseUp("NON_SPEAKER");
Makes a close up on the previous speaker, if there is no previous speaker, it will make a close up on a random non speaker

TwoShot("SPEAKER_AND_PRE_SPEAKER");
Makes a shot including the speaker, the previous speaker and all the characters in-between them.

TwoShot("SPEAKER_AND_NON_SPEAKER");
Makes a shot on the speaker, and the non speaker that sits on the speaker’s side.

GroupShot("ALL_CHARACTERS");
Makes a group shot including all the characters

GroupShot("SPEAKER_CENTER_3_SHOT");
Makes a three shot with the speaker in the middle, if the shot is not possible, it will call the ALL_CHARACTERS function.

GroupShot("SPEAKER_CENTER_5_SHOT");
Makes a five shot with the speaker in the middle, if the shot is not possible, it will call the SPEAKER_CENTER_3_SHOT function.

I have developed several more functions but because I don’t use them in this algorithm I will not describe them.

4.4.5. Algorithm
The final algorithm has three main objectives:
1. Give the most visual attention to the character that has spoken least.
2. Have a big difference between the camera shots following each other.
3. Have big variation, so the show doesn’t have a machine like appearance.

To do this the system has different cases for different situations. The cases are dependent on four different factors:

1. If the speaker was included in the previous camera shot.
2. If the previous camera shot was a single shot or a multiple shot.
3. What the reason for the camera switch was, switch of speaker or time.
4. If the speaker is a dominant speaker or non dominant speaker compared with the previous speaker.

Each time there is a decision for a camera switch, the system checks the present situation, and finds out, what type of shot there is for the moment, what the reason for the switch is and if the present speaker is included in the shot. Thereafter it decides what type of camera shot it will be. Each case consists of probabilities for certain camera shots. The reason to use probabilities is to avoid machine like performance. However there will always be a bigger probability for a close shot on the speaker when he/she is a non dominant speaker. In the beginning of the development I didn’t have the probability functions, they were added later. The probabilities are fixed numbers, and I started out setting them so they would fit my objectives. They were later calibrated under the calibration step, described in the calibration part.

If you compare the different situations with each other you can see that a non dominant speaker always have higher probability for a close shot or narrow shot compared with a dominant speaker. If the reason for the camera switch is transition probability, the total probability for close shots or very narrow shots will be higher compared to when there is a duration probability switch. If the speaker is not included in the previous shot, it will be a very high probability for close shots. The show will always start with a group shot, and until the second person starts speaking will the choice of camera shot be executed by random numbers. There is also a built in check that makes sure that one type of shot can not be repeated more than two times in a row. The initialization scripts usually contains enough cameras to make the same type of shots from two different angles, but if I would allow to have more than two shots of the same type after each other it would probably result in unnatural behavior.

I will here under describe each of these units with pseudo code and why I have chosen to design them as I have.

### 4.4.6. Camera switch when speaker is not in shot

In this case it has no meaning what the previous shot was focusing on or what the reason for the switch was, most important is to get the speaker in the shot. As you can see in the pseudo code, if the speaker is non-dominant he/she will be captured by a close shot and if he/she is a dominant speaker he/she can be captured by a variation of shots.

```pseudo
if(speaker have spoken less than previous speaker ){
    CloseUp("SPEAKER");
} else{
    random =getrandom(); //[getrandom()→0-100]
    if(random<30){
        CloseUp("SPEAKER");
    }
}
```


} else if(random<85){
    TwoShot("SPEAKER_AND_PRE_SPEAKER");
} else {
    GroupShot("ALL_CHARACTERS");
}
}

### 4.4.7. Camera switch when the character already is in the shot, the shot is a close shot and the reason for the switch is duration

This situation is quite common situation. There is a close shot on the speaker when there is time for a shot- or camera-change. This situation is divided in to 2 different main situations, if the speaker is a non dominant speaker or a dominant speaker. As you can see there is a bigger possibility for a high attention shot when the speaker is non dominant.

if( speaker have spoken less than previous speaker){
    random=getrandom();
    if(random<15){
        if(shot==pre_shot){
            CloseUp("SPEAKER");
        } else{
            GroupShot("SPEAKER_AND_PRE_SPEAKER");
        }
    } else if(random<75){
        TwoShot("SPEAKER_AND_PRE_SPEAKER");
    } else{
        GroupShot("SPEAKER_CENTER_3_SHOT");
    }
} else{
    random=getrandom();
    if(random<70){
        TwoShot("SPEAKER_AND_PRE_SPEAKER");
    } else if(random<80){
        GroupShot("SPEAKER_CENTER_5_SHOT");
    } else if(random<90){
        GroupShot("ALL_CHARACTERS");
    } else{
        CloseUp("NON_SPEAKER");
    }
}

### 4.4.8. Camera switch when the character already is in the shot, the shot is a multiple shot and the reason for the switch is duration

This situation is also quite common. There is a multiple shot including the speaker when there is time for a shot- or camera-change. As you can see there is a bigger possibility for a high attention shot when the speaker is non dominant.

if(speaker have spoken less than previous speaker){
    if(shot==pre_shot){

CloseUp("SPEAKER");
}

if(speaker have spoken less than previous speaker){
    CloseUp("SPEAKER");
} else{
    random = getrandom();
    if(random < 90) {
        CloseUp("SPEAKER");
    } else if(random < 96) {
        TwoShot("SPEAKER_AND_PRE_SPEAKER");
    } else {
        GroupShot("SPEAKER_CENTER_3_SHOT");
    }
}

4.4.9. Character already in the shot, the shot is a close shot and the reason for the switch is transition
This situation is the most uncommon situation to appear, the reason is that the previous shot has to be a close shot on a non-speaker, and that non-speaker then have to start to speak. To avoid a camera switch that only would result in confusing behavior, I have decided to skip this camera switch, and have a so called empty switch.

4.4.10. Character already in the shot, the shot is a multiple shot and the reason for the switch is transition
This situation is quite common. There is a change of speaker but the character that is speaking is already included in the camera shot. This situation is divided in to 2 different situations, dominant and non dominant speaker. As you can see there is a bigger possibility for a high attention shot when the speaker is non dominant.

if(speaker have spoken less than previous speaker){
    CloseUp("SPEAKER");
} else{
    random = getrandom();
    if(random < 50) {
        CloseUp("SPEAKER");
    } else if(random < 60) {
        TwoShot("SPEAKER_AND_PRE_SPEAKER");
    } else if(random < 90) {
        GroupShot("ALL_CHARACTERS");
    } else {
        CloseUp("NON_SPEAKER");
    }
}
4.4.11. Special situation when the program starts

When the program starts there is no previous speaker and all the camera choices are then made by a random function. As you can see, the main objective for this part is to alternate close shots with different types of group shots.

4.5. The decision which camera to use

The decision on which camera to use is done after the shot is decided. The reason for that is that I believe that the choice of what to be included in the shot is the most important decision because it tells the viewer what is important in the show. Before the decision on which camera to use, the system has to decide which type of camera switch it wants, a static clip or a moving camera (panning and zooming). The decision on what type of switch the user wants is done by using random values in combination with preset probabilities. The user can himself/herself modify the probability for a moving camera switch in the advanced settings menu. When the software starts an appropriate probability value is loaded as a presetting. The decision of which camera to use then uses two different algorithms depending on, if there is going to be a moving or static camera switch. If the decision is to have a moving camera switch, the system will check if it is possible for the camera in use to get the decided shot if not, the system will switch to the static camera switch. The static camera switch will always choose a camera that can make the shot and the camera that will be chosen is the camera that has been used the least. Every time a camera is in use the time is measured and saved. This is to get maximum variation of the cameras. The basic structure of the algorithm can be seen in picture 10.
4.6. **Gesture generation**

This is a feature that was not supposed to be included in this thesis from the beginning, it is only added to get a more “live” show. What it does is to turn the head of the listeners so they
face the speaker. A random function decides if the speaker is looking forward or at the previous speaker. The turning of the heads is done whenever there is a change of speaker.

4.7. Start-up data

To make a TVML show with automatic camera work you first have to consider which data the TVML Player needs and at what time. The basic idea is that a TV producer will write a dialog script looking something like this:

Anna: Hi how are you.
Bob: Fine thanks.
Anna: So what shall we do?
Bob: I don’t know.

And that the rest of the show should be automatically generated. But there are some question marks. Where does the choice of studio get decided, who decide where the characters are going to sit in the studio etc.?

The two previous projects about automatic camerawork used an initialization file which contained the studio and character data, and I am choosing to take the same approach. Dr. Hayashi’s [7] software produced only a two man show and had one default initialization file that was hard coded. Ariyasu-sans [4] software could generate shows with 1-8 characters but only with the characters sitting in an octagon pattern, this limited the need for startup scripts to 8, which were also hard coded. As you can see, in the previous developed applications there was only one default initialization file for each situation which means that the viewer was locked to one studio environment. In my software the user himself/herself chooses the initialization file. This is done by using a naming convention. When the user loads his/her dialog script, the software detects how many characters there are in the show and suggest several initialization files with different studios and characters. The initialization files are named according to how many guests and hosts there are in the show. For example if there is a initialization file for a show with 3 guests and 1 host, the file ending would be “G3H1”. The designer of the initialization files can then name the initialization file with an appropriate name. For example if there is an exclusive talk show studio with 3 guests and 1 host, the file name could then be “exclusive_talkshow_studio.G1H3”. This makes the usage of the software very dynamic. When the user has loaded his/her dialog script and chosen an initialization file, he/she only has to push the play button and relax and enjoy.

4.7.1. Placing the characters

The characters position in the studio is described in the header of the dialog script as you can see in picture 11. Where “A” is the Left most position and “Z” the right most position. This data are then used by the system to place the characters.
As you can see, the script contains a head and a text part. The head is for placing the characters and the text part is to define the utterances.

### 4.8. Using the system

When you start the system the basic control GUI appears.

![External Control](image)

*Picture 12: The picture shows the main GUI.*
The first thing for the user to do is to chose a dialog script, this is done by clicking on the “Select dialog” button, the system will then allow the user to search for dialog scripts called *.skit.

When the user has chosen a dialog script, the system allows the user to choose an initialization file by clicking on the “Select Init File” button. The system, then displays the initializing files that fits the dialog script.
Thereafter the user pushes the “Execute Player” Button, and the TVML Player window appears.

Picture 15: The picture shows the main GUI when the TVML Player has been started.

The user then pushes the “Play” button and views a terrific show.

Picture 16: The picture shows a TVML show that is automatically generated by my software.
4.9. Advanced settings

When the user pushes the advanced settings button the advanced settings window appears. All the settings are loaded when the user starts the application, the user can change the settings and save them by pressing the “save” button. If the user names the file “default.frq”, it will be loaded automatically the next time the user starts the application. In picture 16 you can see the Advanced settings GUI, the red numbers are not part of the GUI, they are the references to the list under picture 16.

![Advanced Settings GUI](image)

**Picture 16:** The picture shows the advanced settings GUI.

1. Probability for a camera switch when change of speaker, is the probability for a camera switch when there is a change of speaker. The highest value is 100 and the lowest is 1.
2. Minimum time for a camera switch is the minimum time between the camera switches when the speaking character is in the picture and there is no change of speaker.
3. Maximum time for a camera switch is the maximum time between the camera switches when the speaking character is in the picture and there is no change of speaker.
4. Probability for a camera switch by the minimum time is the probability for a camera switch at the minimum time. The probability rises linearly until the maximum time where the probability for a camera switch is 100, which is the maximum probability value that forces a camera switch to occur.
5. Maximum time for a camera shot without the speaker in it is the maximum time for a camera shot when the speaker is not in the picture.
6. Probability for moving camera is the probability for panning and zooming action.
7. Moving camera speed is the speed of the camera.

4.10. Calibrating the system

The development was conducted by me under a five month period. Under that time I tested the system to get a good algorithm. To judge the software’s performance I used the visual output in combination with a log file (For example see Enclosure 6) that my software produces. My testing material was interviews with famous people, discussion shows from CNN (Crossfire) and some teleconferences.

It is hard to decide when to quit the development, because it is hard to say how perfect camera work should look in a debate show. What I did was to develop the algorithm until it produced what me and Dr. Hayashi thought was a good discussion show. Thereafter I tested it on several professional media and broadcasting engineers to get their response on it. Between each interview, I corrected errors that the persons discovered. These interviews worked as a powered development step.

4.10.1. The interviews

Totally I made six interviews. Each interview target got to watch a TVML show produced by my software. The dialog scripts used was taken from the TV debate show CNN Crossfire. Each time an interview target thought that something was wrong he/she pushed the ERROR button, which then made a mark in the log file. The show took about 10-15 minutes (until the first commercial break). Thereafter I discussed the show with the interview target, asking him/her about the reasons for pushing the ERROR button. I also asked them about their total impression from the show, if it was easy or hard to follow, if they thought it was hard to understand the spatial placement of the characters and if they had some other remarks about the show. Then I explained the software for the interview target, let him/her try it and asked them to give comments about the software. The total interview had the structure of a deep interview and took approximately 45 minutes. After each interview session I evaluated the comments and reevaluated the previous interviews and if there was something that I thought needed and could be improved within my algorithms structure, I did so.

The CNN shows used, contained video clips, these were not included in my shows. Therefore I rewrote the scripts so they fitted my format. To make the show more understandable I added some extra speech lines. The complete interviews are viewable in Appendix 1 and the dialog script and initialization file used in the interviews are viewable in Enclosure 3 and 4.

4.10.2. Changes made during the interviews

The interviews were really useful, a lot of changes were made during this process. I will here give a briefing on the comments that the interview targets made and the changes that resulted from that. To see the full interviews please see Appendix 1.

Some of the researchers thought that there were many “mismatch” switches. A mismatch switch in their meaning is a camera switch to the same type of shot, but the
angle between the cameras is too small so the switch just looks like noise. Two different types of changes solved this problem.

The camera positions are defined in the initialization scripts. By changing the camera positions in the initialization script most of the mismatch problem were solved. But even when there was a big distance between the cameras, some people still thought that there was a mismatch problem. I then changed the probabilities in the part that decides “what type of shot to use” so the probability to have the same type of shot two times in a row became much lower.

Going from the same type of shot but with a different camera was something that some persons thought was good, but others thought was against the basic camera rules even when there was a big angle between the cameras.

This resulted in that I lowered the probability for doing this in the part that decides “what type of shot to use”.

Using zooming and panning was something that most of the interview targets thought gave strange results except for one person that thought it was really good. Most of them thought that zooming is a very “strong” camera action and it needs very good timing.

My software uses random values combined with the user settings to decide when there should be a zooming or panning action (moving camera). To solve this, one have to develop a more advanced control unit to decide when to use zooming or panning, this was not possible for me at the moment. But there was one problem that I could solve quite easy. I constructed a locking function that limited the zooming and panning so there could at most be a moving camera shot every second camera switch. This action made it impossible for the system to have two zooming actions following each other, for example zooming in and thereafter zooming out, which may result in a home video feeling.

Some of the interview targets thought that camera switches that went from one type of shot to a similar type of shot were annoying. For example if there is a show with 5 characters and the first shot includes all five characters and the second shot includes four characters. They thought that a basic rule in TV production is to have big differences between the different types of shots.

This was partly solved by changing the control part and the probabilities for the type of camera shot in the unit that decides “what type of shot to use”.

Most of the interview targets would have liked to have gesture and eye movements.

This was not a part of my job so I left that for further research.

Some of the interview targets thought that the time it took to switch camera shot when there was a change of speaker was to long, but some thought that the delay was good and gave a realistic touch.
The time delay is built-in in the version of the TVML Player that I used. In the next version of the TVML Player it will be possible to have faster switches.

Some people thought that it was hard to understand some parts of the GUI. Before the last interview session the Advanced Settings menu hade an option that allowed the user to change the ratio for the attention value between the different shots.

I changed the system so that it was done dynamical.

Some persons thought that the group shots appeared to long and some thought that they appeared too frequently.

To solve this problem, I would have to expand the part of the algorithm that decides “when to switch shots”. Because of time reasons I have left this for further development.

4.11. Testing the system

The testing of the system took the same approach as the calibrating part, with the only difference that I did not change the software in-between the interviews and I used three different shows with different amount of characters. Totally I interviewed six TV technology researchers. The dialog script and initializing file were changed after every second interview.

4.11.1. The interviews

I let two researchers watch one type of show and then test the software. The length of each show was approximately 10-15 minutes. Thereafter, I interviewed them about their impression and thoughts about the show. The whole test took an approach similar to a deep interview and was held approximately 25 min.

The session started with me introducing them to the software. Thereafter they got to watch 15 minutes from a show produced with my software. The software contains an ERROR button. Each time they thought that there was something strange about the camera work they pushed the ERROR button and a mark was made in the log file.

After the show was over, I ask them:

What their total impression was from the show.
If there was something annoying or strange about the camera work
The characters spatial placement was easy to understand

The questions were not asked strictly, they worked more as a base for a discussion. In the discussion, I introduced the software for the interview target and let him/her test it.

4.11.2. Results

The overall impression of the shows was good. Their comments varied between “good” to “enjoyed the show”. There was however some things that they thought could be improved. Some that was related to my work and some that was not related to my work. There were some differences that depended on how many characters that appeared in the show, but the overall comments were the following.
Quite many persons commented that there were too many group shots compared with the amount of one shots. Some persons thought that the group shots appeared to long time and some thought that they appeared too frequently. When there were very fast speaker switches the system generated confusing camera work because of the time delay, most of the interview targets thought that that looked unnatural. As in the calibration step, people thought that zooming and panning needed better timing. Some people thought that the camera angles “were not perfect”.
5. Summary

I have developed a software for automatic generation of camerawork. The software allows multiple characters, user interaction and uses a knowledgebase algorithm for deciding what will be included in each camera shot. To do this have I developed a naming convention that transforms camera and character names (that are easy for humans to understand) to binary numbers, which the computer can process very fast.

The software consists of three layers. The first layer decides if there is going to be a camera switch or not, this is done by comparing random values with a user preset value range. There are three different value ranges depending on the situation of the moment in the show. The second (which is knowledge based) layer decides what will be included in each shot and it is divided in to several situations depending how much the speaker has spoken compared with the previous speaker, the reason for switching camera, how many characters that was included in the last camera shot and if the speaker was included in the last camera shot. During the show, data is collected for each character and camera and that information is used in the second layer to support the decisions. The third layer consists of different parts from the system and it decides if there is going to be a moving camera switch (panning or zooming) or a static camera switch. This decision is based on random values that are compared with user-preset values. If the decision is to have a static camera switch, the system will chose the camera that has been used the least.

In the end of the development process I calibrated the system after testing it on six NHK researchers. I there after tested the calibrated system on six other NHK researchers to get a overall judgment of my work. The results from the testing were good and the overall judgment varied between “good” and “enjoyed the show”.

39
6. Conclusion

Most of my interview targets thought that the show was quite good. It was hard to get a good judgment on my prototype. The reason was that I had to ask the testing object to look for some specific errors. If one would ask a person to find errors in a regular talk show I believe that most people would find something that he/she would get annoyed from, because they are concentrated on finding errors. But there are some things that several of my testing objects noticed and these matters are subjects for further research. The main things are:

- Zooming is a very strong camera action and in my software it is controlled by random values, it needs better timing and rules that controls what type of pictures it switches in-between.
- The control of the length of the camera clips must be improved. Ariyasu-san developed a good system for this. To develop a user controlled version of her algorithm would be very useful.
- My algorithm has a problem when there is very fast speaker switches, it would be good to develop a system that could handle these cases. An idea would be to have a built in delay or to use group shots when there are very fast camera switches. However one must then predict when there will be fast speaker switches.

Automatic TV program production is a very young technology and it needs a lot of further research to generate shows that have professional broadcasting quality. The topics mentioned above needs further development (some might only be connected with my software). Other areas in automatic TV production that needs to be developed are gesture generation and speech analysis. Camera work is only one part of automatic TV program production, without gesture there will never be a high quality shows. The question is when to generate gesture. For example, it is very easy to make a character nod or shake his/her head. But it needs timing or some qualified estimation. For example, if character A says something that character B agrees with and character B shakes his/her head, it will give a confusing impression. To connect emotions with gesture is a big challenge. The best way would be to make the computer understand what the characters are saying, but that would take a couple of years of development in text analyze. To find a method that makes qualified guesses is a very challenging task.
7. List of literature

Research Articles


8. Enclosures

1. TVML Manual
2. TVML script
3. Initialization script
4. Speech line script
5. The default script produced by my software
6. Log-file
Appendix 1: Calibrating the system

Interviewee 1

Name: Takashi FUKAYA
Position: Researcher
Experience from production: 4 years post production, 4 years CG production
Specialty: Human Interaction
Date: 2003-02-12
Material used: CNN Crossfire [Debate Over the State of the Union Aired January 28, 2003 - 19:00 ET]
Interviewee time: 1h

Person’s comments: His impression from the show was not good. Fukaya-san had complains about the following topics:

- Mismatch shots, camera switches between cameras that are positioned very close to each other.
- He missed some common camera switching rules.
- He didn’t like the delay that appears for the camera switch when there is a change of speaker.
- He didn’t like the camera positions.
- He thought that the characters looked to much alike.
- He missed gesture and eye movement

He liked the idea with real-time control over the studio environment, especially the possibility to change the amount of moving camera shots. But he would like to have a simpler GUI. He liked the settings with a lot of camera movements, zoom-ins, zoom-outs and panning action and would like to see more of that.

Changes made in my software: To solve the mismatch and camera position problem I rewrote the initialization scripts. The Switch delay can not be helped because the version of TVML Player that I used needs a certain time amount between each command line sent to it. The fact that the characters looked alike was also hard to help because to use different characters would make shows with more than two characters impossible because of the need of computer power. Gesture and eye movements were not a part of my work and in this version of TVML there are no such options.
Interview 2

Name: Masanori Sano

Position: Research Engineer

Experience from production: 3 years production control room including working as camera operator, 30% of these 3 years was dedicated to the above tasks the rest was dedicated to development.

Specialty: Meta data, MPEG7

Date: 2003-02-12

Material used: CNN Crossfire [ Debate Over the State of the Union Aired January 28, 2003 - 19:00 ET]

Interviewee time: 1h

Person’s comments:

Over all experience was “Quite Good”, thought that some of the camera switches was really good. The complains he had about the system was the following:

When there was a big probability for moving cameras it some time resulted in a series of moving camera actions, he thought that that was annoying, especially when it resulted in zooming action.

According to his experience zoom-ins is quite uncommon especially going from a multiple shot to another multiple shot. But he thought the zoom outs were good.

He also thought that he could see some patters in the camera work

He thought that panning was inappropriate for this type of show

He thought that some camera switches were too fast, (those ware caused by Duration switch followed fast by a transition switch, he suggested a delayed transition switch)

He thought the angles between the cameras had to be bigger.

Changes made in my software: After this interview I built in a lock so moving camera action can’t occur two times in a row.

Interview 3

Name: Hideki Sumiyoshi

Position: Research Engineer

Experience from production: 2 years VTR operator and news show switching

Specialty: Production system based on databases

Date: 2003-02-13

Material used: CNN Crossfire [ Debate Over the State of the Union Aired January 28, 2003 - 19:00 ET]

Interviewee time: 1h

Person’s comments: Overall impression was good. Watched the show two times and liked it at most when the time in-between the camera switches was long.

The complains he had about the system was the following:

He would have liked bigger angles between the cameras.

He would have like to have gestures on the characters, nodding or shaking head.

When there are fast changes between the speakers he would have liked group shots.

He would like to have back shots, camera shots that shoot one person’s face and the opposite person’s back head.

Changes made in my software: None
Interview 4

Name: Ichiro Yamada
Position: Research Engineer
Experience from production: 3 years as video engineer, specialty sports shows
Specialty: Natural Language Processing
Date: 2003-02-13
Material used: CNN Crossfire [Debate Over the State of the Union Aired January 28, 2003 - 19:00 ET]
Interviewee time: 30m
Person’s comments: Over all impression was “good and interesting work”. Thought the show was good considering it was automatic camera work. The complains he had about the show was:

He thought that camera changes that resulted in same size changes was not good, for example going from one close shot to another close shot. He thought that it would be better to have different zoom-ins on two after each other following camera shots.
He thought that the direction of the characters eyes was misleading in some places in the show.

Changes made in my software: After this interviewee I changed the main algorithm so the probability to alternate close shots with group shots became higher.

Interview 5

Name: Narichika Hamaguchi
Position: Research Engineer
Experience from production: 3 months production training
Specialty: Automatic production system
Date: 2003-02-13
Material used: CNN Crossfire [Debate Over the State of the Union Aired January 28, 2003 - 19:00 ET]
Interviewee time: 20 min
Person’s comments: Over all impression was good. He had comments concerning the following subjects:

He thought that there was to long shots on non speaker.
He thinks that zoom-in is a very strong camera movement and that it needs better timing.

Changes made in my software: The fact that it was to long shots on the non-speakers was the result of a bug that I discovered after this interviewee, this bug disturbed the camera switching for characters on the right and left end of the studio. This bug was fixed.
Interview 6

Name: Kyoko Ariyasu
Position: Research Engineer
Experience from production: 2 years and 6 months (system operator at the News center)
Specialty: Automatic production system
Date: 2003-02-14
Material used: CNN Crossfire [Debate Over the State of the Union Aired January 28, 2003 - 19:00 ET]
Interviewee time: 35 min

Person’s comments: Over all impression was very good. She thought that the show was easy to follow and understand. The comments that she had about the show were the following:

Some times she experienced that the biggest possible group shot was kept too long time.

Short utterances could at some places cause confusing camera clips

Changes made in my software: Before this interviewee it was possible to affect the ratio of the attention value saved. I developed a dynamic system to save the attention value.
Appendix 2: Testing the system

Interview 1

Name: Takao Tsuda
Position: Research Engineer
Experience from production: None
Specialty: Automatic robot camera for broadcasting
Date: 2003-02-19
Material used: CNN Crossfire [Debate Over the State of the Union Aired January 28, 2003 - 19:00 ET]
Interviewee time: 35 min
Person’s comments: Over all impression was good, thought that the camera clipping was better in the second half of the show than in the first half of the show. The points that he thought could be improved was:
  The big group shots were too long, he meant that big group shots were mostly used when production crew wants to move other cameras.
  He thought that the zooming actions were missed placed. In his meaning zooming action gave the TV producer opinion about the characters in the show.
  He thought that some of the camera angles were not perfect.

Interview 2

Name: Masahiro Shibata
Position: Associate Director
Experience from production: 2 years work for video production in a local station
Specialty: Database, Information retrieval, Natural language processing
Date:
Material used: CNN Crossfire [Debate Over the State of the Union Aired January 28, 2003 - 19:00 ET]
Interviewee time: 30 min
Person’s comments: Over all impression was that he “Enjoyed the show”. The points that he thought could be improved was:
  He thought that going from one type of group shot to the same type of group shot, but with other characters was strange. For example going from a 3 shot on (A B C) to a 3 shot on (B C D).
  He thought it was strange when there was a very fast series of speaker switches that resulted in some delayed camera switches.
Interview 3

Name: Masaru Takechi
Position: Research Engineer
Experience from production: Limited
Specialty: Digital TV Receiver System Architecture Development
Date: 2003-02-19
Material used: CNN Capital Gang (5 person show) [Bush Administration Sets Scene for Iraq War; Bush Previews State of Union Address; Interview With Steve Sabol. Aired January 25, 2003 - 19:00 ET]
Interviewee time: 45 min
Person's comments: The over all impression was “pretty good”. Points that he comment was the following:

When there was a speaker switch there were some delays in the camera switch and he liked that, and said that it gave him a very natural feeling.
He disliked the camera switching when there was very fast switching of speakers, he thought the camera clipping was to fast at that time and took away the attention from the context of the show.
He would have liked bigger differences in the voices.

Interview 4

Name: Yuko Yamanouchi
Position: Research Engineer
Experience from production: 2 years at NHK News Technical Center
Specialty: Virtual Studio System, Image processing
Date: 2003-02-20
Material used: CNN Capital Gang (5 person show) [Bush Administration Sets Scene for Iraq War; Bush Previews State of Union Address; Interview With Steve Sabol. Aired January 25, 2003 - 19:00 ET]
Interviewee time: 40 min
Person's comments: Her overall impression was “almost comfortable”, while she was looking at the show she also tested to readjust the advanced settings. The comments she made about the software was that she:

She thought that there were too many group shots, there was no problem with the length of them but she thought they appeared too frequently.
She would like to have some gesture of the character.
Interview 5
Name: Konishi Hirokazo
Position: Research Engineer
Experience from production: 2 years and 6 months (system operator at the News center)
Specialty: Digital TV Receiver System Architecture Development
Date: 2003-02-20
Material used: Interviewee with Salman Rushdie (2 person show) [http://www.salon.com/06/features/interview2.html]
Interviewee time: 45
Person’s comments: His overall impression was “Comfortable”. He also played around with the settings in the advanced settings menu. The things that he commented were the following:
- He thought that the camera switching was natural for the guest (non dominant speaker) but quite unnatural for the host. He would like to have more two shots when the host was talking and more one-shots when the guest was talking.
- He thought that there was too many big group shots.
- He thought the zooming and panning action was strange in some places, he would like to have better coordination when to use them.
- He liked the non speaker shots when the frequent speaker was speaking for a long time.

Interview 6
Name: Kinji Matsumura
Position: Research Engineer
Experience from production: 2 years and 6 month as cameraman and video switcher
Specialty: Data Broadcasting
Date: 2003-02-21
Material used: Interviewee with Sadman Rushdie (2 person show) [http://www.salon.com/06/features/interview2.html]
Interviewee time: 30 min
Person’s comments: Over all impression was good. His comments about the TV show were the following:
- He would have liked to have more one shots on the guest
- He would like to have different types of two-shots, he would have preferred two shots from the side instead of from the front.
- He would like to have more non speaker shots, especially instead of two shots.
- He thought the frequency was ok.
- He would like to have slower zooming.