Automatic mashup generation of multiple-camera videos

Automatic Mashup Generation of Multiple-camera Videos

The work described in this thesis was carried out at the Philips Research Laboratories in Eindhoven, The Netherlands. It was financially supported by the Dutch BSIK research program MultimediaN.

ISBN: 978-90-74445-89-4

Cover design by Paul Verspaget

Automatic Mashup Generation of Multiple-camera Videos

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus, prof.dr.ir. C.J. van Duijn, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op donderdag 19 november 2009 om 16.00 uur

door

Prarthana Shrestha

geboren te Kathmandu, Nepal

Dit proefschrift is goedgekeurd door de promotor:

prof.dr. E.H.L. Aarts

Copromotoren:
dr. M. Barbieri
en
dr. J. Weda

Contents

1

Introduction

The popularity of non-professional videos has grown along with the technical developments in video recording and sharing. Until 1980s, video recording was a complex technique used by television studios and advanced amateurs. Then a portable consumer device, the *camcorder*, was introduced into the mass market, with an embedded audio and video storing facility on a tape. Starting from the early 2000, the tapes in the camcorders are being replaced by devices like optical disks, hard disks and flash memories. Similarly, the embedded video processing techniques such as automatic color adjustment, shakiness correction have made it possible to improve the signal quality of a video recording. Camcorders have become cheaper in price, smaller in size, richer in functionality and more accessible to wider population. In addition to camcorders, video cameras are now widely available in compact digital still cameras and mobile phones.

Along with the development in recording technologies, video sharing has become fast and easy. The most common medium for sharing a video used to be via a physical storage device because of the large size of the video data and the limited bandwidth provided by the Internet and mobile networks. Presently, with the development of video compression technologies, increase in broadband internet connections and popularity of video sharing websites such as YouTube, started in 2005, it is possible not only to share but also to publish the videos.

At present, video recording and sharing has become more accessible and pop-

ular among general users than ever before. It has become a common sight at social occasions like parties, concerts, weddings, vacations that many people are shooting videos around the same time. Furthermore, many places like lecture halls, meeting rooms, playgrounds, theaters, amusement parks are equipped with cameras. If we search in YouTube for a musical performance by a popular artist, multiple hits are returned containing different non-professional videos captured at the same performance. For example, the search phrase "nothing else matters london metallica 2009" submitted to YouTube on date 08-08-2009 returned 18 user-captured clips from the same performance ranging from 35 sec to 8 min.

The availability of multiple-camera recordings captured simultaneously provide coverage of the same time and event from different viewing angles, however, it does not provide in itself a richer user experience. Watching all these recordings individually takes a long time and likely to become boring due to the limited viewing angle of a camera and similarity in the content. Furthermore, non-professional recordings are likely to contain ill-lit, unstable and ill-framed images as the recordings are captured spontaneously, generally by hand-held cameras, with insufficient light and unpredictable subjects. Such recordings with poor signal quality becomes unappealing to a general audience [Lienhart, 1999].

In professional video productions, recordings are made using multiple cameras and the most appropriate segments from the different recordings are selected and combined during editing to enhance the visual experience. A well established set of rules for video aesthetics, also called *film grammar*, is widely used to create the desired effect in the final video. However, in the case of non-professional recordings, the multiple-camera recordings are rarely edited. The popular software tools for editing multiple camera videos are Adobe Premiere, Ulead, iMovie, etc. These editing tools are considered too time-consuming and too complex and are thus rarely used by an average user. Figure 1.1 shows a screen shot of an interface of Adobe Premier with five audio and video streams during editing. A complex time-line representation of the recordings, too many buttons and very technical terminologies used in the interface limit these tools only to experts and advanced amateurs.

This thesis presents an approach to automatically select and combine segments from concurrent multiple-camera recordings resulting in a single video stream, called *mashup*. Unlike a summary, which represents a temporally condensed form of a recordings, a mashup represents different camera views interleaved in a single stream. The presence of multiple views adds visual dynamics in a mashup, which reduces monotony of viewing angle. Similarly, the signal quality of a mashup can be raised by selecting higher quality segments from the available recordings.

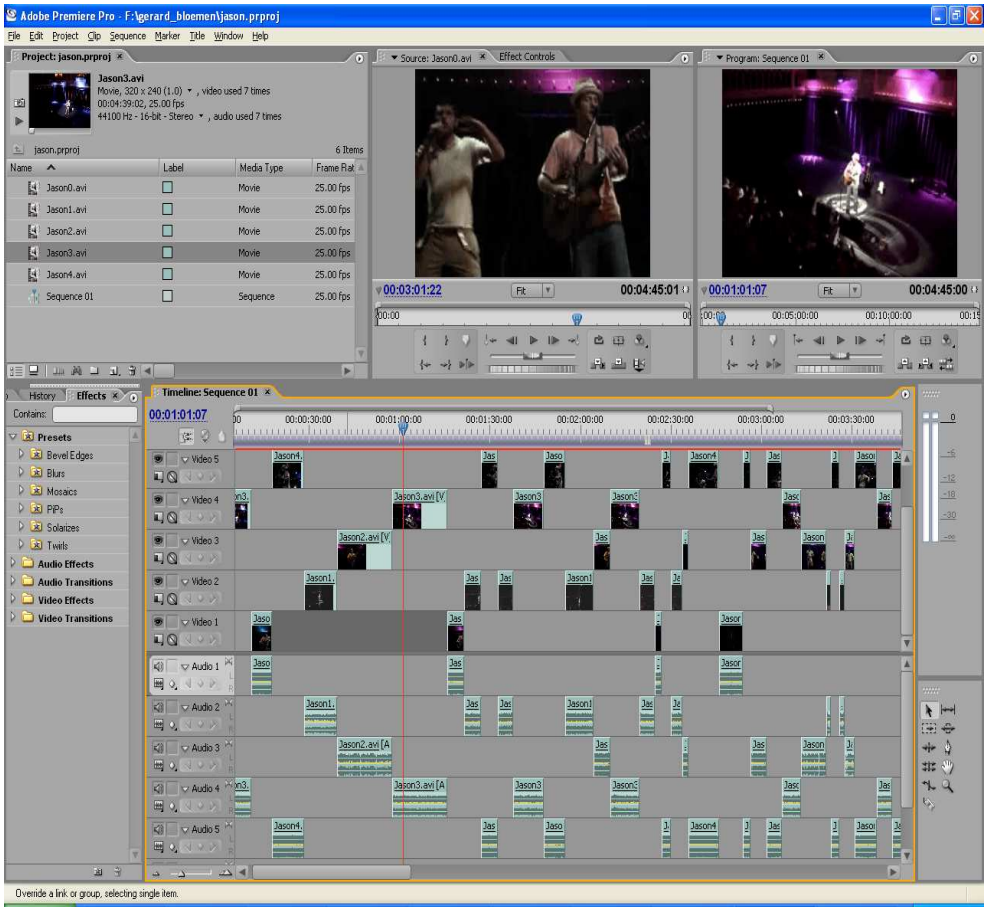


Figure 1.1. Screen shot of an interface of Adobe Premiere Pro, an editing tool for multiple-camera recordings, while editing audio-visual streams from five cameras.

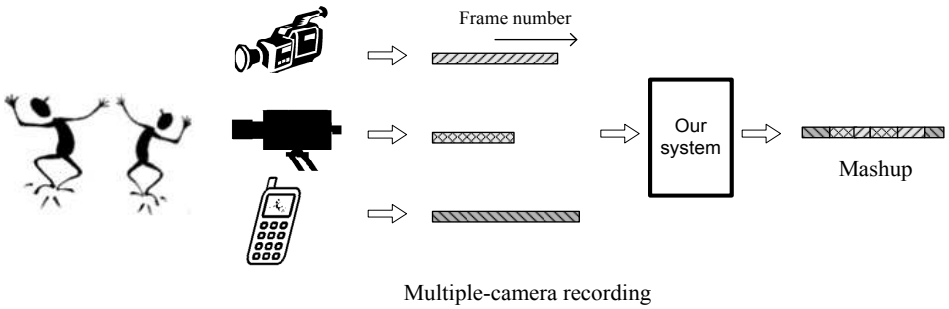


Figure 1.2. Illustration of a mashup generation system using concurrent recordings from different capturing devices.

1.1 Objectives

The objective of this thesis is to design an automatic system that creates a mashup video from concurrent multiple-camera recordings captured by non-professional users for enriching the video experience. Figure 1.2 illustrates the composition of a mashup video from three recordings from different cameras having different durations. The resulting mashup generated by our system contains interleaved segments from the given recordings. The system can be used by amateur videographers to enhance their personal recording or general video audience to experience multiple-camera recordings. In this thesis, *user* is meant for non-professionals who have access to multiple-camera recordings and like to combine the contents from the different recordings.

1.2 Research questions

The thesis focuses on designing a system that generates mashups from multiple-camera recordings captured by non-professionals. The research questions involved in designing such a system are:

1. *What are the requirements for generating a mashup?*
   Multiple cameras are used at different occasions like wedding, parties and sports. A mashup is applicable in the recordings from such occasions, however, the requirements may be different for different applications. Therefore, the first research question is what are the requirements for generating a mashup.
2. *How can the requirements be addressed by a mashup generating system?*
   Given a list of requirements, a system for generating mashup should address the requirements. So the question is how to formalize each of the requirements and model a mashup generation system.

3. *How can a mashup that satisfies the requirements be generated?*
   A mashup generation system should be able to generate mashups while satisfying the requirements as defined by the formal model. The appropriate video segments from the multiple-camera recordings should be selected that best fulfills the requirements. The research question is how to utilize the quantitative measure of the degree of fulfillment of a requirement.
4. *How can the generated mashups be evaluated?*
   The automatically generated mashup should be evaluated to measure if the original requirements are fulfilled both subjectively and objectively. An objective evaluation shows the mashup quality according to the formal model and the subjective evaluation shows the perceived quality of the mashups. The research question is how to evaluate the mashup quality both subjectively and objectively.

1.3 Related work

In this section, we describe work on mixing videos from multiple-camera recordings based on our literature research. Since the use of multiple-cameras is growing for different applications, the recordings are used for different purposes. We broadly classified the works into four categories based on their purpose: *video summarization, object reconstruction, event recording*, and *collaborative production*.

1.3.1 Video summarization

A video summary is a temporally condensed video, which presents the most important information from concurrent multiple-camera recordings. The applications include surveillance, monitoring, broadcasting and home videos.

A summarization method for multiple-camera videos is presented in [Hata, Hirose & Tanaka, 2000] for surveillance system covering a wide area, such as a university building, containing a number of cameras. The different cameras capture sparse recordings, which is difficult to understand. In order to summarize the entire state of an event, the video scenes are first evaluated and assigned an importance score according to the presence of the objects of importance, such as humans, buildings and cars in space, time and the relationships among the objects. Then the high scoring scenes from the recordings are displayed with a map and objects by three dimensional graphics.

[Hirano, Shintani, Nakao, Ohta, Kaneda & Haga, 2007] describe a multiple-camera recording and editing system, called *COSS-MC*, designed for nursery schools and kindergarten. The system controls simultaneous capturing from fixed cameras, provides an interface for manually editing and mixing videos for each child and distributes the edited video by a streaming server. While editing, the

school teachers can choose video fragments (up to 5 min long), which are displayed as thumbnails in multiple columns, representing multiple cameras. The teachers select the fragments from different cameras along the time and the system generates a single video by interleaving the selected fragments. A distribution server streams the video to authorized people, typically to the parents, and accepts comments.

1.3.2 Object reconstruction

Multiple perspectives available from different concurrent recordings provide information to re-create a scene with more details than a single camera recording. The applications include panorama creation, wide screen movies and 3D reconstruction.

[Sawhney, Hsu & Kumar, 1998] present a method to create seamless mosaics, planar and spherical, using inexpensive cameras and relatively free hand motions. The frames from different cameras are aligned using consistent estimation of alignment parameters that map each frame to a consistent mosaic coordinate system. As a similar application, [Greene, 2009] reports a near-real time technology, developed at Microsoft, which seamlessly stitches together videos taken at a certain location, from different mobile phone cameras. The technology uses location information from mobile phones and image recognition algorithms resulting in a higher resolution video stream.

An approach for 3D reconstruction of a dynamic event using multiple-camera recordings is presented in [Sinha & Pollefeys, 2004]. The motion of the silhouettes of an object is used for extracting the geometrical properties of the object and the cameras. This results in the calibration and synchronization of the cameras and 3D reconstruction of the moving object.

1.3.3 Event recording

It is now getting common that lecture halls, meeting rooms and theaters are equipped with video cameras. The recordings are used for archiving, analysis and publication. The multiple camera recordings are also being used for different research purposes, for example, to study human interactions during meetings [Carletta, 2007] and to introduce computers into the human interaction loop in a non-obtrusive way [Waibel & Et. al., 2004]. While the individual cameras provide a monotonous view, mixing recordings from different cameras provide a wider coverage and a more informative record of the event.

[Lampi, Kopf, Benz & Effelsberg, 2008] describe a *Virtual camera system*, which uses multiple-cameras for recording and broadcasting classroom lectures. The classroom is equipped with cameras and Wi-Fi access points, and both the lecturer and students communicate to the system using PDAs and PCs. During a

lecture, an active camera is pointed at the lecturer's face. A student wishing to ask a question informs the system via PDA or notebook PC. The system identifies the location of the student using Wi-Fi information and forwards the request to the lecturer via a pop-up window. When the lecturer accepts the request, another camera at the student's side also becomes active. Only one camera is chosen for recording at a time, based on the audio-level and motion detection. The recording cameras also follow some cinematic rules such as appropriate shot duration and overview shot after two or three close up shots.

[Stanislav, 2004] proposes an algorithm for automatic video editing of meetings recorded by multiple-cameras. An importance score is calculated for all the recordings based on participant's activity (speaking, motion of head and hands), participant's visibility (position of head, image brightness). To avoid quick camera changes, a minimum duration for a camera is imposed once it is selected.

1.3.4 Collaborative production

The concurrent multiple-camera recordings captured by individual users can be viewed as a collaborative effort. The following works facilitate the use of such recordings to produce a video.

Collaborative video capturing by mobile phones for live video jockeying (VJing) is presented in [Engström, Esbjörnsson & Juhlin, 2008]. The authors proposed the *SwarmCam* system which allows club visitors to transmit videos to a central database simultaneously during capture by mobile phones. A VJ can preview all the incoming videos in the SwarmCam mixer display. As long as the user keeps capturing, the VJ can apply effects on the videos using real time image processing tools. The processed videos are then combined with each other or with other materials, the same way a DJ mixes between two discs, using a hardware mixer. The research is aimed at mobile users to become content creators and isolated VJs to communicate with the audience for an appealing club atmosphere.

[Cremer & Cook, 2009] present an interface for mixing multiple recordings captured by different users in a musical performance. The work is intended to provide an alternative source of income for artists by providing a channel to leverage user generated content. First, an audio-fingerprinting technology, proposed in [Haitsma & Kalker, 2002], is applied to align different recordings. Then the aligned recordings are presented in a user interface, where the users can select video segments from different recordings. A separate audio stream, an official recording or a high quality user recording, is added to the final video.

1.3.5 Discussion

The research on mixing audio-visual content from concurrent multiple-camera recordings is growing with the increase in the use of multiple cameras. In the

application area of meeting rooms, a corpus of annotated audio-visual data has been created for research purposes as described in [Carletta, 2007] and [Stanford, Garofolo, Galibert, Michel & Laprun, 2003]. Similarly, the MPEG-4 community on multi-view coding for 3D displays and free-view TV is standardizing the use of multiple-camera recordings [MPEG, 2008]. The different systems and applications described in this section utilize the availability of content to re-create an event or enrich an experience. However, a common theme missing from all these works is the evaluation of their approach. The authors claim that their system, prototype or method is working, however, its performance in a real world setting or from the point of view of the user is missing.

In the described related work, the multiple-camera recordings are analyzed and applied differently for different applications. No prior work is found on automatic mashup generation from multiple cameras captured by non-professional users, where the environment is uncontrolled and there are no constraints on the number and movement of cameras. Therefore, the related works provide an overview of the possible applications involving multiple-cameras and solution approaches, however, they cannot be used for a comparative evaluation of our research on mashup generation.

1.4 Research approach

Since our research is aimed at enriching the video experience of the users, we followed a user centric approach to design the automatic mashup generation system. First, we conducted a study to find the target application and to elicit the requirements involving general camera users, video-editing experts, design students and multimedia researchers. We selected musical concerts as the application for our automatic mashup generation system. An additional advantage of using concert recordings is that they are available in huge numbers in online web archives like YouTube and Google Video. Then based on the elicited requirements , we designed a formal model to generate a mashup by maximizing the degree of fulfillment of some requirements. The degree of fulfillment of a requirement is derived from the analysis of different audio-video features. Next an algorithm was designed to efficiently compose the mashup by selecting the clips which best satisfy the requirements. Finally, the resulting mashup was evaluated in terms of mashup quality, objectively and subjectively, with respect to the mashups generated by naive and manual approaches.

1.5 Thesis organization

The remainder of the thesis is structured as follows. In Chapter 2, we describe the methodologies followed to define the target application and to elicit requirements

for a mashup generation system. In Chapter 3, we present a formal model to generate a mashup based on the requirements. We define the concepts and requirements and present our solution for generating automatic mashups based on an optimization approach. In Chapter 4, we propose an automated approach to synchronize multiple-camera recordings based on detecting and matching audio and video features extracted from the recorded content. We describe three realizations of the approach and assess them experimentally on a common data set. In Chapter 5, we present the audio-visual feature analysis techniques used to measure the degree of fulfillment of the requirements. In Chapter 6, we propose an algorithm to compose a mashup from a given synchronized multiple-camera recording by selecting clips that best satisfy the requirements. We measure the performance of the proposed algorithm in terms of mashup quality and compare it with the mashups generated by two other methods. In Chapter 7, we describe the subjective evaluation of the mashup quality by means of a user study. Finally, in Chapter 8, we present our conclusions and suggestions for future work.

1.6 Thesis contributions

The thesis addresses the research questions described in Section 1.2. The main contributions of the research described in this thesis are:

- Techniques for automatic synchronization of multiple-camera recordings using audio-visual features.
- A formal model for mashup generation and an algorithm to automatically create mashups of multiple-camera recordings.
- A methodology to validate mashup generating algorithms by means of a user study.

2

Mashup target application and requirements

Capturing videos with multiple cameras is popular in different social occasions like wedding, parties and travels. The content and the purpose of recordings from such occasions are different. Consequently, the requirements for generating mashups are also different. Therefore, we conducted a study to understand the usage of multiple-camera recordings, to find the target application for further research and to elicit requirements. The study involved professional video-editors, amateur camera users, and researchers at Philips Research working in the area of multimedia applications. In this chapter, we describe the method followed to select the target application and present the requirements for mashup generation.

This chapter is organized as follows. Section 2.1 describes the interviews conducted with the professional video-editors with the tips and techniques followed in capturing and editing multiple-camera recordings. Section 2.2 describes the focus group study using three different application scenarios and the selection of the target application. Finally, in Section 2.3, we present a list of requirements to be addressed while generating mashup for the selected application.

2.1 Interviews with experts

As a first step in our explorative study, we interviewed three professional video-editors on the usage of multiple-camera recordings. The goal of the interviews was to learn the practical aspects of multiple-camera recording and editing, which can be useful for eliciting the requirements for generating the mashup.

All three of the interviewed experts were associated with Philips Research, Eindhoven, The Netherlands. They had been working since more than 10 years on video shooting and editing, especially documentaries, home or personal videos, and wedding videos. They generally worked with single camera recordings but and they were experienced also with the multiple-camera recordings.

The interviews were conducted individually in a semi-structured way to explore the topic. We prepared a set of points such as shooting techniques with multiple cameras, editing process, the most time consuming and the most difficult task in editing, which we would like to know about. Then during the interview, the experts explained the process of editing and the rules they apply for video editing in general and in the case of multiple-cameras in particular. Meanwhile the questions were asked to cover the prepared set of points.

The interview gave us insights in the available commercial tools and shooting and editing tips regarding multiple-camera videos. The three experts agreed that shooting and editing multiple-camera recordings is very time consuming and costly compared to single-camera recordings. Shooting with more than one camera requires planning and communication among cameramen. Before editing, the recordings should be synchronized very precisely. Then segments from the different recordings are chosen according to the personal or artistic preference of the editor.

The editors were positive about the availability of different functionalities in the software tools, such as color filters and special effects, however, they were not in favor of automatic video editing systems. They considered video editing as a medium of expression of their creativity and they would prefer to have tools to accomplish it. In present home video editing tools, they find that the most time-consuming and uncreative task is the synchronization of multiple-recordings in a common time-line before editing. A remark representative to all the participants was ‘I want to have a tool for automatic synchronization of the multiple-camera recordings and the rest I will do the rest of the editing by myself.’

The experts were aware of the film grammar, a set of aesthetic rules such as appropriate length of a shot and smooth transition between scenes, generally followed while making videos. However, all of them said that they do not follow the rules consciously while editing. They were in favor of spontaneity and following their ‘mood’ and ‘gut feeling’. Our impression was that there is a general consensus on

some basic editing rules, which may not require conscious decisions. The knowledge on shooting and editing videos acquired during the interview corresponds to the established film grammar rules, which are available in the literature. However, the interviews with the experts helped us to identify the rules used most widely in practice and how the rules are supported by the existing editing tools. Below are the tips from the experts on multiple camera recordings organized in four categories based on their usage.

2.1.1 Synchronization

- During shooting, once the multiple-cameras are turned on, they are kept on as long as the shooting continues or as allowed by the camera battery or memory. A recording needs to be synchronized with other recordings from multiple-cameras every time it is turned on and off.
- The recordings are synchronized in a common time-line before editing.

2.1.2 Content uniformity

- The cameras involved in the shooting are calibrated to set a uniform white-balance. The different color settings in the cameras may produce visibly different colors so that seamless mixing of recordings becomes difficult.
- Audio is recorded from a fixed location near the source, and not by the moving cameras. Otherwise, the audio around the different cameras may be different and a uniform audio quality throughout the edited video cannot be maintained.

2.1.3 Editing

- The recordings are watched repeatedly to prepare a mental map of the final edited video.
- During editing, segments are selected from the recordings based on their signal quality (stability, sharpness, brightness, motion) and their artistic or emotional value.
- The selected segments are trimmed and combined. The position of the segments are set after testing their suitability in different locations.
- The videos from different cameras are sometimes displayed as picture-in-picture or in split screens to provide more information in limited time and space.
- While segmenting the videos, frames are carefully chosen as cut points to avoid abrupt cuts. Generally, the cut points are selected when an action seems to be complete.

- The duration of the segments are chosen based on the content, for example if the audio to be associated with the video is fast, the segments durations are kept short to match with the audio. If a segment is too short, it is difficult to understand and it becomes visually annoying to watch a sequence of such short segments. On the other hand, if a segment is too long it becomes monotonous and boring. In the case of non-professional recordings, the minimum and maximum length of the clips are generally set in between 3 sec and 12 sec respectively, depending on the content.
- If the edited video is going to contain music, a music track is generally chosen that matches the video content, for example romantic songs in wedding videos. Then the video cuts are aligned with changes in audio characteristics like tempo. If the segments retain their original audio, the audio is normalized to have the same volume throughout the video.

2.1.4 Post-processing

- Segments are glued together usually with a hard-cut and sometimes with special effects like dissolve and fade.
- Semantically meaningful or matching music, or voice-over is added to the video.

2.2 Focus group study

Since there are several possible applications involving non-professional multiple-camera recordings, we wanted to determine a target application, which addresses an existing user need. The application should also be scientifically challenging and should be feasible to build within the available time and resources. To interact with users we followed a group interview approach, called *focus group*, used in qualitative research to test new ideas, acquire information about user needs and opinions. The focus groups allows interaction among different users and builds opinions on one another's responses, which is considered more productive method for idea generation than a one-on-one interview. The discussions were guided by a moderator to maintain focus on the topic and involve all the participants.

To initiate discussions in the focus group, we used *scenarios*, short stories written on envisioned real life situations concerning people and their activities. The participants expressed their personal opinions on the scenarios, discussed its relevance in their lives, suggested modifications and proposed alternative scenarios. The study was conducted in three groups involving 18 subjects in total. The first group consisted of seven researchers on video processing systems from Philips Research, the second group consisted of seven industrial-design students from Eindhoven Technical University, and the third group consisted of four amateur video-

editors. All the participants had captured video at least once but none had formal education on video capturing or editing. Only one participant in the third group had experience with multiple-camera recordings. The three groups were selected to receive ideas and views from potential users on the usage of multiple-camera recordings captured by non-professionals.

The focus group meetings were held at the High Tech Campus in January, 2007. In the focus group meeting, the participants were introduced with the concept of the mashup generation using multiple-camera recordings. Next, a scenario, printed on paper, was given to the participants to read and write their first thoughts about it. Then the participants discussed their opinions on the scenario and suggested improvements or alternative scenarios. The discussions were guided by a facilitator.

There were three scenarios presented in the meeting, that were used to discuss about the applications of non-professional video mashups. In the presented scenarios, the term 'summary' is used instead of 'mashup' because mashup is sometimes used to refer to mixing different media items such as overlaying images on a online map, or in a photo pasting a face of a person to somebody else's body.

Figures 2.1, 2.2 and 2.3 present the *Wedding editor*, *Recollection* and *Family album* scenarios, respectively, used in the focus group. The Wedding editor scenario was received well by the participants as they had experienced problems similar to the one described in the scenario. They see a big market for such an application, and suggested additional features such as segmenting the videos according to events such as dance, church and dinner and according to faces for people to choose what they would like to have in a video. However, some amateurs video-editors were against the idea of using an automatic system on personal content. A typical remark was 'I can not trust a program to create my video'. The Recollection scenario was considered as an interesting application but the participants had doubts about maintaining the privacy of the people who would not like to be recorded. In all the groups the participants said they would like to keep their videos irrespective of low image quality, as long as the persons are recognizable. The Family album scenario was found to be useful by the group of researchers and video-editors who considered themselves as potential users of such a system. They suggested to include functions like easy browsing of the videos and modifying the automatic mashup. However, the group of students were against the idea of sharing their content with parents. They prefer to have their own videos and mashups independent of the parent's content. They suggested alternative scenarios on mixing and sharing mashups of concerts and party videos with friends. The different application scenarios and analysis of the focus group results are published in [Shrestha, Weda & Barbieri, 2007a].

During the focus group discussions, the 'concert music videos' was suggested

Wedding editor

Frank got married in a church. It was a standard one-hour long ceremony consisting of speeches, songs, and wedding vows. After the church they went for a reception followed by a party in a community hall. Three of his friends are recording videos of the daylong event independently from different angles and locations. Many other friends and relatives were taking photos and recording short videos on mobile phones.

Representative image for the Wedding editor scenario.

Most of the attendees would like to have a combined and summarized video of the events of the day. So at the end of the party, the three cameramen friends and attendees who shot videos during the day leave a copy of their content to the Wedding Editor system, available as a service from the party hall. The system synchronizes the available recordings and the time overlapping content is identified. The order of the events is preserved and still pictures or other relevant pictures from public domain like maps, and pictures of party hall are added to the summary.

Most of the attendees accept the standard half hour summary, containing high signal quality segments covering most of the event. However, some have different requirements. For example, Frank would like to have the video with wide coverage without any time restriction. His friends would like to have even shorter version containing only the best quality pictures. The Wedding editor generates different versions of summaries based on the given criteria. The summaries can be copied to physical memory devices like DVD, CD, USB drive or accessed via internet.

Figure 2.1. The Wedding editor scenario presented to the focus group.

Recollection

The family of Sally went to Disneyland with two other friends and their children. Each family had a video camera and they captured a couple of hours of video during the day. Additionally, there were some video and still picture cameras from the park like on the roller coaster, and tour to Neverland.

Representative image for the Recollection scenario.

At the end of the day each of the families would like to have combined summary videos from all the available recordings. They prefer to keep all the material captured by their individual camera plus the interesting ones from their friends and from the park.

The Recollection service at the park accepts the recordings made by Sally and her friends. It also obtains videos and pictures from the park camera recordings that contains them and their children. The quality of the personal videos is enhanced and the interesting parts of the friends' videos and park videos are included in the summary. Sally copies the summary video on her cameras memory, while her friends decide to download it online.

Figure 2.2. The Recollection scenario presented to the focus group.

Family album

John goes on a holiday in Turkey with his wife. Their children went to a summer camp in Spain. Together the family has about 8 hours of video recording after summer. John would like to show their summer trips to his parents, but the recording is just too long to view.

Representative image for the Family album scenario.

John has a subscription for a Family album web service. It provides storage and sharing of videos for authorized people like a common family space. The service also provides a facility to generate summaries from the videos. John logs into the service and creates a 15 minute video summary containing his trip and that of his children. Then he saves the summary video to his family web space, which can be accessed by his parents.

Figure 2.3. The Family album scenario presented to the focus group.

by the three groups as an interesting scenario. All the participants were aware of the availability of a large number of multiple-camera concert videos by amateurs in YouTube. They could identify themselves as a user of such a system, which generates a mashup from concurrent concert recordings. The group of researchers considered the concert videos as a relatively novel application domain. The group of students were enthusiastic about using concert video mashups to publish in their blogs and enhancing their personal concert video collections. The group of video amateurs considered it as a useful application considering a very large number of recordings involved where manual editing is extremely time consuming. Therefore, as a promising application area, we selected the application domain of personal/home videos captured during concerts.

2.3 Requirements for concert video mashups

After the interviews with experts and focus group meetings, we decided to focus our research on recordings from musical concerts. The recordings are captured concurrently by multiple cameras typically in the audience, generally using hand-held cameras. The length of the recordings varies from a couple of seconds to a couple of minutes, usually containing a song or a segment of it. In a typical concert recording, the main audio content is music and the video content are views of the audience and the stage.

A concert video mashup is aimed to enrich the video experience. The requirements for a concert videos mashup were compiled based on the user opinions that came up during the focus group meetings and interviews with the professional video-editors reported in Section 2.1. Furthermore, we also consulted literatures like [Zettl, 2004] on film grammar rules and [Reeves & Nass, 1996] on user perception of media. The literatures helped on understanding the techniques used in videos and their effect on audience. The following paragraphs present a list of requirements for a concert video mashup.

Requirement 2.1 (Synchronization). The audio and visual streams used in a mashup should be time continuous. The time delay between audio and video causes lip sync problems and a delay between two consecutive videos causes either repetition or gaps. Therefore, for a complete and smooth coverage of the concert, it is required to align the multiple-camera recordings in a common time-line. The requirement corresponds to the experts' opinion in Section 2.1.1.

Requirement 2.2 (Image quality). A good signal quality video is desirable for clarity and pleasure of watching. Since the non-professional concert videos are generally shot by hand-held cameras, under insufficient lighting, it is difficult to continuously capture a high quality recording. A desired a high quality mashup can

be achieved by selecting good quality segments from the multiple-camera recordings. The requirement corresponds to the experts' opinion in Section 2.1.3.

Requirement 2.3 (Diversity). A mashup should offer variety in terms of its visual content for providing a wide coverage of the concert and a dynamic video experience. If the visual content in two consecutive mashup segments is similar, a mashup can not produce the diversity effect. For example, where two cameras are close to each other and have the same field of view. Therefore, consecutive video segments in a mashup should come from different cameras containing different content. The requirement originates from the focus group meeting with the design students.

Requirement 2.4 (User preference). A user may have different personal preferences over different recordings. For example, when a user wants to enhance his own recording by using segments from other recordings, he may prefer to have more of his own recording in the mashup. Therefore, a mashup should contain more segments from the preferred recordings. This requirement stems from the participants' wish expressed during focus group meetings to have more control on the mashup generation.

Requirement 2.5 (Suitable cut point). In video editing, cuts are made at some particular instants so that the video is perceived as seamless and aesthetically pleasing. Similarly, in mashups, camera switches should be made at appropriate points to avoid disturbing changes. The requirement corresponds to the experts' opinion in Section 2.1.3.

Requirement 2.6 (Suitable clip duration). If a video segment is very short it is difficult to understand and it becomes visually annoying to watch a sequence of such short segments. On the other hand, if it is too long it passes over the attention span of the viewer. Therefore the video segments in a mashup video should be limited within a minimum and maximum time duration. This requirement originates from the experts' opinion in Section 2.1.3.

Requirement 2.7 (Completeness). In general, a concert can be better covered by a larger number of cameras because they provide multiple perspectives and capture more information. Since a user chooses the recordings to generate a mashup, it is natural to expect them to appear in the mashup. Therefore, it is required that all recordings should be represented in a mashup. The requirement originates from the focus group meeting with the design students and video amateurs.

Requirement 2.8 (Special Effects). Different effects such as transitions and picture-in-picture display are used to enhance the aesthetics and to make optimal use of the available time and space. For example, fade in/out to represent the

long time difference between two segments to be combined, close up of a face as picture-in-picture if the background video is not crowded. Therefore, a mashup should be composed with matching special effects. The requirement corresponds to the experts' opinion in Section 2.1.3.

Requirement 2.9 (Audio). Audio is an essential part of a video, especially in musical concerts. The video and audio in a mashup should always be synchronized to maintain lip sync. To achieve a good and consistent quality, audio from a single source with high quality should be used as the main audio source. The audio associated with the selected video can be added to give a background or ambiance, like cheering of a crowd. The requirement corresponds to the experts' opinion in Section 2.1.2.

Requirement 2.10 (Semantics). A concert video is considered more desirable if the audio and the video content matches the context, such as close-up view of an artist while singing, faces of audience while cheering. These features add information and meaning to the content. Therefore a mashup video should contain segments based on semantic information. The requirement corresponds to the experts' opinion of Section 2.1.3.

Requirement 2.11 (Color balance). The recordings from different cameras may look different in color, even if they contain the same object at the same time, due to camera quality and settings. A mashup from such recordings may look patchy and distracting. Therefore, color synchronization is required to normalize the color appearance of the mashup. The requirement corresponds to the experts' opinion in Section 2.1.2.

Requirement 2.12 (Editing interface). Since perception of a video is subjective, is it very unlikely that a mashup can meet all the needs of a user. Therefore an intuitive user interface is required for simple modifications and application of personal touches to an automatically generated mashup. The interface should allow users to perform simple editing tasks such as visualizing the recordings, changing the segments selected in the automatic mashup with segments from other recordings, adding and deleting segments and adding text or other effects. The requirement was elicited during the focus group meeting with the design students and researchers.

2.4 Conclusions

This chapter describes our explorative study to gain insights into the expert's view on multiple-camera recordings, to select the target application and to elicit requirements. The study included interviews with the experts, focus group meetings with potential mashup users such as students, video amateurs and multimedia re-

searchers. Based on the studies we selected musical concert as a target application and compiled 12 requirements for generating the mashup.

During focus group discussions, different requirements were suggested for generating mashups for different scenarios. Here we present some examples where the requirements were different from the musical concerts. In the case of home-videos, involving friends and families, users were less concerned about the image quality and their general opinion was that if a person's face is recognizable the video is acceptable. However, in the case of concert videos, which may come from unknown sources, users demanded high quality of the recordings. In all the three scenarios presented, two main user requirements were summarization and user control. Since the recordings may last for hours, users would like to have a summary of the recordings and have more control over the process. The synchronization was not recognized as a requirement in these scenarios. However, in concert recordings users wanted to see the mashup as happened in the live event.

In the following chapters we will focus our work on generating mashups from multiple camera concert recordings captured by non-professional users based on the requirements compiled in this chapter. In the next chapter, a solution approach will be proposed to satisfy the requirements. Based on the proposed approach the requirements will be formally defined and applied on the algorithm for generating mashups. Finally, the generated mashups will be evaluated by users to measure how the requirements are satisfied.

3

Formal model

In the previous chapter we elicited requirements for generating mashups from concert recordings captured by non-professional users. The requirements were compiled regardless of whether they can be incorporated in an automatic system. In this chapter, we present a system that addresses all the requirements and we further define our focus with respect of the automatic generation of mashups. We translate the requirements into computable elements and present a formal model for mashup generation.

The rest of the chapter is organized as follows. We introduce an overview of the proposed approach for mashup generation in Section 3.1. We define the concepts and elements to be used in the formal model in Section 3.2. Section 3.3 and 3.4 describe the components of the proposed system for generating automatic mashups. We present a formal definition of the mashup generation problem in Section 3.5 and finally present the solution approach of the problem in Section 3.6.

3.1 Overview

The requirements compiled in Section 2.3 provide insights for generating a mashup, such that if a system can generate a mashup that addresses the requirements then the mashup will be perceived by the users as of high quality. We propose an approach for generating such a mashup from concert recordings, captured

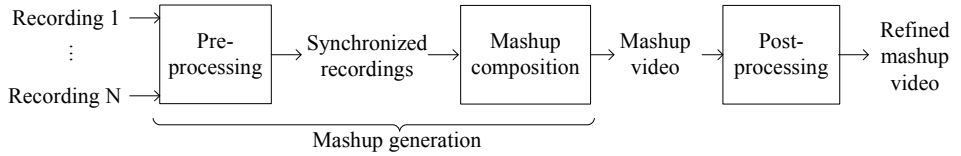


Figure 3.1. Overview of the proposed system for mashup generation and post-processing.

by non-professional users, which aims to fulfill the compiled requirements. The proposed approach consists of three processing steps, where each step addresses certain requirements that help in addressing other requirements in successive steps. The three steps are: *pre-processing, mashup composition* and *post-processing*. Figure 3.1 shows the schematic representation of the proposed approach.

At the first step, *pre-processing*, we compute the temporal relationship among the given multiple-camera recordings. The step is aimed to satisfy Requirement 2.1 (synchronization).

In the next step, *mashup composition*, a mashup video is generated from the synchronized recordings which addresses the Requirements 2.2 (image quality), 2.3 (diversity), 2.4 (user preferences), 2.5 (cut-point suitability), 2.6 (suitable clip duration), 2.7 (completeness) and 2.10 (suitable semantics). The resulting mashup video is a single video stream containing segments from the given recordings. The pre-processing and mashup composition steps are referred together as *mashup generation* in the thesis.

At the third step, *post-processing*, the generated mashup is further processed to address the Requirements 2.8 (special effect), 2.9 (audio), 2.11 (color balance) and 2.12 (editing interface). At this stage, a mashup video is refined to give a seamless effect such that the audio-visual discrepancies caused by the clips from different recordings are removed. The mashup can also be customized according to the personal taste of a user by allowing user interaction with the mashup. The post-processing step is not covered in this thesis and some of its requirements will be discussed as future work in Chapter 8.

In the following sections we provide formal definitions of the concepts used in the thesis and model the proposed mashup generation approach.

3.2 Definitions

In this section we formally define the concepts, such as video stream, recording, camera-take, which are used frequently in the thesis. Since these concepts are used informally in different contexts, sometimes even interchangeably, the formal definitions are given to avoid ambiguity in their use in the rest of the chapters. For

ease of reference, Table 3.1 at page 38 provides a complete list of the symbols used in the formal mashup generation approach.

Definition 3.1 (Video stream). A *video stream* is a sequence of video frames captured continuously by a capturing device from the moment the device starts capturing to the moment it stops. A video stream v containing n number of video frames f^v is represented as:

$$v = (f_1^v, \ldots, f_n^v) \; . \qquad \square$$

Definition 3.2 (Audio stream). An *audio stream* is a sequence of audio frames captured continuously by an audio capturing device from the moment the device starts capturing to the moment it stops. An audio stream a containing n number of audio frames f^a is represented as:

$$a = (f_1^a, \ldots, f_n^a) \; . \qquad \square$$

Definition 3.3 (Camera-take). A *camera-take* τ is a couple (v,a), where v is a video stream and a is an audio stream captured concurrently at the same occasion. $\square$

The audio and video streams may be captured by a single device like a camcorder or any device with video and/or audio capture capabilities. If a device captures only an audio stream or a video stream, then the missing element in the camera-take is substituted by an empty stream.

Definition 3.4 (Recording). A *recording* R is a non-empty sequence of camera-takes captured by a single device during an event.

$$R = (\tau_1, \ldots, \tau_m), \qquad m \geq 1 \; . \qquad \square$$

The camera-takes in a recording are non-overlapping in time.

Definition 3.5 (Multiple-camera recording). A *multiple-camera recording* $\overline{R}$ is a set of recordings made during an event using different capturing devices. Each element of the set represents a recording. A multiple-camera recording is represented as:

$$\overline{R} = \begin{pmatrix} R_1 \\ R_2 \\ . \\ . \\ . \\ R_N \end{pmatrix}, \qquad N \geq 2 \; . \qquad \square$$

Definition 3.6 (Video segment). A *video segment* s^v consists of a sequence of frames from a video stream v. A video segment is defined as:

$$s^v = (f_x^v, \ldots, f_y^v), \qquad 1 \leq x \leq y \leq |v| \; . \qquad \square$$

Definition 3.7 (Audio segment). An *audio segment* s^a consists of a sequence of frames from an audio stream a. An audio segment is defined as:

$$s^a = (f_x^a, \ldots, f_y^a), \qquad 1 \leq x \leq y \leq |a| \; . \qquad \square$$

Since recordings in a multiple-camera recording are captured at the same occasion, the video and audio segments from different recordings may have been captured at the same time. These segments are called *overlapping segments*. The overlapping audio and video segments may contain similar or sometimes the same audio and video content, respectively.

Definition 3.8 (Video frame rate). The *video frame rate* r^v of a video stream is the number of frames captured per time unit. A typical frame rate for a standard digital video camera in Europe is 25 frames per second. $\square$

Definition 3.9 (Duration). The *duration* of a video or an audio segment is obtained by dividing the total number of video or audio frames by the video or audio frame rate, respectively. For example, the duration $d(s^v)$ of a video segment s^v containing a first frame x and a last frame y is given by:

$$d(s^v) = \frac{y - x + 1}{r^v} \; . \qquad \square$$

Definition 3.10 (Clip). A *clip* S is a couple (s^v, s^a), where s^v and s^a are segments from a video stream and an audio stream, respectively, captured concurrently at the same occasion and with the same duration. $\square$

The audio and video segments of a clip may be captured by a single device like a camcorder or by separate devices like audio with an external microphone and video with a camera. If only a video or an audio segment is available, then the missing segment in the clip is substituted by an empty stream. In a multiple-camera recording, clips having overlapping video or audio segments are called *overlapping clips*.

Since both audio and video segments in a clip have the same duration, the clip duration is given by the duration of its audio or the video segment:

$$d(S) = d(s^v) \; .$$

Definition 3.11 (Universal time). The *universal time* of a frame is an instant referring to the continuous physical time. Frames captured at the same instant by multiple cameras should refer to the same universal time. The universal time of a video or an audio frame is given by the functions $t_u(f^v)$ and $t_u(f^a)$, respectively. $\square$

Definition 3.12 (Recording time). The *recording time* of a frame is a time instant referring to the first frame in the recording. The recording time of a video or an audio frame is given by the functions $t_r(f^v)$ and $t_r(f^a)$, respectively. For example, assuming one camera-take in a recording the recording time of a video frame f_v^x is given by:

$$t_r(f_x^v) = \frac{x}{r^v} \ . \qquad \square$$

The recording time starts when the first frame of a recording is captured. The total duration of a recording is equal to the recording time of the last frame of the recording. Figure 3.2 illustrates a recording according to its recording time.

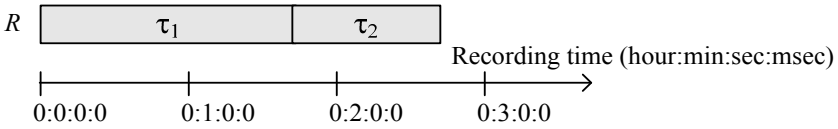


Figure 3.2. Recording time of a recording R containing two camera-takes.

Definition 3.13 (Camera time). The *camera time* of a frame is the capture time of an audio or a video frame according to the internal clock of the capturing device. The camera time of a video or an audio frame is given by the functions $t_c(f^v)$ and $t_c(f^a)$, respectively. $\square$

The camera time may be embedded in the video frames. Generally, the internal clock of a capturing device is set manually according to the universal time. In highly professional settings 'jam-sync' devices are used for precise clock setting of multiple capturing devices. The camera time provides information about the duration of a camera-take and also the time-interval between two camera-takes. Figure 3.3 illustrates an example recording according to its camera time.

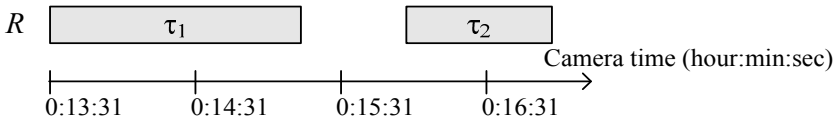


Figure 3.3. Representation of the recording R of Figure 3.2 according to its camera time.

Definition 3.14 (Common time). The *common time* of a frame is a time instant referring to an audio or a video frame with respect to the first frame captured in a multiple-camera recording. The common time of a video or an audio frame is given by the functions $t_s(f^v)$ and $t_s(f^a)$, respectively. $\square$

The common time starts when a recording starts capturing the first frame of a multiple-camera recording and continues until the last frame of the multiple camera

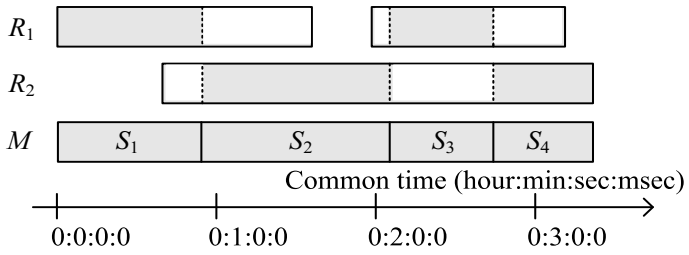


Figure 3.4. Representation of a mashup M from two partially overlapping recordings R_1 and R_2. The mashup clips, $S_1 - S_4$, selected from R_1 and R_2 are represented by gray areas.

recording. Therefore, the reprentation of a multiple-camera recording according to the common time allows the visualization of overlapping clips. Figure 3.5 shows a multiple-camera recording containing two recordings according to the recording time and Figure 3.6 shows the same multiple-camera recording according to the common time. The corresponding frames of two overlapping clips have the same common time.

Definition 3.15 (Mashup). A *mashup* M is a sequence of non-overlapping clips from a multiple-camera recording.

$$M = (S_1, \ldots, S_l) \ , \tag{3.1}$$

where l is the total number of clips and

$$\exists R_j, R_{j+1}, \forall S \in M : S_i \in R_j, S_{i+1} \notin R_j \ . \qquad \square$$

Two consecutive clips in a mashup are selected from different recordings in a multiple-camera recording. An example mashup created by two recordings in a common time-line is visualized in Figure 3.4.

The duration of a mashup is determined by the sum of the durations of the individual clips:

$$d(M) = \sum_{i=1}^{l} d(S_i) \ .$$

3.3 Pre-processing (Synchronization)

In order to satisfy Requirement 2.1(synchronization), the recordings in a multiple-camera recording should be represented in a common time-line. However, the available time information in the recordings, camera time and recording time, are based on the individual capturing devices and are most likely to be different for every device. Therefore, *synchronization* involves finding the time displacement

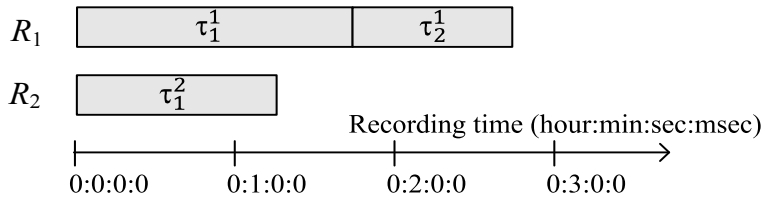


Figure 3.5. A multiple-camera recording, according to the recording time, before synchronization. No time-gap between the camera takes τ_1^1 and τ_2^1 is considered. The synchronization time-offset between the recordings R_1 and R_2 is unknown.

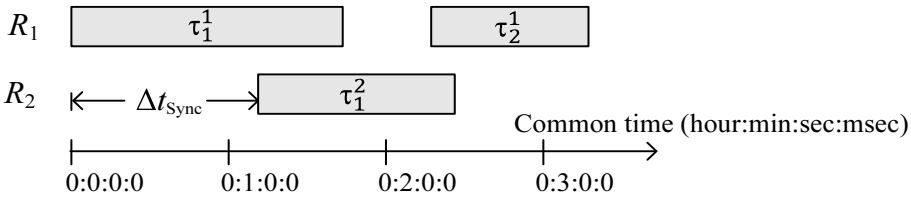


Figure 3.6. Multiple-camera recording of Figure 3.5 after synchronization. The recordings including the time gap between the camera takes and the synchronization time-offset Δt_{sync} are represented in a common time.

among the recordings. The time-offset between two recordings is given by the synchronization offset-time Δt_{Sync}. Figures 3.5 and 3.6 show a multiple-camera recording containing two recordings before and after synchronization, respectively.

The following paragraphs present a formal description of the synchronization problem. The descriptions are based on two video streams, however, they are applicable to more than two video or audio streams.

If we consider two recordings R_1 and R_2, the video streams are given by:

$$\begin{aligned} v &= (f_1^v, \ldots, f_n^v), v \in R_1 \ , \\ v' &= (f_1^{v'}, \ldots, f_{n'}^{v'}), v' \in R_2 \ . \end{aligned}$$

If the camera times of the frames, given by $t_c(f_i^v)$ and $t_c(f_j^{v'})$, refer to the same instance in the universal time, then the video streams v and v' are called *perfectly synchronized*. The synchronization time-offset Δt_{Sync} between the video frames is given by:

$$\Delta t_{\text{sync}} = t_c(f_i^v) - t_c(f_j^{v'}) \ : \ t_u(f_i^v) = t_u(f_j^{v'}) \ . \tag{3.2}$$

The synchronization time-offset between the two recordings can also be calculated using the recording times of the frames in Equation 3.2, instead of the camera times, if each of the recordings contains only one camera-take. However, if there

are multiple camera-takes the time interval between two camera-takes cannot be determined from the recording time. Therefore, the Δt_{Sync} calculated for a pair of camera-takes is not directly applicable to other camera-takes in the recordings. In this case, the synchronization offset time Δt_{Sync} should be calculated separately for each pair of camera-takes in the recordings.

The universal time is continuous and both the recording time and camera time represent a sampled time referring to the instant when a frame is captured. Therefore, it is highly unlikely that frames from two different recordings are perfectly synchronized. In practice, we compute the synchronization time such that the difference in the universal times between two synchronized frames is minimized, i.e. $|t_u(f_i^v) - t_u(f_j^{v'})|$ is minimal for the synchronized frames $f_j^{v'}$ and f_i^v.

In this thesis we propose an approach to automatically synchronize the multiple-camera recordings by detecting and matching audio and video features extracted from the recorded content. The synchronization between two recordings is verified by carefully listening and watching the synchronized recordings played simultaneously. Even a slight inaccuracy in the synchronization offset, for example by 4 video frames, causes echo in audio and motion delay in video. The synchronization methods and their performance in a common data-set are presented in Chapter 4.

3.4 Mashup composition

The mashup composition problem consists of selecting clips from a synchronized multiple-camera recording, while satisfying the set of requirements as described in the mashup generation approach in Section 3.1. Since the requirements represent user preferences, the perceived quality of a mashup depends on how well the requirements are satisfied.

The mashup composition problem can be solved by using different approaches. The main approaches are: rule-based and optimization. In the first approach, described in [Russell & Norvig, 2002], rule bases are developed, which imitate the mashup composition procedure followed by an expert. This approach is used in artificial intelligence applications, for example, in an expert system to help doctors choosing the correct diagnosis based on a number of symptoms. In the case of mashup composition, this approach can be applied by writing rules for determining if a requirement is satisfied or not and defining the order in which the requirements should be satisfied. For example, if a candidate clip satisfies the requirement diversity, then check if the requirement image quality is satisfied else discard the candidate clip.

In the optimization based approach, the degree of fulfillment of the requirements are represented by numeric values computed by corresponding functions.

The approach aims to maximize the degree of fulfillment of all the requirements. This approach is popular in different application areas including video summarization [Campanella, 2009], [Barbieri, 2007]. In the case of mashup composition, since there are multiple requirements, an objective function should be defined, which combines the different functions corresponding to the requirements. The value of the objective function corresponds to the higher the level of satisfaction of the requirements or the mashup quality. Therefore, during mashup composition the objective function should be maximized.

The rule based approach is useful in applications where rules can be established from the available domain knowledge. However, in the case of mashup composition, the requirements represent user preferences rather than strict rules and the degree of fulfillment of the requirements effects the quality of a mashup. Therefore, we selected the optimization based method to solve the mashup composition problem. There are some additional benefits of using optimization based approach. In an objective function, the functions corresponding to different requirements can be combined with variable weights. So if we want to assign different priority to different requirements, we can simply assign new weights without changing the functions. Another advantage of using an objective function is that it provides a numeric value to a mashup, corresponding to the degree of fulfillment of the requirements, which represents the objective quality of the mashup.

In order to apply the optimization approach, we defined an objective function that combines the different functions providing the degree of fulfillment of the requirements: Requirements 2.2 (image quality), 2.3 (diversity), 2.4 (user preferences), 2.5 (suitable cut-point) and 2.10 (suitable semantics). Since Requirements 2.6 (suitable clip duration), and 2.7 (completeness) are strict requirements, they are measured in a binary scale representing if the requirements are met or not. These requirements are called *constraints*, and applied as a condition to the objective function that should be fulfilled in a mashup. There might be cases of multiple-camera recordings where it is impossible to satisfy the constraints and an optimal mashup cannot be generated. We will discuss these limitations while modeling the Requirements 2.6 (suitable clip duration), and 2.7 (completeness) in Sections 3.4.6 and 3.4.7, respectively.

3.4.1 Objective function

An *objective function* is designed to estimate the overall quality of a mashup based on how well the given requirements are fulfilled, such that the mashup quality can be maximized. The objective function $\mathrm{MS}(M)$ of a mashup M, called *mashup score*, depends on the following functions: image quality score $Q(M)$, diversity score $\delta(M)$, user preference score $U(M)$, cut-point suitability score $C(M)$, and semantics suitability score $\lambda(M)$ corresponding to Requirements 2.2 (image qual-

ity), 2.3 (diversity), 2.4 (user preference), 2.5 (suitable cut-point) and 2.10 (suitable semantics), respectively.

$$\mathrm{MS}(M) = \mathcal{F}(Q(M), \delta(M), U(M), C(M), \lambda(M)) \; . \tag{3.3}$$

The requirements addressed in the objective function influence the quality of the mashup but their priority order and effectiveness are not known. Therefore, we used a simple linear approach, also used in earlier cases [Barbieri, 2007], to combine the functions $Q(M)$, $\delta(M)$, $U(M)$, $C(M)$ and $\lambda(M)$ in the objective function. The objective function can be formalized as:

$$\mathrm{MS}(M) = a_1 Q(M) + a_2 \delta(M) + a_3 U(M) + a_4 C(M) + a_5 \lambda(M) \; . \tag{3.4}$$

The coefficients a_1, a_2, a_3, a_4 and a_5 are used to weigh the contributions of the different requirements. They allow flexible generation of the mashups by changing the weights of the requirements. The effect of different weights on the mashup quality will be discussed in Chapter 6. The following paragraphs describe the mashup composition problem by modeling the requirements that are included as constraints and included in the objective function.

3.4.2 Modeling Requirement 2.2 (Image quality)

A good image quality is desirable in a video stream for ease of understanding and pleasure of watching. The image quality of a video segment can be determined by analyzing different low-level video features in a frame, such as brightness, blur, and between frames such as motion. The image quality of a frame is given by a function $q(f^v) \rightarrow [0,1]$. For a video segment, the image quality $Q(s^v)$ is represented as the mean quality of the frames present in the segment:

$$Q(s^v) = \frac{1}{y-x+1} \sum_{i=x}^{y} q(f_i^v) \; .$$

The image quality score of a mashup is given by the mean quality value of the clips as:

$$Q(M) = \frac{1}{l} \sum_{i=1}^{l} Q(S_i) \; . \tag{3.5}$$

The extraction of visual features and quality estimation of a video frame is presented in Section 5.1.

3.4.3 Modeling Requirement 2.3 (Diversity)

According to Requirement 2.3, a mashup should contain diverse visual information from the multiple recordings. For example, if a clip contains a close up view of an artist and the two candidates for a successive clip contain a view towards the same artist and a view towards the audience, then to add diversity the candidate clip

with the audience view should be selected. The diversity in a mashup increases the information content and enriches the visual experience.

The diversity in a mashup is modeled by the visual distance between two consecutive clips: $\delta(S_i, S_{i+1})$ .The diversity score of a mashup, $\delta(M)$ is calculated as:

$$\delta(M) = \frac{1}{l-1} \sum_{i=1}^{l-1} \delta(S_i, S_{i+1}) \ . \tag{3.6}$$

The visual distance between two clips is measured based on the difference in features, such as brightness, color, texture. The extraction of features and visual distance calculation between two video segments is described in Section 5.3.

3.4.4 Modeling Requirement 2.4 (User preference)

If a user has different preferences for the different recordings in a multiple-camera recording, for example one of the recordings was self-made, the user may like the mashup to contain more clips from the preferred recording. Therefore, users can provide their preference score to each of the recordings. The higher the preference score of a recording, the higher the likelihood that a clip from that recording is chosen in the mashup. However, if no preference is given by a user, all the recordings in a multiple-camera recording are assigned a uniform score.

If the preference score of a recording in a multiple-camera recording is given by a function, $u(R) \rightarrow [0,1]$, the preference score of a clip $U(S)$ corresponds to the score of the recording.

$$\forall S \in R_j, \ U(S) = u(R_j) \ .$$

We model the preference score of a mashup by the mean of the preference scores of its clips:

$$U(M) = \frac{1}{l} \sum_{i=1}^{l} U(S_i) \ . \tag{3.7}$$

3.4.5 Modeling Requirement 2.5 (Suitable cut point)

In professional music video editing, the video streams captured from different cameras are cut into segments according to their visual content and the change in audio tempo. Cuts made at suitable times create an aesthetically pleasing transition among segments. For example, if a cut is made during a camera motion the viewer perceives it as an abrupt break. Therefore, in a mashup the clips should be cut in appropriate instants to give smooth transitions among the different recordings.

The cut-point suitability of a frame is given by a function $\theta(f^v) \rightarrow [0,1]$ based on the analysis of audio and visual content, where a higher value of $\theta(f^v)$ indicates higher degree of suitability as a cut-point. For a video clip, the cut-point suitability score is calculated by averaging the suitability scores corresponding to its first and

the last frame. If f_x^v and f_y^v are the first and last frames of a clip S, the cut-point suitability score is computed as:

$$C(S) = \frac{\theta(f_x^v) + \theta(f_y^v)}{2} .$$

The cut point suitability score of a mashup by the mean of the scores of its clips:

$$C(M) = \frac{1}{l} \sum_{i=1}^{l} C(S_i) . \tag{3.8}$$

The cut-point suitability of a video frame is determined by extracting audio and visual features from the recordings and analyzing their changes. The feature extraction method and their analysis are presented in Section 5.2.

3.4.6 Modeling Requirement 2.6 (Suitable clip duration)

A video segment becomes incomprehensible if it is too short and becomes boring if it is too long, which is a basic rule followed commonly in video editing. In professional music videos, the duration of a clip depends on the music genre, for example for fast beat music the clips can be shorter than a second while for slow music the clips can be about 12 sec. Therefore, according to Requirement 2.6, the clips in a mashup video should be longer than a minimum value ($d_{\min}$) and shorter than a maximum value ($d_{\max}$). The values of $d_{\min}$ and $d_{\max}$ are adapted to the audio genre. The requirement is modeled as:

$$\forall S_i \in M : d_{\min} \leq d(S_i) \leq d_{\max} . \tag{3.9}$$

As described in Section 3.4, the suitable clip duration is applied as a constraint, a requirement that must be fulfilled for a mashup composition. However, in some cases the requirement may not be possible to fulfill such as, when there is only one recording available for longer duration than given by the maximum clip duration ($d_{\max}$). Therefore, the requirement is applied in generating mashup only when there is more than one recordings available.

3.4.7 Modeling Requirement 2.7 (Completeness)

According to Requirement 2.7, a mashup requires to include clips from all the synchronized recordings of a multiple-camera recording. If there are N recordings in a multiple-camera recording, the completeness requirement is modeled as:

$$\forall j \in [1,\ldots,N], \exists S_i \in M : S_i \in R_j . \tag{3.10}$$

From the users' perspective, it is understandable that they want all the recordings in a multiple-camera recording be represented in the mashup. However, in some cases it may not be possible to fulfill this requirement. One case is when there are two very short recordings of length $d_{\max}$ present at the same time such

that both are available only for a single clip. Another case is when there are too many clips to include all in the available time for the mashup. In case this requirement is not satisfied the user could be asked for input or the system could continue without satisfying the requirement.

3.4.8 Modeling Requirement 2.6 (Suitable semantics)

In professional video productions, audio and video segments are presented based on their semantics. For example, when there is a guitar solo in the audio, the video is focused on the guitar or the guitarist and cheering in audio is accompanied by video of the crowd. Semantically matching content together in audio and video conveys a clear and coherent message to the audience.

We defined concepts in audio domain such as guitar, solo, cheering, silence, and in video domain such as stage, guitarist, singer, audience. Each concept in the audio domain is linked to the concepts in the video domain, where the strength of the link is based on their semantic match. For example, an audio concept guitar is linked to a video concept guitarist by a higher value of the strength of the link than to a video concept audience.

If the semantic match in a clip is measured by $\lambda(S)$, the semantic suitability score in a mashup is given by:

$$\lambda(M) = \frac{1}{l}\sum_{i=1}^{l}\lambda(S_i)\ . \tag{3.11}$$

3.5 Problem definition

In the previous sections, we defined the concepts and the requirements for the automated mashup generation system and proposed a formal model based on optimization approach to satisfy the requirements. In this section, we define the mashup generation problem that will be addressed in the rest of the thesis.

Definition 3.16 (Mashup generation problem). Given a multiple-camera recording $\overline{R}$, synchronize the recordings and compose a mashup M, that maximizes the objective function and satisfies the constraints. □

The mathematical model of the mashup-composition problem can be given as:

$$\text{maximize} \quad \text{MS}(M) = a_1 Q(M) + a_2 C(M) + a_3 \delta(M) + a_4 U(M) + a_5 \lambda(M)\ , \tag{3.12}$$

$$\text{subject to} \quad \forall S \in M : d_{\min} \leq d(S_i) \leq d_{\max}\ , \tag{3.13}$$

$$\forall j \in [1,\ldots,N], \exists S_i \in M : S_i \in R_j\ , \tag{3.14}$$

where coefficients a_1, a_2, a_3, a_4 and a_5 are used to weigh the contributions of the different requirements. Equation 3.12 shows the objective function to be maximized and Equations 3.13 and 3.14 represent the constraints suitable-clip dura-

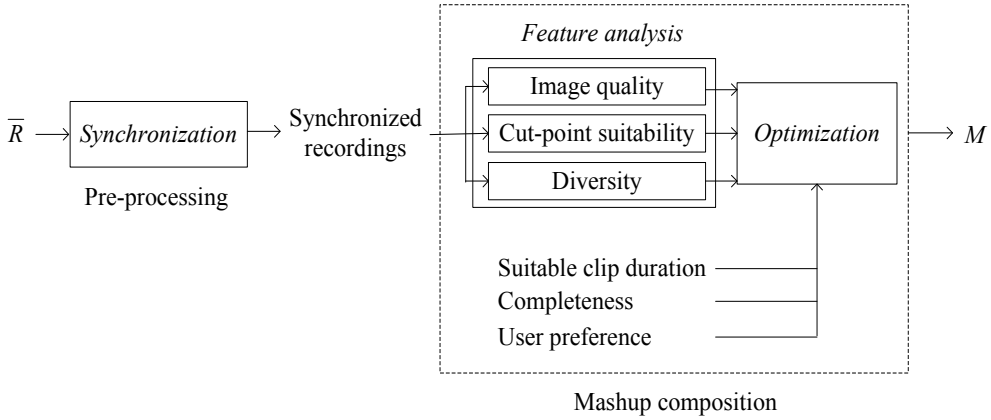


Figure 3.7. Schematic representation of the proposed solution for the mashup generation problem. The processes indicated in italics are described in separate chapters under the same name.

tion and completeness, respectively. The following section proposes a solution approach to solve the mashup generation problem.

For ease of reference, Table 3.1 at page 38 gives an overview of the symbols used in this chapter to define the mashup generation problem.

3.6 Proposed solution

In Section 3.1, we presented an overview of the proposed approach for a mashup generation system consisting of pre-processing and mashup composition steps. The requirements involved in the steps are modeled in the Sections 3.3 and 3.4. The mashup generation problem has been modeled as an optimization problem, where some requirements are treated as constraints and others as maximization parameters. The model describes the requirements in an abstract and generic way. To obtain a solution for a mashup generation problem, we further specify and implement the requirements in the following chapters. The schematic representation of the solution approach is presented in Figure 3.7.

The first step, *synchronization*, consists of fulfilling the Requirement 2.1 described in Section 3.3. The audio-visual content in the recordings is used to find the time offset between the recordings. If a recording fails to be synchronized, it is filtered out from the multiple-camera recording employed for generating a mashup. The detailed description of the synchronization methods and their performance is described in Chapter 4.

In the next step, *feature-analysis*, audio and video features are extracted and

analyzed, providing numeric values for the functions given in Equation 3.5 (image quality), Equation 3.6 (diversity) and Equation 3.8 (cut-point suitability). The description of the feature extraction and their analysis is presented in Chapter 5. As for the requirement on suitable semantics, described in Section 3.4.8 advanced audio and video content analysis techniques are required. We tested available software, explained in [Breebaart & McKinney, 2003], to detect audio concepts such as noise, music, silence but the results were not reliable enough for being applied in our solution. The state of the art techniques used for audio-video concept detection in concerts, such as [Snoek, Worring, Smeulders & Freiburg, 2007], [Naci & Hanjalic, 2007], show the problem is content dependent and they do not cover all the desirable concepts. Due to the non-availability of a reliable solution and our time limitation, the semantic suitability requirement is not implemented in our mashup composition solution.

In the final step, *optimization*, we develop an objective function based on Equation 3.4 and an algorithm that evaluates and selects the clips to be included in a mashup by maximizing the value of the objective function (Equation 3.4) while satisfying the constraints. Its description and an objective evaluation of the mashup quality is presented in Chapter 6.

Table 3.1. Symbols used in the formalization of the mashup generation problem.

symbol	*description*	*page*
a	audio stream	25
$C(M)$	cut-point suitability score of a mashup	32
$C(S)$	cut-point suitability score of a clip	34
$d_{\max}$	maximum clip duration	34
$d_{\min}$	minimum clip duration	34
$d(s^v)$	duration of a video segment	26
$d(M)$	mashup duration	28
f^a	audio frame	25
f^v	video frame	25
l	number of clips in a mashup	28
M	mashup	28
$\mathrm{MS}(M)$	objective function to maximize	32
N	number of recordings in a multiple-camera recording	25
n	number of frames	25
$Q(M)$	image quality score of a mashup	32
$Q(S)$	image quality score of a clip	32
$q(f^v)$	image quality of a video frame	32
R	recording	25
$\overline{R}$	multiple-camera recording	25
r^v	video frame rate	26
S	clip	26
s^a	audio segment	26
s^v	video segment	26
$t_c(f^v)$	camera time	27
$t_r(f^v)$	recording time	27
$t_s(f^v)$	common time	27
$t_u(f^v)$	universal time	26
$U(M)$	preference score of a mashup	32
$U(S)$	diversity score of a clip	33
$u(R)$	user preference score of a recording	33
v	video stream	25
$\theta(f^v)$	cut-point suitability score of a frame	34
$\delta(M)$	diversity score of a mashup	32
$\delta(S_i, S_{i+1})$	diversity score of a clip	33
Δt_{Sync}	synchronization offset time	28
τ	camera-take	25
$\lambda(M)$	semantic suitability score of a mashup	35
$\lambda(S)$	semantic suitability score of a clip	35

4

Synchronization

According to the proposed solution approach for the mashup generation problem in Chapter 3, the multiple-camera recordings need to be synchronized before composing a mashup. Figure 4.1 highlights the synchronization step in the proposed solution approach. The synchronization requirement is formalized in Section 3.3. In this chapter, we propose a novel automated approach to synchronize multiple-

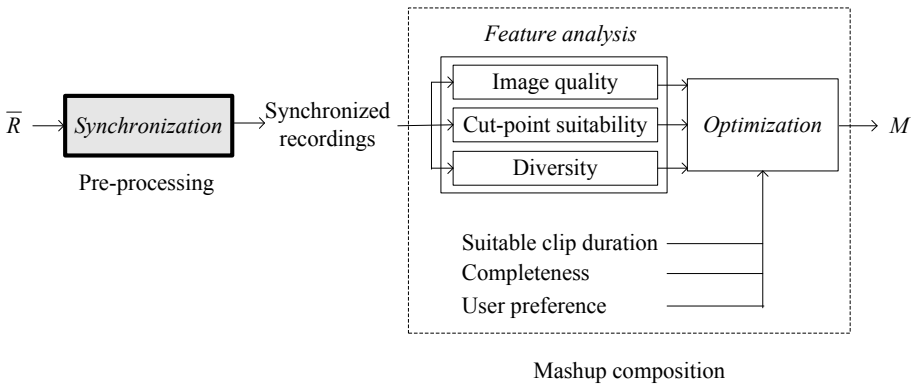


Figure 4.1. Schematic representation of the proposed mashup generation problem, in which the synchronization step is highlighted.

camera recordings based on detecting and matching audio and video features extracted from the recorded content. We develop three realizations of the approach and assess them experimentally on a common data set. The synchronization techniques described in this chapter are applicable not only to mashup generation but also to other applications that require synchronizing of multiple-camera recordings.

The rest of the chapter is organized as follows. We introduce the synchronization problem in Section 4.1 including possible application areas and related work. In Section 4.2, we describe the basic principles of our proposed synchronization approach. In Sections 4.3, 4.4 and 4.5 we develop three realizations of our approach based on still-camera flashes, audio-fingerprints and audio-onsets, respectively. The sections describe methods of extracting audio and video features and matching them to compute the synchronization offset between recordings of a multiple-camera recording. We assess the three realizations experimentally on a common data set in Section 4.6. Discussions regarding the usability of the proposed realizations in view of practical use cases are provided in Section 4.7. In Section 4.8, we present our conclusions.

4.1 Introduction

In professional video productions, the use of multiple-cameras is very common. The synchronization of the recordings is ensured by physically connecting the cameras to a device called "jam-sync" or by the use of a "clap" at each camera-take. The clap produces and strong impact sound, which introduces a distinct signal in all the audio recordings, which can be used as a time reference to synchronize the recordings. The jam-sync sets the internal clock of all the connected cameras exactly to the same time. However, in the case of non-professional user recordings, control and coordination among different cameras are not feasible. The cameras are set independently and turned on and off at the will of their users.

Currently, in video-editing involving multiple-camera recordings the synchronization point is searched manually. The search is considered very time consuming and difficult, as expressed by the video-editing experts in the interview presented in Section 2.1.1. It involves finding an instant in all the available recordings with a distinctive object motion or sound, for example, the sound of a clap or a dance action. Then further careful observation is required to accurately locate the synchronization point, for example when the audio-signal shows a peak due to the clap or when a dancer just touches the floor.

Synchronization requires high precision because even a slight misalignment between two videos results in time discontinuity in the mashup. For example, Figure 4.2 shows two camera recordings, captured at a frame rate of 25 frames per second, from a dance event out of synchronization by 0.40 sec (10 frames). If they

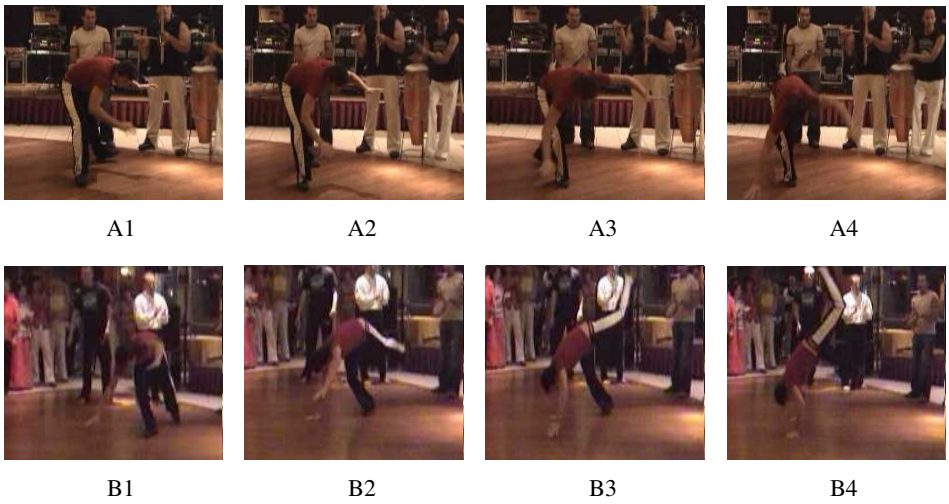


Figure 4.2. The two rows of images A1-A4 and B1-B4 represent two different camera recordings, where camera A is ahead of camera B by 0.40 sec (10 frames). The images are taken from every fourth frame of each recording. If a video is created as a combination of frames B1, B2, A3, A4, the same motion is repeated twice, creating a visual hiccup.

are combined by alternating the videos along the time, upon each camera change, the dancer's movement is either repeated or missed by 10 frames. Similarly, the misalignment between audio and video of different cameras causes lip-sync problems. According to [BT.1359-1, 1998] editing multiple recordings without losing lip synchronization requires the audio and video to be aligned in the range of +45 msec to -125 msec, where a positive value indicates that sound is advanced with respect to vision. The required high precision for synchronization cannot be obtained by a manual setting of camera times because in devices like amateur cameras and camcorders, the time setting option for the clock is usually given in the resolution hour:minute:seconds, while the required accuracy is in the level of milliseconds. Therefore, if the frame rate of a recording is 25 frames per second, the synchronization should be accurate by ± 1 frame to maintain lip-sync in a mashup.

4.1.1 Other applications

Many other applications involving multiple cameras use synchronization, which is generally done manually. We present some of the applications that can benefit from automatic synchronization.

Video editing tools such as Adobe Premiere Pro, Final Cut Pro, Ulead, which

currently require manual synchronization, could offer automatic solutions in their existing products. In stereo or multi-view coding standards, where synchronization is listed as a requirement [JTC1/SC29/WG11-N9163, 2007], automatic synchronization could be applicable to 3D video and free-viewpoint video technologies. Another useful application of automated synchronization would be for large video repositories like YouTube, Google, Yahoo! that contain multiple clips of an event recorded by different people. Presently, the clips are not organized according to an event and each of the clips can only be accessed individually. Automated synchronization of multiple clips available from an event facilitates easy managing and simultaneous watching of the clips. Other applications, such as annotating official-meeting videos [Michel & Stanford, 2006], [Carletta, 2007], improving audio quality using multiple recordings [O'Shaughnessy, Kabal, Bernardi & Others, 1990], which use multiple recordings captured simultaneously in a room and require synchronizing the recordings, can also benefit from automated synchronization.

4.1.2 Related work

Our work on synchronizing non-professional recordings from multiple cameras is closely related to the works presented in [Cremer & Cook, 2009] and [Kennedy & Naaman, 2009], where audio-fingerprinting techniques have been used to synchronize recordings from musical performances captured around the same time. In [Cremer & Cook, 2009], the authors have applied the same fingerprinting method as in this chapter, which was originally proposed in [Haitsma & Kalker, 2002]. The synchronization is used for visualizing multiple-camera recordings in an editing system. Similarly, in [Kennedy & Naaman, 2009] another audio-fingerprinting method is used to synchronize multiple recordings for organizing a large database of community-contributed collections of concert videos. In this chapter, we develop three realizations of a synchronization approach using flashes in video, audio-fingerprinting and audio-onsets and assess them experimentally on a common data set. We also provide recommendations on the usability of the different realizations in practical use cases including video editing and organizing large databases.

Synchronization of videos captured in controlled setups has been a research topic for the purpose of object tracking, 3D-scene rendering, multi-view coding and multi-sensor fusion. The known synchronization methods are based on multi-view geometry, which uses geometrical properties between the camera and the object features being captured such as points and lines.

The synchronization work in [Stein, 1998] assumes the cameras are static and images are related by a homography (i.e. any given point in one figure corresponds to one and only one point in another figure and vice versa such that there is no paral-

lax effect). A similar assumption is made in [Caspi, Simakov & Irani, 2004], where synchronization is achieved by minimizing the sum of squared differences between the sequences. The method presented in [Lei & Yang, 2005] depends on tracking a line feature in multiple videos. This method allows limited camera motion, requires at least three cameras and the cameras should have identical frame rates. In [Whitehead, Laganire & Bose, 2005] moving features are computed from the recordings that best agree with the pre-computed camera geometries from stationary image features. The method is applicable independent of camera frame rates, however, there should be at least three cameras that remain stationary throughout the video capture process and, furthermore, moving objects should have sufficient texture for tracking.

In addition to the above described methods, other state of the art papers on synchronization [Caspi & Irani, 2002], [Yan & Kankanhalli, 2002], [Sinha & Pollefeys, 2004], [Tuytelaars & Gool, 2004] are intended for a controlled environment and have limitations on the number of cameras, camera movement, object features, frame rates, etc. So they are not applicable in case of non-professional videos where the settings, quality, movement and number of cameras are non-uniform. In our synchronization approach, we use audio and visual features extracted from the recordings that provide an abstract and time accurate representation of the content, and are robust against the noises in the recording. The synchronization offset is calculated by matching the features at multiple points and not across the frames. Since our approach does not require analyzing the geometry of the cameras or tracking an object, there are no limitations on the number of cameras, continuity of the recordings and co-ordination among the cameras.

4.2 Proposed approach for synchronization: basic principles

The most intuitive and simple approach to synchronize two audio or video recordings would be to compare the audio-visual signals. However, a recording captured at the same time by multiple cameras may look or sound different because of camera position (next to a light source, noisy surrounding), quality of the camera components (lens, resolution, microphone), camera settings (white-point, gamma, audio gain), user handling (shaky hands, jerky movements). Therefore, the raw audio-video signals are not suitable for matching purposes.

Our approach to find a synchronization offset between two recordings involves extracting features and searching for an approximate match between the corresponding features. The features should be accurate in representing time in high resolution, compact in size, easy to extract and match. They should be robust against different camera characteristics and surrounding noises. Furthermore, the features should not be case-specific but applicable to various events such as wed-

dings, concerts and official-meetings. Based on these considerations we selected three features: still-camera flashes in the video domain and onsets and fingerprints in the audio domain. In comparing the features, it is very likely that multiple matches occur, which may result in different synchronization offsets. To determine the most reliable synchronization offset, a voting scheme is applied among the possible offsets. In cases where there are more than two recordings, the synchronization offset is calculated by comparing all possible pairs. The calculated synchronization offsets were checked against the ground truth offsets computed manually by listening and watching the recordings.

The proposed synchronization approach is applicable to recordings with different frame rates. However, depending on the accuracy required by the application, there is a limitation in the required minimum frame rate of the recordings. Since a video frame represents a sampled time instant, the synchronization offset accuracy is determined by the lowest frame rate among the given recordings. According to Equation 3.2 at page 29, the accuracy of a synchronization offset is given by:

$$|t_u(f_i^v) - t_u(f_j^{v'})| < \max(\frac{1}{r^v}, \frac{1}{r^{v'}}) \; . \tag{4.1}$$

For example, to meet the synchronization accuracy for lip-sync, defined by [BT.1359-1, 1998], the frame rate should be at least 22.2 frames per second.

The synchronization offset calculated between a pair of camera-takes from two recordings is valid only for those camera-takes. If the recordings consist of more than one camera-take there are several options to calculate the synchronization offset. One option is to compute individual synchronization offsets for each of the camera-takes. Another option is to compute the time interval between multiple camera-takes in a recording, for example from the camera-time embedded in the video frames. Then calculate the synchronization offset from one of the camera-takes, which can subsequently be applied to the other camera-takes including the interval. Yet another option is to fill the time interval between the camera-takes, for example with dark frames when using flashes, white noise when using audio-fingerprints and silence when using audio-onsets. The fill-ins are chosen such that they do not influence while feature comparison. Then the synchronization offset between the filled recordings can be calculated as in the case of recordings with single camera-takes.

Since all our test recordings consist of 25 video frames per second and a single camera-take, the computation of the synchronization offset, given in Equation 3.2, can be represented in terms of the number of offset frames between the recordings. If f_i^v and $f_j^{v'}$ correspond to synchronized frames of two recordings, the synchronization offset is given by:

$$\Delta_{\text{Sync}} = i - j. \tag{4.2}$$

4.3 Synchronization using flashes

Flash devices in still-photo cameras produce instantaneous flashes of light to help illuminating a scene. Since the typical duration of a flash is about 1 millisecond, a flash forms a very sharp time marker. Most video cameras used in Europe record 25 frames per second. Typically these cameras probe the surroundings at a rate of 50 times per second and interlace the results into 25 frames per second, which makes the shutter time of the cameras 2 msec. Therefore there is a fair chance that a video camera captures a flash if it is within the range of the flashing camera (depending on the power of the flash device, typically around 7 meters).

The effect of flashes on video is a well known problem in shot cut detection. Most algorithms that detect shot cuts are based on measuring discontinuities of luminance values of video frames. Therefore, flashes cause false detections. Robust shot cut detectors have been designed to ignore flashes using other image features such as edges, or luminance independent color spaces described in [Vlachos, 2000] [Guimaraes, Couprie, Araújo & Leite, 2001]. In [Yeo & Liu, 1995] flashes are detected by using the difference in average intensity of consecutive frames to improve shot cut detection in motion-JPEG and MPEG compressed video. In [Takimoto, Satoh & Sakauchi, 2006] similarity in the sequence of flashes among the recordings are used to identify scenes from the same event from a large TV video archive. The flashes are detected using average luminosity, optical flow analysis, and validated assuming the arrival of flashes follows a Poisson distribution.

We propose a method for flash detection, which is independent of video compression standards and assumptions on the flash arrival distributions. The method is based on the luminosity variance across the frames and a locally adaptive threshold is applied to determine the flashes. As a result the video is represented by a flash sequence containing 1 and 0 corresponding to the frames with and without flashes, respectively. The synchronization among multiple videos is computed by matching the flash sequence using cross-correlation and dynamic programming. The synchronization based on flashes is published in [Shrestha, Weda, Barbieri & Sekulovski, 2006]

4.3.1 Flash Detection

The flash appears in video as a sharp increase of brightness in one frame. A luminance histogram of a frame with a flash, shows a concentration of pixels in the higher side of the histogram. For example, Figure 4.3 illustrates four sequential video frames, including a flash frame, and the corresponding luminance histograms. It is evident that the second frame is visibly brighter than the rest, and also the associated histogram shows a peak at the brighter side (right handed part) of the histogram.

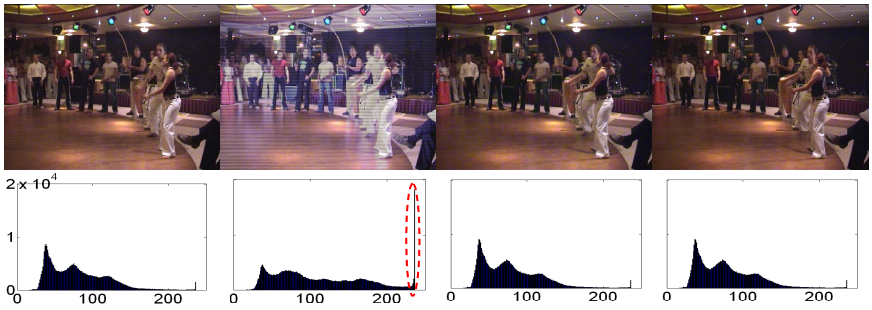


Figure 4.3. Video frames (above) and their corresponding 256 bin luminance histograms (below). The frame second from left contains a flash, and a peak in the corresponding histogram is indicated by a dotted ellipse.

In order to detect a flash we counted the number of pixels contained in the range 171-255 out of 256 luminance bins for each frame and computed the difference across consecutive frames. If the *luminance value* L_i of a frame i represents the total number of pixels contained in this predefined range of bins, the *luminance difference* ΔL_i across the frames is computed by:

$$\Delta L_i = L_{i+1} - L_i.$$

In the luminance difference curve a typical flash is seen as a positive peak followed by a negative peak, where both peaks are similar in height. A locally adaptive threshold is applied in the luminance difference for distinguishing flashes from other peaks, such as when the brightness of a video changes due to an abrupt movement of the camera to another direction with a different lighting condition.

The threshold to detect flash peaks in the luminance difference is calculated in two phases. In the first phase, the behavior of the luminance difference across the majority of the frames is estimated by an initial threshold T_i^{initial} based on a symmetric sliding-window median filter:

$$T_i^{\text{initial}} = K \cdot \text{median}(|\Delta L_{i-m}|, |\Delta L_{i-m+1}|, \ldots, |\Delta L_{i+m}|). \tag{4.3}$$

The median value is calculated on the absolute luminance difference values of a window containing $2m+1$ frames. The values of K and m are chosen heuristically as 10 and 20, respectively. A high value of K and m may miss flashes represented by smaller peaks, however, lower values may result in false detections. The luminance difference is then updated to $\Delta L_i'$ by removing all $|\Delta L_i|$ exceeding the threshold such as:

$$\Delta L_i' = \begin{cases} \Delta L_i & \text{if} \quad |\Delta L_i| < T_i^{\text{initial}} \\ 0 & \text{if} \quad |\Delta L_i| \geq T_i^{\text{initial}}. \end{cases}$$

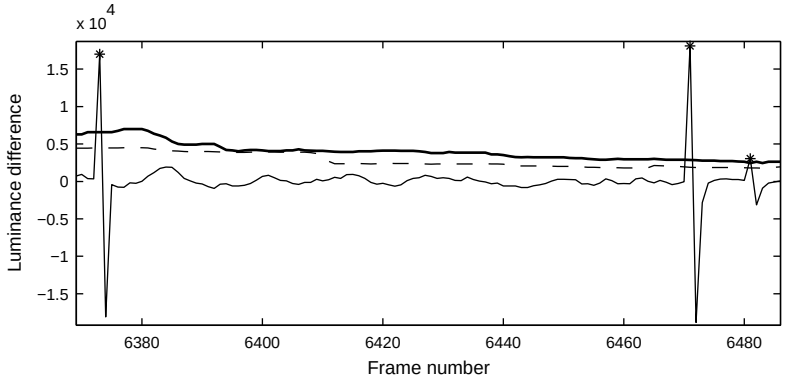


Figure 4.4. The luminance difference curve (thin line), initial threshold (dotted line) and the final threshold (bold line). Detected flashes are represented by '$*$'.

In the second phase, a final threshold is calculated using the updated luminance difference data. A sliding window is applied on the data to calculate the maximum of the absolute value and the standard deviation. The final threshold T^{final} is calculated as:

$$T_i^{\text{final}} = \max(|\Delta L'_{i-m}|, |\Delta L'_{i-m+1}|, \ldots, |\Delta L'_{i+m}|) + \eta \cdot \text{std}(\Delta L'_{i-m}, \Delta L'_{i-m+1}, \ldots, \Delta L'_{i+m}). \quad (4.4)$$

The value of η and m are chosen heuristically as 3 and 20, respectively. Large values of m and η misses flashes with small peaks, while small values cause false detections.

The final threshold is applied to the original luminance difference data to detect the flashes. Once a positive peak above the final threshold is found, the possible results are: (i) a shot cut if no following negative peak is found, (ii) a flash if followed by a negative peak, or (iii) multiple flashes if the negative peak is found after a couple of frames. Multiple flashes appear when a sequence of flashes is used to allow longer illumination time, or when some small flashes are used before the 'real' flash to avoid red-eye effect, or when flashes from different cameras appear shortly after each other. Therefore, once a positive peak above the threshold is found, the search for a negative peak is carried out in a couple of consecutive frames. In this case, we searched for three successive frames, based on our observation of the test recordings where we found a maximum of three flashes present in consecutive frames. Figure 4.4 shows the luminance difference curve, the initial and final thresholds and the detected flashes.

4.3.2 Performance of the flash detection method

The performance of the described method is applied on multiple-camera recordings captured at two different occasions. The first set of recordings is from two cameras inside a church during a wedding ceremony, and the second set is from two cameras at a dance event inside a hall (see page 70 Table 4.2, events 1 and 2). In total, the length of the test video streams was about two and a half hours. The still camera flashes in the recordings were annotated manually by carefully watching the video frames. The results from the flash detection are given in Table 4.1.

Table 4.1. Results from the flash detection method

Description	**Number of flashes**
manually annotated	238
correctly detected	210 (88.2%)
falsely detected	0 (0%)
not detected	28 (11.8%)

The missed detections were caused by very weak flashes due to too large distance from the object or low camera sensitivity. In order to determine an optimal range of luminance for detecting flashes in the histogram, we also tested two other ranges of bins: 128–255 and 192–255. The results show that the former range is more sensitive to smaller changes in luminance thus causing more false detections, while the latter range results in more missed detections. Similarly, we also analyzed the luminance difference across smaller regions of a frame, such as the central region that covers half of the area and four quadrants. The luminance difference across the corresponding regions of consecutive frames becomes very sensitive to smaller variations, such as the reflection of a metal watch during hand movement, resulting in many false detections. Therefore, we used the bin range 171–255 on the full frame luminance histogram for detecting flashes.

4.3.3 Matching flash sequences

After the flashes are detected, the video is represented by a binary flash vector. The elements of the vector are given the value 1 for the frames with a detected flash, and 0 for other frames. In multiple-camera recordings from an event, the corresponding flashes between two videos are related in three possible ways, namely, a *flash match*, a *flash miss*, and a *frame mismatch*.

In case of a *flash match*, the two cameras capture the same flashes. In case of a *flash miss* a camera misses a flash captured by another camera, which maybe due to the closing of a shutter or difference in the distance between the cameras and the flash source. In a *frame mismatch*, a flash captured in one of the recordings is displaced by ± 1, which maybe due to an error while capturing or to drifting of the internal camera-clock. Figure 4.5 illustrates the flash frames from two syn-

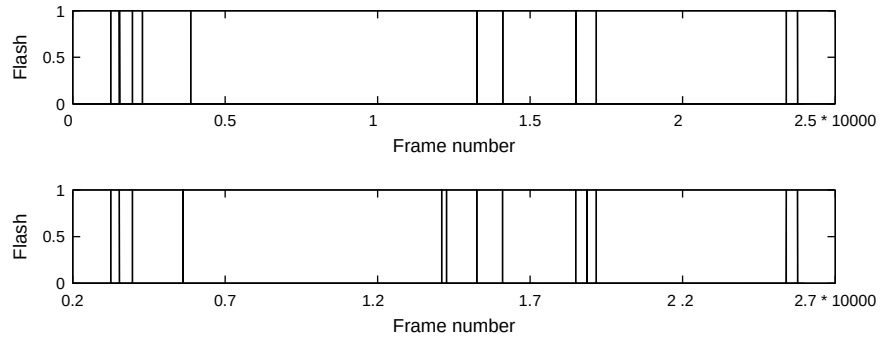


Figure 4.5. Flash sequences from two synchronized test videos from event 1 (see Table 4.2). Video frames are represented by 1 if they contain a flash.

chronized recordings. To synchronize the flashes from different video streams two alternative methods are applied: cross-correlation and dynamic programming.

Cross-correlation

The cross-correlation method operates as a "sliding dot-product" between two flash sequences. The value of the correlation coefficient increases with the number of flash matches and decreases with the flash misses. To model the flash shift, the value of a flash frame is blurred among its neighboring frames. This operation on the flash vector X can be represented as; if $X_i = 1$, then $X_{i-1}, X_{i+1} = 1$ and $X_i = 2$, where the value of i ranges from one to the total number of frames.

The cross-correlation method applied to two vectors results in a vector of correlation coefficients of length $2m+1$, where m is the longest size of the two vectors. The synchronization offset, Δ_{Sync}, is computed as a difference between the indices where the maximum cross-correlation coefficient occurs and $m+1$. Figure 4.6 shows a cross-correlation result between two flash sequences from the recordings of event 2 (see Table 4.2).

In order to test the reliability of the synchronization offset and the robustness of the cross-correlation method on flash matching, we computed the synchronization offset by randomly adding and deleting flashes in two flash sequences corresponding to the videos of event 1. The sequences contain 37 and 50 flashes. Both sequences were subjected to the same amount of change, by flash addition and deletion, and the test was repeated over 2100 iterations for each amount of change. We run a number of iterations sufficient to observe convergence in the run. Figure 4.7a shows the average correlation coefficient or scores obtained by the first and the second highest scoring offsets for different amounts of deleted flashes. The two scores decrease gradually and become closer with the increasing amount of

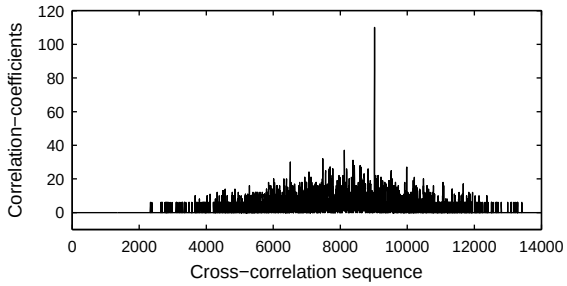
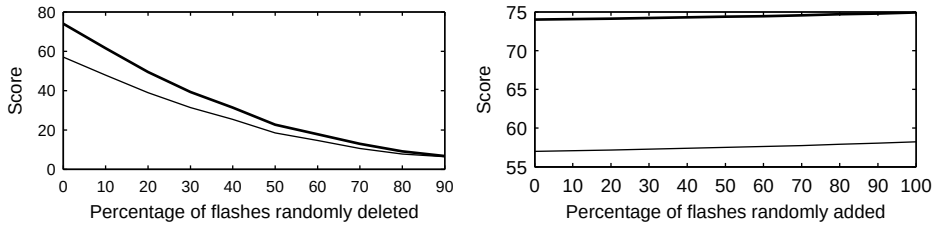


Figure 4.6. The cross-correlation coefficients from two flash sequences from two recordings from event 2 (see Table 4.2). The prominent peak indicates the possible synchronization offset between the recordings.



(a) (b)

Figure 4.7. Scores obtained by the first and the second highest scoring offsets in event 1 (see Table 4.2) represented by a bold line and a thin line, respectively. The scores represent the average scores obtained over 2100 iterations after randomly (a) deleting and (b) adding flashes by the amounts given in the horizontal axis.

deleted flashes. It shows that increasing flash misses decreases the reliability of the synchronization offset. Figure 4.7b shows scores corresponding to the two possible synchronization offsets when different amounts of flashes were added. The score corresponding to the synchronization offset is consistently higher than that of the other offset. The figure shows that the reliability of the synchronization offset is not influenced by the additional flashes. Figure 4.8 shows the number of cases where a correct synchronization offset was computed when different amounts of flashes were randomly added or deleted over 2100 iterations. The cases of successful synchronization for adding flashes remained 100% irrespective of the amount of flashes added. However, for deleting flashes, the number of synchronized cases decreased: when 50% of the flashes was deleted, only 79% of the cases were synchronized. This shows that deleting flashes reduces the possibility of successful synchronization more severely than adding the same amount of flashes.

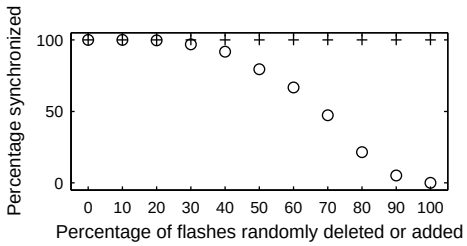


Figure 4.8. Number of times the correct synchronization offset is computed on cross-correlating the videos of event 1 (see Table 4.2) over 2100 iterations when different amounts of flashes were randomly added (+) or deleted (○).

Since the complexity of this cross-correlation method depends on the number of frames, the method is rather computationally demanding. Therefore we also represented the flashes as a vector of inter-flash distances and used dynamic programming [Cormen, Leiserson, Rivest & Stein, 2001] to find the matches. The complexity of this approach depends on the number of flashes rather than the total number of frames. For comparison, the complexity of the cross correlation method using FFT, the most efficient implementation, is $O(n \log n)$, for n being the number of frames while the implementation of the dynamic programming has a complexity of $O(k^2)$, for k being the number of flashes.

Dynamic programming

Dynamic programming is a popular algorithm for string matching applications, which decreases the time complexity of recursive searches by storing intermediary results [Cormen, Leiserson, Rivest & Stein, 2001]. In the case of matching two flash vectors corresponding to videos from different cameras, the problem can be reduced to finding the longest common subsequence with the smallest matching cost. The following paragraphs describe the flash sequence representation, dynamic programming algorithm and the synchronization offset computation.

To represent flash matching as an approximate longest common subsequence matching problem, the flash sequences from two videos are first converted into sequences b_1 and b_2, where each element corresponds to the number of frames between two successive flashes. This shortens the length of the sequences to be matched from being equal to the number of frames to the number of flashes. The cases of flash match, flash miss and frame mismatch with respect to two hypothetical sequences b_1 and b_2 are shown in Figure 4.9. In case of two consecutive flash matches, the number of frames between the flashes should be the same in both sequences. This can be expressed as $b_1(i) = b_2(j)$, where i and j are indices pointing to the elements of b_1 and b_2 respectively.

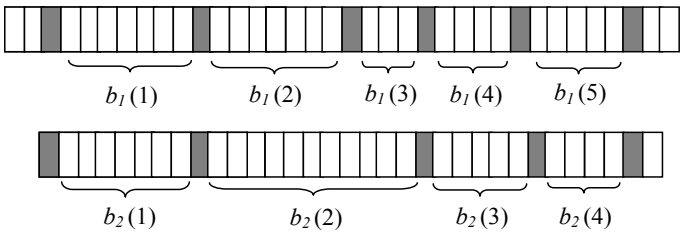


Figure 4.9. Illustration of different possible flash patterns in two recordings, where shaded rectangles represent frames containing flashes. The values b_1 and b_2 denote the number of frames between two flash frames. The possible cases are (i) Match: $b_1(1) = b_2(1)$, (ii) Miss: $b_1(2) + b_1(3) + 1 = b_2(2)$, and (iii) Frame mismatch: $b_1(4) = b_2(3) - 1$ and $b_1(5) = b_2(4) + 1$.

A flash miss in the first video with respect to the second video is given by $b_1(i-1) + b_1(i) + 1 = b_2(j)$. This can be generalized to x misses in the first recording and y misses in the second recording, which can be expressed as:

$$x + \sum_{k=i-x}^{i} b_1(k) = y + \sum_{l=j-y}^{j} b_2(l). \tag{4.5}$$

A frame mismatch, when a flash is detected in the first recording one frame later than in the second recording, can be expressed as $b_1(i) = b_2(j) - 1$ and $b_1(i+1) = b_2(j+1) + 1$.

Using dynamic programming, the optimal match is computed for all possible combinations of subsequences of b_1 and b_2. The flash match and frame mismatch are represented by the *matching* (μ) operation and the flash miss is represented by the *grouping* (γ) operation. The cost function for matching between two values $b_1(i)$ and $b_2(j)$ is given by:

$$\mu(b_1(i), b_2(j)) = |b_1(i) - b_2(j)|^m, \tag{4.6}$$

where m is the cost parameter of a mismatch. A large mismatch means the two videos do not match while a small mismatch (i.e. one frame in our data) could be due to an error in the flash capture, perhaps due to the drift of the internal clocks of the cameras with respect to each other. Therefore the rate of increase of μ is chosen to be a power function of the size of the mismatch. Depending on the probability of having a certain mismatch size in a video, m can be assigned to any positive value. Based on observations on multiple test-runs, we empirically set the value of $m = 2$. Larger values of m might falsely treat flash mismatches as flash misses, while smaller values may falsely treat them as matches.

If x and y represent missed flashes in b_1 and b_2 respectively, the cost function

for grouping $x+1$ elements and $y+1$ elements is given by:

$$\gamma(x+1,y+1) = g \cdot (x+y), \tag{4.7}$$

where g is the cost parameter of a grouping operation. The value of g should be a positive number, which is a penalty for a flash miss. In a multiple-camera recording, the chances are high that the same flash is not captured by all the cameras. For example, a test recording in event 2 (see Table 4.2) shows 13 consecutive flash misses. Based on observations on multiple test-runs, we empirically set the value of $g = 0.2$. Larger values of g may allow less flash misses in the subsequence, while smaller values may result in a false synchronization offset.

First we define the cost of matching the sequences b_1 and b_2, with sizes n_{b_1} and n_{b_2}, for positions $i \in [1, n_{b_1}]$ and $j \in [1, n_{b_2}]$ and for $x+1$ and $y+1$ number of grouped elements in the sequences. The cost of matching at i, j with the given grouping is the sum of the matching costs before the grouping, the costs of the grouping and the match between the produced groups. This is given by:

$$\begin{aligned} D_{x,y}(i,j) = D(i-x-1, j-y-1) + \\ \gamma(x+1,y+1) + \lambda(i,j,x,y), \end{aligned} \tag{4.8}$$

where $\lambda(i,j,x,y)$ is defined as:

$$\lambda(i,j,x,y) = \mu\left(x + \sum_{k=i-x}^{i} b_1(k), \quad y + \sum_{l=j-y}^{j} b_2(l)\right). \tag{4.9}$$

Here $D(i-x-1, j-y-1)$ is the cost of matching the subsequences at i and j before grouping $x+1$ and $y+1$ elements respectively, γ is the cost of the grouping and λ is the cost of matching the groups, as represented in Equation 4.5. The optimal total cost is computed as the minimum of the costs for all possible groupings for up to p elements. This can be represented as:

$$D(i,j) = \min_{x,y} \{D_{x,y}(i,j)\}, \tag{4.10}$$

where

$$\begin{aligned} x &\in \{0,1,..,\min\{i-1,p\}\}, \\ y &\in \{0,1,..,\min\{j-1,p\}\}, \end{aligned}$$

with boundary conditions

$$\begin{aligned} D(0,0) = D(i,0) = D(0,j) = 0 \mid i \in \{1,2,..,n_{b_1}\}, \\ j \in \{1,2,..,n_{b_2}\}. \end{aligned} \tag{4.11}$$

The value of N gives the allowed number of consecutive flash misses. After multiple test-runs, the value is set empirically to $N = 8$.

Figure 4.10 depicts the values of $D(i,j)$ for all the possible groups of flashes in

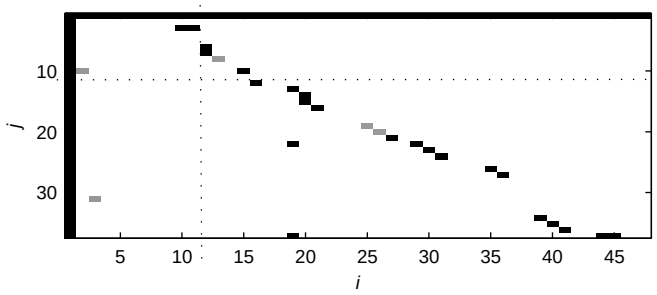


Figure 4.10. The cost of alignment $D(i,j)$ for all the possible groups of flashes in one of the test videos of event 1 (see Table 4.2). Costs larger than a certain threshold are clipped and appear white. The areas above and left of the dotted lines represent the matching costs between very short flash sequences, which are ignored while voting for the synchronization offset.

the videos of event 1 (see Table 4.2). The darker rectangles represent lower matching costs for the subsequences, where the corresponding (i,j) refers to a possible synchronization offset. Costs larger than a certain threshold are clipped and appear white. The dark rectangles at the top and left side of the figure are caused by the boundary conditions employed in the dynamic programming, as given in Equation 4.11. The matches between very short subsequences, for example a detected match at $i = 3$ and $j = 34$ shown in the figure, can be simply due to chance. Therefore, for a reliable match, we discard the matches of the subsequences up to $\frac{1}{4}$ of the shortest sequence. These discarded (i,j) are shown in the figure by a region separated by dotted lines.

The pair (i,j), with lower matching cost refers to a possible synchronization offset represented by Δ_{ij}. The calculation of the possible synchronization offsets is given by:

$$\Delta_{ij} \quad = \quad b_1(i) - b_2(j). \tag{4.12}$$

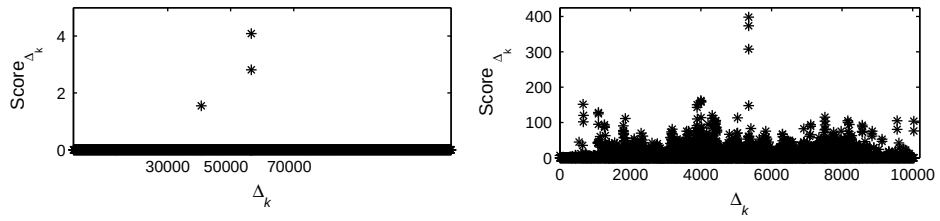
The corresponding match value $V_{\Delta_{ij}}$ is directly proportional to the number of matched subsequences and inversely proportional to the matching cost. The operation for computing the match value is given by:

$$V_{\Delta_{ij}} \quad = \quad \frac{i \cdot j}{D(i,j)}\ , \tag{4.13}$$

where,

$$i \in \left\{\frac{r}{4}, \ldots, n_{b_1}\right\}, j \in \left\{\frac{r}{4}, \ldots, n_{b_2}\right\} \text{ and } r = \min(n_{b_1}, n_{b_2})\ .$$

To select the most reliable synchronization offset Δ_{Sync}, we apply a voting



(a) (b)

Figure 4.11. Examples of possible synchronization offsets (Δ_k) and their corresponding scores (Score_{Δ_k}) based on the results from the dynamic programming. (a) event 1, (b) event 2, given in Table 4.2.

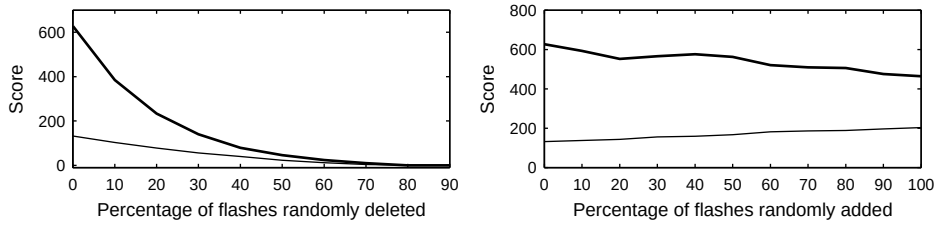
scheme on the unique offsets in the list of possible synchronization offsets Δ_{ij}. The score is calculated by adding all the match values ($V_{\Delta_{ij}}$) that correspond to a unique synchronization offset Δ_k. The scoring operation can be expressed as:

$$\text{Score}_{\Delta_k} = \sum_{v \in V_{\Delta_k}} v. \tag{4.14}$$

The synchronization offset Δ_{Sync} is selected by taking the highest scoring Δ_k. Figure 4.11a and Figure 4.11b show two typical examples of possible synchronization offsets and their corresponding total scores in the test recordings of event 1 and 2 (see Table 4.2). The difference between the scores corresponding to the synchronization offset and the second highest scoring offset indicates the reliability of the synchronization offset.

In order to test the reliability of the synchronization offset and the robustness of the dynamic programming approach, we computed the synchronization offset by randomly adding and deleting flashes in the same flash sequences as described in Section 4.3.3. Figure 4.12a shows the average scores obtained by the first and the second highest scoring offsets for different amounts of deleted flashes. The two scores decrease gradually and become closer with the increasing amount of deleted flashes. This shows that increasing flash misses decreases the reliability of the synchronization offset. Figure 4.12b shows scores corresponding to the two possible synchronization offsets when different amounts of flashes were added. The score corresponding to one of the synchronization offsets is consistently higher than that of the other offset at least by 2 times. This shows that the reliability of the synchronization offset is not influenced by the additional flashes.

Figure 4.13 shows the number of cases where a correct synchronization offset was computed when different amounts of flashes were randomly added or deleted over 2100 iterations. The cases of successful synchronization for adding flashes remained about 70% irrespective of the amount added. However, for deleting flashes,



(a) (b)

Figure 4.12. Scores obtained in dynamic programming by the first and the second highest scoring offsets in event 1 (see Table 4.2) represented by a bold line and a thin line, respectively. The scores represent the average scores obtained over 2100 iterations after randomly (a) deleting and (b) adding flashes by the amounts given in the horizontal axis.

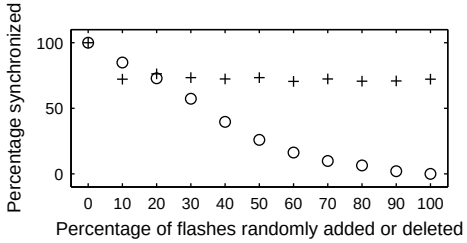


Figure 4.13. Number of times the correct synchronization offset is computed using dynamic programming between the videos of event 1 (see Table 4.2) over 2100 iterations, on randomly deleting (○) and adding (+) flashes by the amounts given in the horizontal axis.

the number of synchronized cases decreased at a fast rate: when 50% of the flashes was deleted, only 25% of the cases were synchronized. This shows that deleting flashes reduces the possibility of successful synchronization more severely than adding the same amount of flashes.

Since flashes are available only in indoor or night recordings, the flash based realization is applicable only in limited cases. Furthermore, audio is generally captured along with the video. Therefore, we present the audio based realizations for synchronizing multiple-camera recordings in the next sections.

4.4 Synchronization using audio-fingerprints

Audio-fingerprints are compact and accurate representations of an audio segment. They can be used in comparing two audio streams, which are typically larger in size. This allows simple comparison and precise temporal match between the streams. The audio-fingerprint based synchronization has been published in [Shrestha, Weda & Barbieri, 2007b].

4.4.1 Fingerprint extraction

We have used the audio-fingerprinting method developed by Haitsma and Kalker, presented in [Haitsma & Kalker, 2002], which has been described in numerous publications and successfully used in applications to identify a music received as a query from a mobile phone. The method is robust against different noises and distortions.

The method generates a 32-bit binary number, called *sub-fingerprint*, for every interval of 11.6 msec based on the spectrum-temporal analysis of the audio in a three second long analysis window. The analysis windows of two successive sub-fingerprints are temporally overlapped by a factor of 31/32. To find a match between two audio streams, a *fingerprint-block* consisting of 256 consecutive sub-fingerprints is used as basic unit. Two fingerprint-blocks are considered to be matching if the Hamming distance or the number of bit errors (BER) between them is less than a threshold T. In [Haitsma & Kalker, 2002], it is proven theoretically and experimentally that the false positive rate of matching fingerprints becomes 3.6×10^{-20} with a threshold $T = 35\%$. In our experiments, therefore, the value of T is set to be 35%. Figure 4.14 shows two fingerprint-blocks from two synchronized recordings with a BER of 24%.

4.4.2 Fingerprint matching

In order to find the synchronization offset between two recordings, fingerprint-blocks from the first recording are compared with those of the second recording. The consecutive fingerprint-blocks from the first recording are non-overlapping sub-fingerprints, whereas the fingerprint-blocks from the second recording are

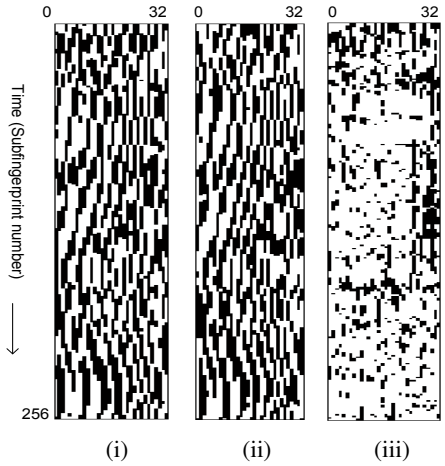


Figure 4.14. (i) and (ii) Fingerprint-blocks from two synchronized recordings from different cameras of event 7 (see Table 4.2). (iii) The difference between (i) and (ii) is given by black color.

overlapping by a factor of 255/256 to achieve maximum synchronization accuracy. If there are X_1 and X_2 sub-fingerprints in two recordings, the total number of fingerprint-block comparisons K is given by:

$$K = \lfloor X_1/256 \rfloor \cdot (X_2 - 255) \; . \tag{4.15}$$

Figure 4.15 shows the BER between fingerprint-blocks between two recordings from different cameras, which contain conversations of durations 83 sec and 45 sec (see Table 4.2, event 7). The darker colors show lower BER. The horizontal line on the colorbar represents the threshold T, such that the blocks darker than the threshold are considered a match. The periodically appearing dark blocks along the diagonal line in Figure 4.15 indicate matches between the two recordings. Figure 4.16 is a zoomed-in view of a seemingly dark block in Figure 4.15, which shows gradual change in BER along the consecutive fingerprint-blocks. The gradual change is caused mainly by the overlapping sub-fingerprints in the consecutive fingerprint-blocks used for matching, where there is a little change in the neighboring fingerprint-blocks. The BER values of multiple neighboring blocks may fall below the threshold T.

As seen in Figures 4.15 and 4.16, multiple blocks satisfy the matching condition of BER less than T. Sometimes there are outliers or false matches, for example due to moments of silence in both recordings. Each of the matching blocks refers to a possible synchronization offset, represented by Δ_j. The calculation of the pos-

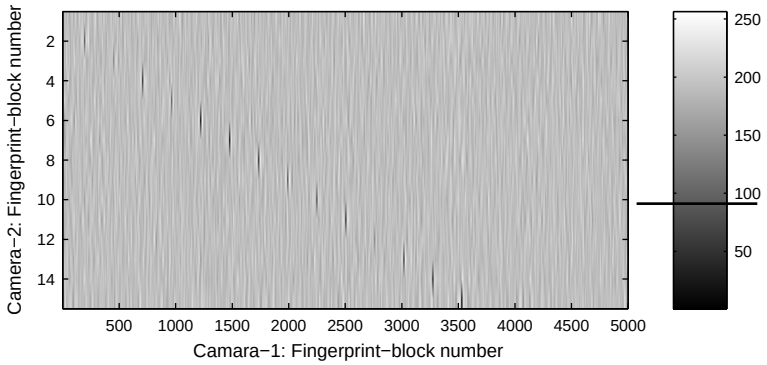


Figure 4.15. The calculated bit errors from two camera recordings of event 2 (see Table 4.2). The bold line on the colorbar indicates the 35% threshold value on the gray colorscale. The fingerprint-blocks having darker colors below the threshold are considered as matches.

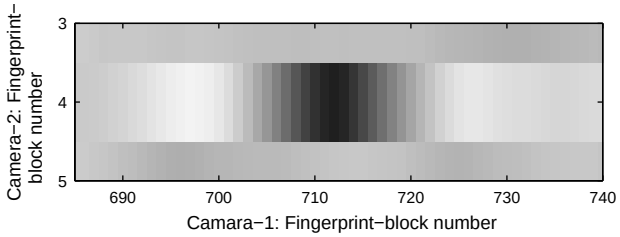


Figure 4.16. Zoomed-in part of Figure 4.15 showing multiple fingerprint blocks that represent matches.

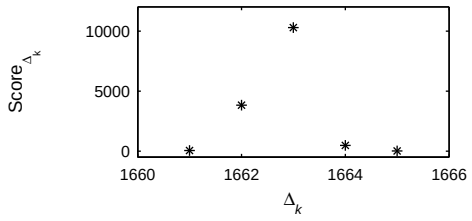


Figure 4.17. Example of a typical distribution of the possible synchronization offsets (Δ_k) and corresponding scores (Score_{Δ_k}), from event 3 (see Table 4.2), based on the results of the fingerprint matching. The synchronization offset with the highest score is selected as the synchronization offset.

sible synchronization offsets and their corresponding BERs (E_{Δ_j}) is given by:

$$\Delta_j = x - y : \text{BER}_{x,y} \leq T \text{ and} \tag{4.16}$$

$$E_{\Delta_j} = \text{BER}_{x,y} . \tag{4.17}$$

Where:

$$x \in \left\{ 1, \ldots, \frac{X_1}{256} \right\} \text{and } y \in \{1, \ldots, X_2 - 255\} .$$

To select the most reliable synchronization offset Δ_{Sync}, we apply a voting scheme on the unique offsets Δ_k out of all the possible synchronization offsets. The score of a unique synchronization offset is proportional to the number of times the offset occurs as a possible synchronization offset and the total sum of the difference between T and the BERs corresponding to the offset. The score calculation of a synchronization offset Δ_k can be represented as:

$$\text{Score}_{\Delta_k} = |\Delta_k| \sum_{e \in E_{\Delta_k}} (T - e). \tag{4.18}$$

$|\Delta_k|$ is the number of times a synchronization offset Δ_k is repeated, which is used as a weight factor for the score calculation. Since the score gets multiplied by the number of occurrences of a synchronization offset, the offsets due to outliers get low scores as the chance of their reoccurrence is very low. The difference between T and BER represents the degree of a match. Finally, the highest scoring Δ_k is selected as the synchronization offset Δ_{Sync}.

The reliability of audio-fingerprint match is thoroughly described in [Haitsma & Kalker, 2002]. The method is proven to be robust under significant audio noise and distortions. Figure 4.17 shows the scores of different possible synchronization offsets. The difference between the highest score and the second highest score shows a reliable selection of the synchronization offset in the test data-set.

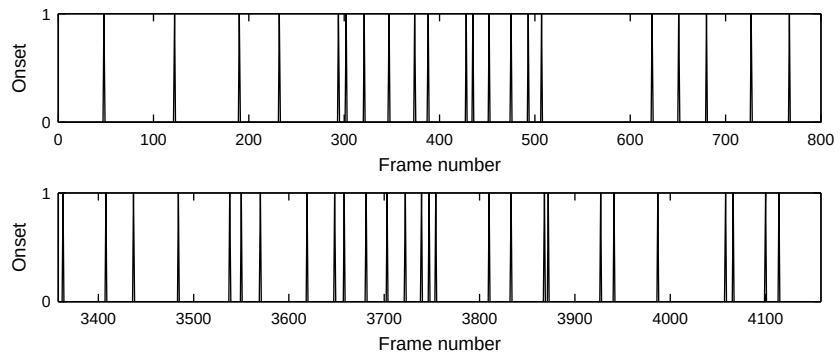


Figure 4.18. Visual representation of the first band audio-onsets of two synchronized recordings from event 3 (see Table 4.2). The horizontal axis shows the frames along time and onset detected in a frame is given by value 1.

4.5 Synchronization using audio-onsets

Onsets are the perceived starting points in an auditory event, with increase in signal power and changes in spectrum [Bello, Daudet, Abdallah & Others, 2005]. They are mainly used for analyzing rhythm, such as beat and tempo, in music. Onsets can be seen as flashes in audio. Since the multiple-camera recordings may contain similar audio, it is expected that two recordings from an event will contain correlated onsets. The onsets represent only positive changes in energy and not the energy throughout the signal as fingerprints.

4.5.1 Onset extraction

We have used the onset extraction method described in [Schrader, 2003]. In this method, the audio signal is divided into 24 bands based on the Equivalent-Rectangular-Bandwidth scale, a psychophysical loudness-frequency scale. Each of these bands is analyzed in a sample window of three seconds, where an energy measurement is generated for every 11.6 msec representing an audio frame. Then the difference in energy is calculated across consecutive frames. If the resulting difference is larger than a threshold, the audio frame is assigned an onset bit 1, otherwise 0. The threshold is optimized to match with the human annotated data sets described in [Schrader, 2003] and [Leveau & Daudet, 2004]. The overall error rates of the method for one-by-one onset detection ranges from 42.8% to 56.6%, which is proven in [Schrader, 2003] as comparable to existing state of the art onset detection methods [Klapuri, 2003]. The onsets from the first band of two synchronized recordings are shown in Figure 4.18.

The present implementation of the onset extraction method described in

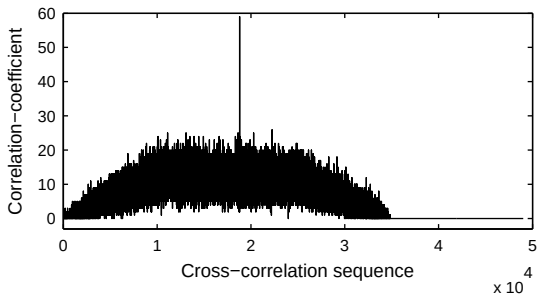


Figure 4.19. The resulting coefficients of cross-correlation between onset sequences from the first out of four frequency bands corresponding to two recordings from event 3 (see Table 4.2). The prominent peak indicates a possible synchronization offset between the recordings.

[Schrader, 2003] allows calculation of onsets in the number of bands that are factors of 24 due to the downsampling applied in the audio signal. While extracting onsets in a lower number of bands, the energy computed in the 24 bands is averaged according to the given number of bands. Due to the averaging of the energy of multiple bands, the energy variation in each band is reduced. Therefore, the number of onsets per band increases with the increase in the number of frequency bands. We extracted onsets in different number of frequency bands: 2, 4, 8, 12, 24 and tested their performance on synchronizing multiple-camera recordings.

4.5.2 Onset matching

The onset sequences from multiple frequency bands of a recording were compared with the sequences from the corresponding bands of another recording using cross-correlation. The method operates as a "sliding dot-product" between two onset sequences resulting in a sequence of correlation coefficients of length $2m+1$, where m is the length of the longest sequence. The possible synchronization offsets are computed as differences between the indices where the maximum cross-correlation coefficient occurs and $m+1$. The value of the correlation coefficient increases with the number of onset matches and decreases with the misses. Figure 4.19 shows the cross-correlation coefficients between onset sequences from the first out of four frequency bands corresponding to two recordings from event 3 (see Table 4.2). A sharp peak in the coefficients suggests the sequences are correlated, while the coefficients from uncorrelated sequences resemble noise. The correlation between the onset sequences of corresponding frequency bands from two recordings result in a set of possible synchronization offsets. To select the most reliable synchronization offset, we applied a scoring scheme on all the unique possible synchronization offsets. The score is directly proportional to the number of times the unique offset

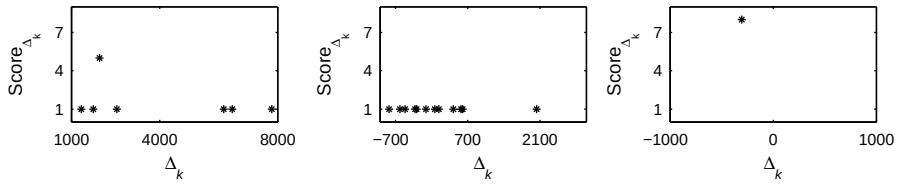


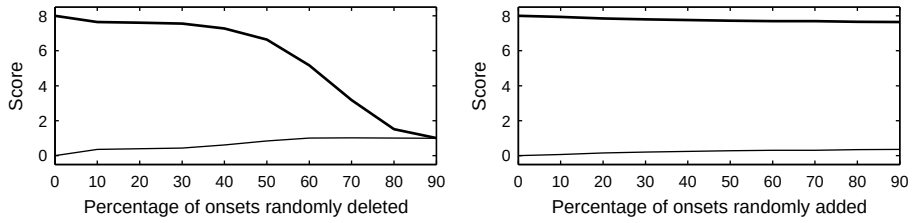
Figure 4.20. Examples of possible synchronization offsets (Δ_k) and their scores (Score_{Δ_k}) from the onset sequence correlation between 8 corresponding frequency bands from the recordings of event 4 (see Table 4.2). Computation of the synchronization offset is accomplished in (a) and (c), but failed in (b).

reoccurs. The score computation for the unique synchronization offset Δ_k can be represented as:

$$\text{Score}_{\Delta_k} = |\Delta_k| \; . \tag{4.19}$$

The highest scoring offset is selected as the synchronization offset, provided the score is equal to at least half of the number of bands used in the onset extraction. This threshold on the score is set after careful observation in the test runs to avoid false positives. For example, if there are 8 bands, an offset should score at least 4 to be selected as the synchronization offset. Figure 4.20 shows three typical examples of possible synchronization offsets and their scores from the onset sequence correlation between 8 corresponding frequency bands from the recordings of event 4 (see Table 4.2). Figure 4.20a shows successful computation of the synchronization offset, where one of the possible offsets scores 5 while the rest of the offsets score 1. Figure 4.20b shows an unsuccessful synchronization, where different frequency bands result in different offsets and all of them score 1. Figure 4.20c shows another successful computation of the synchronization offset, where all frequency bands result in the same offset with the score 8. The difference in score between the synchronization offset and the second highest scoring offset gives a measure of reliability of the computed synchronization offset. For example, the synchronization offset computed in Figure 4.20c is more reliable than the one computed in Figure 4.20a.

In order to test the robustness of the onset cross-correlation method on synchronization and the reliability of the computed synchronization offsets, we randomly added and deleted a predefined number of onsets in two onset sequences corresponding to recordings of event 3. The onsets were extracted in 8 frequency bands, where one sequence contained 70-584 (average 251) onsets per band and another sequence contained 195-612 (average 330) onsets per band. The changes, due to addition and deletion, in the number of onsets were applied equally in both se-



(a) (b)

Figure 4.21. Scores obtained by the first and the second highest scoring offsets in event 3 (see Table 4.2) represented by a bold line and a thin line, respectively. The scores represent the average scores obtained over 2100 iterations after randomly (a) deleting and (b) adding onsets by the amounts, given in the horizontal axis.

quences and the test was repeated over 2100 iterations for each amount of change. The number of iterations was chosen to allow observing convergence in the scores.

Figure 4.21a shows the average scores obtained by the first and the second highest scoring offsets for different amounts of deleted onsets. The score corresponding the highest score decreases and the second highest score increases gradually with the amount of the onsets deleted such that they become very close when 80% of the onsets were deleted. The reliability of the synchronization offset is effected only when high amounts of onsets are deleted. Figure 4.21b shows the scores corresponding to the two highest scoring offsets when different amounts of onsets are added. The difference in scores between the highest and the second highest scores remain above 7 times in all the cases. The reliability of the synchronization offset is not influenced even when the additional onsets are doubled. The reliability of the synchronization offset started to decrease when adding 300% of the onsets.

Figure 4.22 shows the number of successful synchronization over 2100 iterations when different amounts of onsets are randomly added or deleted in the onset sequences same as used in Figure 4.21. The synchronization was successful in all cases until the onsets were added or deleted by 60%. While there was no effect on synchronization until 100% of the onsets were added, however, the number of synchronized cases dropped sharply when more onsets were deleted, such that only 9% of the cases were synchronized when 90% of the onsets were deleted. The robustness of the onset matching method is affected more by deleted onsets than by added onsets, when the rate of acceptable addition or deletion is in the range of 60% to 100%.

When similar tests were performed on different numbers of onset bands, we observed that the robustness of the synchronization offset, both for adding and deleting offsets, increases with the number of bands. Similarly the rate of successful

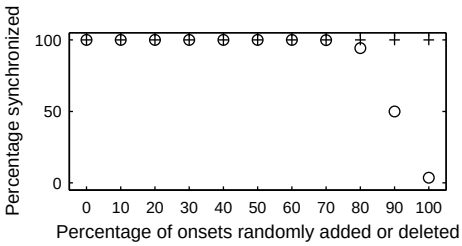


Figure 4.22. Number of times the correct synchronization offset is obtained in event 3 (see Table 4.2) over 2100 iterations, after randomly deleting (○) and adding (+) onsets by the amounts given in the horizontal axis.

synchronization of the recordings, both for adding and deleting offsets, increases with the number of bands. For example, in 2 band onsets, only 20% of the cases were synchronized when 50% of the onsets were deleted while in 24 band onsets 95% of the cases were synchronized when 80% of the onsets were deleted. The two scores decrease gradually and become closer with the increasing amount of deleted flashes.

Both onset and flash sequences are represented by a binary vector and, in principle, dynamic programming is applicable to both. Dynamic programming is beneficial in flash matching because the complexity of the method depends on the number of flashes, which are very sparsely distributed in a video. However, the distribution of onsets is more dense, for example in a 12 min long recording from event 2 (given in Table 4.2) the number of flashes is 80 while the number of onsets ranges from 1030 to 2292 in four bands. Therefore, dynamic programming becomes computationally very expensive and less convenient for onset matching.

4.6 Experimental evaluation

The three realizations of automated synchronization using flashes, audio-fingerprints and audio-onsets were tested on multiple camera recordings made in different occasions in real life conditions. All the recordings consist of a single camera-take captured by independent cameras with the frame rate of 25 frames per second. The flashes and audio in the recordings were not controlled or artificially introduced after capturing.

The description of the event, number of cameras, duration, number of detected flashes and the synchronization results using flashes, audio-fingerprints and audio-onsets are given in Table 4.2. Events 1 (wedding) and 2 (dance) were recorded by amateurs using DV camcorders and the flashes were generated by digital-still cameras of other participants to the event. In event 1 the videos were overlapping

for about 75% of the total time and in event 2 the shorter recording was completely overlapping with the longer recording. The resolution of the videos was 720×576 and the audio sample rate was 48 KHz.

Events 3 and 4 were concert recordings downloaded from YouTube, with video resolution of 320×240 and audio sample rate of 22 KHz. Event 3 (Concert1) was held in an indoor theater with a capacity of 250 people. Event 4 (Concert2) was held in a big stadium with a capacity of 90 000 people. In both events not all the recordings were overlapping. Figure 6.9 in Chapter 6 shows the synchronized recordings of events 3 and 4 as Jason1 and Metallica1, respectively. The rest of the events (outdoor-walk, piano-play and office-talk) were captured using DV camcorders. Event 5 (outdoor-walk) contains a casual conversation between four people walking along a lake amid traffic and wind noises. Event 6 (piano-play) contains a piano recital at a party amid a strong crowd noise. Event 7 (office-talk) contains a conversation among three colleagues in an office. In all three events the shorter recordings are completely overlapping with the longer recordings. Except for two high-definition videos in events 5 and 7, all the other recordings in events 5, 6 and 7 have a video resolution of 720×576 and an audio sample rate of 48 KHz. All the test recordings are made publicly available in YouTube via the link given in [YouTube, 2009b], except for event 1 due to its long duration and privacy issues.

The number of cameras used in the test data set varies from 2 to 10. The duration of the recordings ranges from 20 seconds to 55 minutes. The overlap varies from 20 seconds to 40 minutes. The number and the duration of the recordings are selected to represent different practical cases.

The flash based realization was successful in synchronizing recordings from events 1 and 2. Both events contain a large number of detected flashes ranging from 37 to 80. In the case of event 3, only three out of eight recordings were synchronized. This is due to the big differences in the number of detected flashes (from 0 to 60). Some of the recordings of event 3 were shot far from the concert's stage using mobile phones that captured too few flashes. In the case of event 4, the number of detected flashes in the recordings range from 0 to 35 and none of the recordings could be synchronized. In the recordings from events 5, 6 and 7 no flashes were detected making the flash based realization inapplicable for synchronization. The cross-correlation and dynamic programming approaches used in matching flashes were both able to synchronize the same recordings.

The audio-fingerprint based realization was successful in synchronizing recordings in all the events, except in event 4. Out of ten recordings in event 4 only five were synchronized using audio fingerprints. There were no cases of incorrectly computed synchronization offset. The difficulty in synchronizing event 4 maybe be due to the concert venue: a very large stadium. The cameras were too far apart to capture the same audio source and the recordings were more dominated by

their surrounding noises. Furthermore, the audio was amplified and transmitted by a number of loudspeakers which caused echo effects in the recorded audio.

The audio-onset based realization using different numbers of frequency bands: 2, 4, 8, 12 and 24, are given in Table 4.2. Events 1, 2, 5 and 7 were synchronized by onset sequences of all the given number of frequency bands. However, in events 3, 4 and 6, the number of synchronized recordings increased with a higher number of frequency bands. There were no cases of incorrectly computed synchronization offset. The synchronization performance of using 12 and 24 frequency bands was the same, and in both cases all recordings were synchronized except for one recording in event 4. The recording was captured far away from the stage and the audio was dominated by loud voices of singing crowds rather than music. The audio-fingerprint based realization also failed to synchronize this recording. The overall performance of the onsets in 12 and 24 frequency bands was better than audio-fingerprints.

4.7 Discussion

A successful synchronization of multiple-camera recordings depends on the video and audio quality of the recordings. It is difficult to quantify the quality metrics for synchronization because all of our test recordings are from non-professional cameras with diverse quality and without any standard reference. For example, all the videos downloaded from YouTube contain the same resolution and frame rates, however, the perceptual quality of the recordings are very different. Therefore, we discuss here some general observations made during the experiments.

If a recording is shaky, the frames may contain light sources which resemble flashes causing false detections or capture a flash only in a region of a frame causing missed detections. In the case of an audio recording, if the main audio source, for example music in a concert recording, is corrupted with other audio sources and noises such as voices from audience and echo, the fingerprints and onsets are likely to be different in different recordings. In these cases it becomes difficult to match the recordings.

The synchronization results are also affected by the duration and the number of recordings. A large number of recordings and a long duration create a bigger search space, which increases the likelihood of finding a match. For example, in events 3 and 4, when a pair of recordings failed to match, they were synchronized via a third recording.

The flash based realization is applicable for events where multiple flashes occur and the distance between the recording cameras is not very large, for example events 1, 2 and 3 of Table 4.2. All these recordings were made during events in small areas with no more than 200 people. The chance of capturing the same

flash by different video cameras is high in small areas. In case of events hosted in big areas, for example event 4, there is a big difference in the number of flashes captured by the video cameras and most likely different flashes are captured by different video cameras.

The performance of both cross-correlation and dynamic programming approaches were the same on synchronizing the test recordings. The complexity of the correlation depends on the number of frames, while in dynamic programming the complexity depends on the number of flashes. Since the number of flashes is very low in comparison to the number of frames in a video, dynamic programming is faster in matching flash sequences, especially in case of very long videos like the ones of event 1 in Table 4.2. However, the cross-correlation was more robust and reliable when flashes were randomly deleted or added. Therefore, cross-correlation should be used when complexity is not an issue.

The flash based realization is recommendable in case of events in small area where flashes can also be generated by other means if still-camera flashes are not used, for example in meeting room scenarios [Michel & Stanford, 2006] and [Carletta, 2007], where different cameras record an official meeting. It is also applicable when no audio signal is present like in surveillance applications or when noise is very loud such as in parties.

Flashes are used only in indoor or during night events and are not always captured in recordings due to shutter closure or different fields of view of video cameras than that of the still camera producing the flash. The use of audio based realizations is more practical because audio is very commonly available in recordings. When multiple cameras are present in an event, there is a fair chance that they will record the ‘same’ audio even though they might be pointing at different objects. The realizations based on audio succeeded to find the synchronization offset in cases where it was very difficult for a human ear to find the synchronization point because of the presence of noise, especially from the crowd.

The audio-fingerprint and audio-onset based realizations provide the most reliable synchronization results. Audio-fingerprints represent the signal energy level in 32 bands, while onsets represent only the rise of the signal energy in the given number of bands. Therefore, audio-fingerprints are more vulnerable to echoes and other additional noises like in event 2 where only five out of ten recordings were synchronized. Whereas in the case of onsets, if the level of distortions in consecutive frames is about the same, the difference in energy may not be large enough to influence onsets. Additionally, onsets have more advantages. Since in our realizations onsets use at most 24 bands, compared to 32 bands in fingerprints, the size of the onset sequences are smaller in comparison to the fingerprints. For example, the size of a 12 bands onset sequence is 2.6 times smaller than that of the fingerprints. Furthermore, onset sequences are more compressible because the dis-

tribution of onsets, bit 1, is more sparse than in fingerprints. Another advantage is that the onset based realization can be scaled according to the available computational resources, since onset performance on synchronization improves with the increase in the number of frequency bands. For example, in cases when the number of videos to synchronize is very large, such as videos from the same event in YouTube and video management systems as described in [Kennedy & Naaman, 2009], it is highly recommendable that the synchronization offset is first computed on 4 band onsets and if some videos can not be synchronized then higher bands can be used.

In terms of reliability of a match, audio-fingerprint is proven as highly reliable against different kinds of signal degradation and additional noises, as reported in [Haitsma & Kalker, 2002]. In flash sequence matching, the reliability of a match has a higher decrease rate for missing flashes, than for additional flashes. A similar trend is observed in onset sequence matching, however, onsets are found to be more robust and reliable than flashes. In our experiments, when 50% of flashes were deleted randomly in two test sequences, synchronization was possible only in 25% of the total 2100 cases, whereas when 50% of the onsets were deleted from the 4 bands onset sequences, 91% of the total 2100 cases were synchronized. The better performance of onsets is due to the large number of onsets in multiple bands in comparison to the small number of flashes in a single stream. The availability of a larger number of onset bits in onset sequences allows more opportunities for matching. Since increasing the number of bands used in onset extraction also increases the number of onsets per band, the robustness of the onset sequence matching increases with the number of bands. The audio-fingerprinting method was developed for comparing original music with unknown or low-quality recordings. Similarly, onset extraction was developed for tempo detection in music recorded in studio. Therefore the methods are not meant for analyzing audio from non-professional video recordings with unknown or most likely low quality. Synchronization results could be improved if the methods would be optimized for matching low quality audio.

In the flash, fingerprint and onset based realizations, we extracted features from the audio and the video stream of a recording and compared them with the corresponding features from another recording. In each of the realizations, the synchronization offset is selected from a set of possible synchronization offsets based on how many times an offset has been referred and the value indicating a match, e.g. the matching cost, BER. The multiple realizations, especially an audio and a video feature, can be combined into one system for a more reliable matching. In such a system, the feature extraction and comparison are carried out separately such as in flash, fingerprint, and onset units. Then the possible synchronization offsets resulting from the different units are combined into a single set. The offsets can then

Table 4.2. Test data-set and synchronization results using flashes, audio-fingerprints and onsets. The audio-onset results are given for different frequency bands: 2, 4, 8, 12 and 24.

Event	Num. Cameras	Duration	Num. Flashes	Num. synchronized recordings		
				Flash	Fp	Onset
1. wedding	2	44 – 55 min	37-50	2	2	2, 2, 2, 2, 2
2. dance	2	11 – 12 min	43-80	2	2	2, 2, 2, 2, 2
3. concert1	8	20 – 368 sec	0-60	3	8	6, 8, 9, 9, 9
4. concert2	10	59 – 339 sec	0-35	0	5	4, 9, 9, 9, 9
5. outdoor-walk	2	30 – 38 sec	0	0	2	2, 2, 2, 2, 2
6. piano-play	2	71 – 79 sec	0	0	2	0, 0, 0, 2, 2
7. office-talk	3	45 – 83 sec	0	0	3	3, 3, 3, 3, 3

be evaluated depending on the scores from the individual units and the number of units which refer to the offset. The combined scoring method could increase the reliability of the synchronization offset even more.

4.8 Conclusions

We proposed an automated synchronization approach for multiple camera recordings based on visual and audio features. We developed three different realizations of the approach. The first realization is based on still-camera flashes. In this realization videos are represented by binary vectors representing frames with or without flashes, according to the results of a flash detection algorithm. The matching between these two vectors is treated as an inexact string matching problem and solved using cross-correlation and dynamic programming. The other two realizations of the synchronization approach are based on audio-fingerprints and audio-onsets. We used the fingerprint extraction method described in [Haitsma & Kalker, 2002] and computed the synchronization offset based on bit error rate calculation.

In our experimental evaluation, the audio-fingerprint and onset based realizations were able to synchronize most videos in all test cases, while the flash based realizations succeeded in synchronizing some cases. The choice between flash, audio-fingerprint and onset based realization depends on the type of application. The flash based realization is useful for recordings containing flashes and indoor events localized within a flash range like weddings, meetings etc. It is independent of the presence of audio. To compute the synchronization offset, two videos should capture at least two common flashes. The audio based realizations are applicable in recordings that contain audio. The audio based realizations can be applied in any multiple-camera recording that contain at least three seconds of common audio. We also tested the robustness of the flash and onset based realizations by adding and removing flashes and onsets, respectively, at random frames and computing

the synchronization offset. The onsets are found to be more robust and result in more reliable synchronization offsets than flashes.

The audio-onsets based realization is found to be the most successful for synchronizing multiple camera videos. The performance of audio-onsets increases with the increase in the number of bands. In our experiments, 12 band onsets were able to synchronize 29 out of 30 test recordings. Increasing the number of bands to 24 did not further improve the result. The audio-fingerprint based realization was successful in synchronizing 25 out of 30 test recordings and failed in cases where the audio was of low quality due to noises and echo. In addition to better performance, the onset sequences are smaller in size and more compressible than fingerprints. The onset based realization can also be scaled according to the available computational resources, since performance on synchronization improves with the increase in the number of frequency bands. Such scalable realization is useful in synchronizing very large number of videos like in YouTube and Yahoo! video repositories.

The synchronization precision in the flash based realization is determined by the lowest video frame rate among the given recordings. Therefore, for combining multiple camera videos, as required by the standard set in [BT.1359-1, 1998], the minimum frame rate of a recording should be 22.2 frames per second. In our test data, which consisted of multiple videos recorded in different events using a constant frame rate of 25 frames per second, the precision of a synchronization offset in the flash based realization is ± 40 msec. The synchronization based on the audio results in higher synchronization precision, because audio is sampled with a higher frequency than video. In our audio based methods, an audio frame is generated every 11.6 msec. Therefore, the method offers a synchronization precision of ± 11.6 msec. It can be concluded that synchronization among multiple camera recordings can be achieved using audio-visual contents of the recordings.

In the mashup generation problem from multiple-camera concert recordings, the test recordings are comparable to events 3 and 4. In a concert the main audio source is the performer, however, it is most likely affected by noises from the audience and echo. Since the audio-onset and audio-fingerprint based realizations were found to be successful in synchronizing recordings with similar noise cases, both methods are applicable in our mashup generation.

5

Feature analysis

In Chapter 2, we elicited requirements for generating a mashup from multiple-camera recordings captured by non-professional users in a music concert. Then we proposed a solution approach for the mashup generation problem in Chapter 3. According to the solution approach, in this chapter we propose methods to estimate

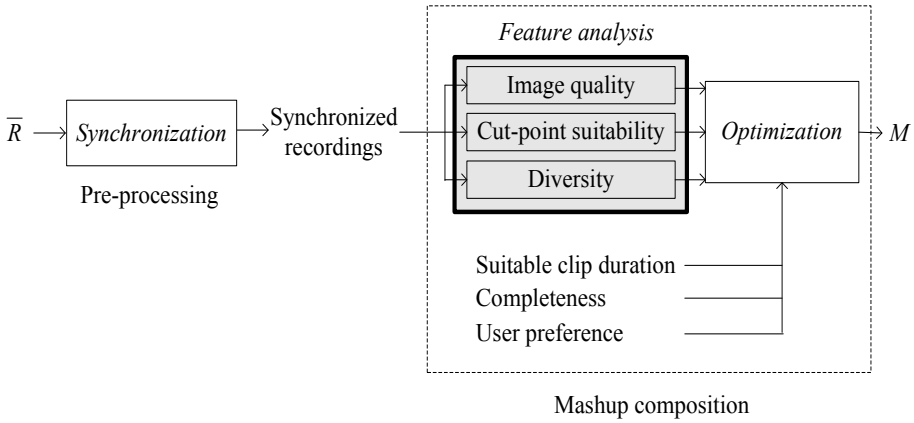


Figure 5.1. Schematic representation of the proposed mashup generation problem, in which the feature analysis step is highlighted.

the degree of fulfillment of the Requirements 2.2 (image quality), 2.3 (diversity) and 2.5 (suitable cut-point) in the recordings based on the audio-visual features. Figure 5.1 shows the proposed mashup generation approach with the feature analysis step highlighted. The degrees of fulfillment of the requirements are measured in terms of numeric values, called *scores*. The higher the degree of fulfillment, the higher the corresponding score. We apply different content analysis techniques to extract features and compute the scores based on image quality, diversity and cut point suitability.

The rest of the chapter is organized as follows. Section 5.1 describes the estimation of image quality based on blockiness, blurriness, brightness and shakiness of the video frames. Section 5.2 describes the estimation of cut-point suitability for the video frames based on the changes in audio, camera motion and brightness. Section 5.3 describes the estimation of diversity based on the image distance. Finally, Section 5.4 presents conclusions of the chapter.

5.1 Image quality estimation

According to Requirement 2.2 (image quality), formalized in Section 3.4.2, the image quality of a mashup and its clips should be as high as possible. As a measure of the degree of fulfillment of this requirement, we computed the image quality score of the frames in a video by extracting and analyzing different visual features. Next, the image quality score of a clip is computed by averaging the scores of the frames contained in the clip.

The quality estimation of an image is widely used in applications like codec design and display calibration. In these applications, the image quality is measured by means of comparing the output or processed image against the input or reference image in terms of metrics like mean square error or peak signal to noise ratio. In our recordings, however, there is no information available about the actual scene or the camera settings that can be used as a reference for estimating the image quality. Therefore, we used a *no-reference*, also called *blind* quality assessment method, which estimates the image quality based on objective measures of different features that influence the perception of quality.

Prior works on no-reference image quality estimation were done in different contexts such as removing artifacts in home videos [Yan & Kankanhalli, 2002], developing perceptual quality models [Li, 2002], [Wang, Sheikh & Bovik, 2002] and estimating network performance in a real-time video transmission [Yang, Wan, Chang & Wu, 2005]. In [Yan & Kankanhalli, 2002] the lighting and shaking artifacts in home videos are first detected, measured and then removed. The quality of a JPEG compressed image is estimated in [Wang, Sheikh & Bovik, 2002] according to the blockiness and blurriness measured in the image, while in [Li, 2002]

(a) $B = 0.01$ (b) $B = 0.09$ (c) $B = 0.17$

Figure 5.2. Examples of test frames with (a) low, (b) medium and (c) high blockiness. The measured blockiness score is given by B.

according to the edge sharpness, random noise level, ringing artifacts and blockiness. In [Yang, Wan, Chang & Wu, 2005], video quality is measured based on the spatial distortions and temporal activities along the frames.

Since there are no standard features and analysis techniques on image quality estimation, we looked for methods to measure some of the popular quality factors in videos: *blockiness, blurriness, brightness, noise* and *shakiness*. We applied the methods on different videos from multiple-camera recordings captured during concerts and evaluated the relevance of the quality factors. The evaluation was done by two experts in video processing at the Philips Research. We showed them 12 test frames from four concert clips and asked for the factors and their level of influence on the perception of quality. The most prominent quality factors identified by the experts were *blockiness, blurriness, brightness* and *shakiness*. The influence of *noise* on the test frames was perceived as minimal because all the test recordings were captured digitally. In the following sections we describe the methods used for measuring the blockiness, blurriness, brightness and shakiness factors.

5.1.1 Blockiness

The widely used lossy video codecs, such as MPEG, JPEG2000, H.26x use discrete cosine transform (DCT) based compression. They require segmenting a video frame into non-overlapping blocks, typically containing 8×8 pixels, and quantizing the blocks separately. Blocking artifacts, which are a major source of distortion in such compression techniques, are caused by discontinuities at the block boundaries. Figure 5.2 shows some test frames with different amount of blockiness.

Existing methods for blockiness measurement are based on the degree of discontinuity or strength of the edges at the block boundaries (typically, every 8^{th} horizontal and vertical pixel of an image). In [Wang, Sheikh & Bovik, 2002] blockiness is measured based on the difference in luminance and signal activity across the block boundaries. If the difference is high in luminance and low in signal ac-

tivity, then the boundary pixel is considered as blocky. In [Gao, Mermer & Kim, 2002] the discontinuity is measured by the luminance variation in block boundaries of the DC component of an image. Then two thresholds, T_{high} and T_{low} are used to measure the strength of the discontinuity, such that if the discontinuity above T_{high}, the boundary pixel is considered a real edge, called hard edge, of the image and if the discontinuity is below T_{high} but above T_{low} the boundary pixel is considered a soft edge, which is the effect of blockiness

The method proposed in [Gao, Mermer & Kim, 2002] requires computing the DCT for every video frame. Considering the high computational cost of DCT, we did not apply the method for our multiple-camera recordings. We tested the method proposed in [Wang, Sheikh & Bovik, 2002] on the concert video frames obtained from YouTube. The results did not correspond to the perceived level of blockiness by our experts. The failure to measure the blockiness in the test frames maybe due to the low visual quality of the test images such that the signal activity measure did not provide any reliable information or perhaps many hard edges were miscalculated as being the effect of blockiness.

We propose an algorithm for blockiness measurement based on detecting edges in an image and applying thresholds to separate the hard edges from the soft edges, similar to the approach in [Gao, Mermer & Kim, 2002] but on a luminance frame. The number of soft edge pixels at the block boundaries represents the blockiness in the image. The following paragraphs describe the blockiness measurement in horizonal direction. A similar approach is used in the vertical direction.

1. In order to detect edges, a horizonal Sobel mask ($\mathcal{S}_h$) is applied by convolving with the grayscale frame Y, derived from the YCbCr color-space of the image. The resulting gradient image E'_h indicates discontinuities in the image. The operation can be represented as:

$$E'_h = Y * \mathcal{S}_h, \tag{5.1}$$

$$\text{where,} \qquad \mathcal{S}_h = \frac{1}{3}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.2}$$

   and '$*$' represents the convolution operation. The gradient of an image is given by the summation of the horizonal and vertical gradients:

$$E' = E'_h + E'_v. \tag{5.3}$$

2. The thresholds T_{low} and T_{high} are applied on the gradient image. The gradient values higher than T_{high} correspond to hard edges while the gradient values higher than T_{low} and lower than T_{high} correspond soft edges. The operation can be represented as:

$$E_h(i,j) = \begin{cases} 1 & : \text{ if } \quad T_{\text{low}} < E'_h(i,j) < T_{\text{high}} \\ 0 & : \text{ otherwise.} \end{cases} \tag{5.4}$$

The values of T_{low} and T_{high} are chosen as 50 and 150, respectively based on their performance on the test data-set.

3. The soft edges are searched in the horizontal direction in every 8^{th} pixel. In a blocky image, ideally, all the 8 consecutive pixels in the boundary should correspond to the soft edge. However, a few pixels might indicate otherwise due to noise or other artifacts. Therefore, based on the results on our test data-set we set the threshold to 6 pixels, such that if the number of boundary pixels corresponding to the soft edge of the block are more than 6, we consider the boundary a soft edge. The average number of boundaries representing the soft edges indicate the amount of blockiness, which can be represented as:

$$B_h = \frac{1}{(\lfloor W/8 \rfloor - 1)(\lfloor H/8 \rfloor - 1)} \sum_{i=1}^{\lfloor H/8 \rfloor - 1} \sum_{j=1}^{\lfloor W/8 \rfloor - 1} \beta_{ij}, \tag{5.5}$$

$$\text{where, } \beta_{ij} = \begin{cases} 1 & : \text{ if } \sum_{k=1}^{8} E_h(8i, 8(j-1)+k) > 6 \\ 0 & : \text{ otherwise,} \end{cases} \tag{5.6}$$

W and H represent the rows and columns of the image, respectively. The blockiness score B is computed as an average of both horizontal and vertical blockiness:

$$B = \frac{B_h + B_v}{2}. \tag{5.7}$$

The range of B lies between 0 and 1, where a higher value indicates more blockiness. Figure 5.2 shows some example test frames to different amount of blockiness. The blockiness score B on the test frames corresponds with the subjective evaluation of blockiness by the experts described in Section 5.1.

5.1.2 Blurriness

Blurriness is characterized by reduction of edge sharpness. It can be caused by coarse quantization during compression, filtering for blockiness or noise during decoding and the lens out of focus or shakiness of the camera during capturing. Figure 5.3 shows some test frames with different amount of blurriness. Related works in blur detection are based on measuring signal activity [Wang, Bovik & Evans, 2000], contrast and orientation of an edge [Li, 2002], and average spread of the edge [Ong & Et. al., 2003].

When we implemented methods described in [Ong & Et. al., 2003] and [Wang,

(a) $Z = 3$ (b) $Z = 3.9$ (c) $Z = 12$

Figure 5.3. Example of test frames with (a) low, (b) medium and (c) high blur. The measured blurriness score is given by Z.

Bovik & Evans, 2000] and tested them on our concert video frames, the blurriness measurement by the former method was a better match to our experts' evaluation, described in Section 5.1. Therefore we selected the method as described in [Ong & Et. al., 2003] for our blur measurement, which is based on the measurement of the spread of the edges. Firstly the edges are detected, then the spread of the slope of the pixels corresponding to the edge is measured in specific directions. The following paragraphs describe the steps followed in the method.

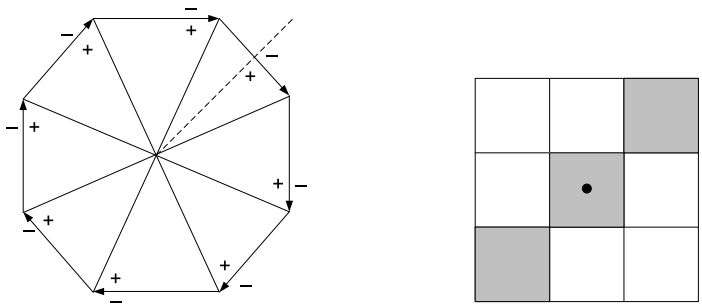
1. In order to detect the edges in an image, the horizonal and vertical Sobel masks are applied in the image Y to obtain the gradients as described in Equations 5.1 – 5.3. The resulting horizontal and vertical gradient images are combined to obtain the amplitude (G) and direction (ϕ) for every pixel:

$$G = \sqrt{{E'_v}^2 + {E'_h}^2}, \tag{5.8}$$

$$\phi = \arctan\left(\frac{E'_v}{E'_h}\right). \tag{5.9}$$

   A pixel is declared as an edge pixel if its gradient value is higher than a threshold T. Based on the experimental results on our concert video frames we set the value $T = 100$.

2. The gradient's direction (ϕ), which ranges from 0 to 360°, points to the direction of increasing luminosity. The direction of an edge is perpendicular to the direction of a gradient. Since each pixel has 8 neighboring pixels, to determine the gradient's direction towards one of the neighboring pixels, the range 0 to 360° is divided into 8 quadrants. Each quadrant represents the direction of a neighboring pixel. For example, $\phi = 170°$ points to a pixel in the horizontal direction, while $\phi = 45°$ points to a pixel in the diagonal direction. Figure 5.4a shows the 8 quadrants representing the direction of ϕ and an example gradient from direction '-' to '+' given by a dotted line, whose corresponding neighboring pixel is shown in Figure 5.4b.



(a) (b)

Figure 5.4. (a) The eight quadrants of the gradient direction. The direction of the edge is given by an arrow and the direction of the gradient is given from '-" to "+". The dotted line shows an example of a gradient direction of 45°. (b) The direction of search for the edge spread corresponding to the gradient direction of 45°. The pixel corresponding to an edge is represented by a gray square with a dark dot in the middle and the pixels on the diagonal directions, given by the two gray squares, are searched for measuring blurriness.

3. The spread of an edge is measured in two sides according to the direction given by ϕ. For every pixel corresponding to an edge in Y frame, the number of pixels are counted towards and opposite direction of ϕ as long as the luminance value keeps on increasing and decreasing, respectively. The sum of the counts in both directions gives the measure of the spread of an edge pixel.
4. The blurriness is estimated by dividing the total number of pixels corresponding to the edge spread by the number of pixels corresponding to the edge. If there are m pixels representing an edge and s^e pixels representing the total edge spread, then the blurriness score (Z) is calculated as:

$$Z = \begin{cases} \frac{s^e}{m} & : \text{ if } m \neq 0 \\ 0 & : \text{ otherwise.} \end{cases} \tag{5.10}$$

The lower bound of Z is 0, when no edges are detected in an image or when no pixel corresponds to the edge spread. The upper bound of Z is limited by the possible edge spread, which is the number of pixels along the diagonal direction of the image. Figure 5.3 shows some test frames and their corresponding blurriness score. The blurriness score Z on the test frames corresponds with the subjective evaluation of blockiness by the experts described in Section 5.1.

5.1.3 Brightness

The brightness of an image is attributed to the visual perception of light in an image. In previous works [Yan & Kankanhalli, 2002] and [Campanella, 2009], the average luminance and contrast are used for brightness estimation. In our observation of the concert recordings, we found frequent occurrence of frames containing pixels, which are clipped by the maximum and minimum luminance range 255 and 0, respectively. Such pixels are called *burned pixels* and frames containing these burned pixels are called *burned images*. Burned images are caused by a very bright light source against a camera or very dark scenes. An image with high brightness and contrast values is desirable for its clarity and sharpness, while burned pixels are undesirable. To measure the brightness of an image we used the average *luminance, contrast* and the *amount of burned pixels*. The method followed in computing brightness is described below:

1. The average luminance I_l of a frame is given by the mean of the pixel values corresponding to a gray scale image Y. The calculation is represented as:

$$I_l = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} Y(i,j). \tag{5.11}$$

   The average luminance value ranges from 0 to 255. In a typical concert setting, except for the outdoor concerts, lights are mainly focused towards the stage and the rest of the venue is poorly lit. The recordings from the audience suffer from poor luminance. Therefore, for concert videos higher luminance is associated with a clearly visible image.

2. The contrast I_ε is a measure of difference in brightness between light and dark areas in an image. It is calculated by the spread of the luminance values of the pixels, such as:

$$I_\varepsilon = \sqrt{\frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} (Y(i,j) - I_l)^2}. \tag{5.12}$$

   The contrast value ranges from 0 to 255. A low contrast image appears, generally, flat or dull.

3. The burned pixels lie near the extreme ends (complete white or black) of the luminance range of 0 to 255. The fraction of burned pixels I_p is calculated as:

$$\begin{aligned} I_p &= \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} p(i,j), \\ \text{where, } p(i,j) &= \begin{cases} 0 & : \text{ if } \quad 2 > Y(i,j) > 254 \\ 1 & : \text{ otherwise.} \end{cases} \end{aligned} \tag{5.13}$$

The amount of burned pixels ranges from 0 to 1. The burned pictures are generally undesirable as they provide little color or texture information and produce a very disturbing effect.

The three factors I_l, I_ε and I_p can be combined in multiple ways to derive a brightness measure. Based on our experimental observation on the concert video frames, we computed the brightness score by a linear combination of average luminosity and contrast, weighted by the amount of burned pixels, which can be represented as:

$$I = \frac{I_l + I_\varepsilon}{I_p}, \tag{5.14}$$

where $0.1 \leq I_p \leq 1$.

We observed in our experiments that although the frames contain some burned pixels, they are not perceived as disturbing. Therefore, we set the minimum value of burned pixels to 0.1 such that all the frames are considered to contain 10% burned pixels. The minimum value of I_p also avoids division by very low values. The lower bound for I is 0, while the upper bound is 5100, when $I_l = I_\varepsilon = 255$ and $I_p = 0.1$. Figure 5.5 shows the luminance, contrast, and amount of burned pixels in example frames from concert videos. Figure 5.5a shows a very dark frame captured when moving the camera from the audience towards the stage. The brightness score of the image is only 2.9, with very low luminance and a high number of burned pixels. Figures 5.5b and 5.5c show typical frames from indoor concerts with brightness score of around 1400-1500. Figure 5.5d shows a frame from an outdoor concert with brightness score of 2627, which has a relatively lower contrast value but a very high luminance.

The still-camera flashes captured in the recordings cause a sharp rise in the values of luminance and sometimes in the number of burned pixels, which might result in very high brightness. Therefore, to avoid such situations, the flash detection technique described in Chapter 4 is applied and the brightness values of the frames with flashes are substituted by the values of the neighboring frames.

5.1.4 Shakiness

Shakiness in a video is caused by the instability of a camera, such as when a cameraman walks with a handheld camera or when he applies fast zooming or panning operations. It induces motion in unwanted and repeating directions along the frames. In order to measure shakiness in a video stream, we used the method described in [Campanella, Weda & Barbieri, 2007] since this method has been successfully applied in measuring shakiness in home videos. The steps used in the method are described below:

1. The sweeping camera motions in horizontal direction, *pan*, and in vertical

(a)
$I_l = 0.16$, $I_\varepsilon = 0.55$,
$I_p = 0.24$, $I = 2.89$.

(b)
$I_l = 83.72$, $I_\varepsilon = 88.51$,
$I_p = 0.11$, $I = 1565$.

(c)
$I_l = 84.17$, $I_\varepsilon = 56.48$,
$I_p = 0.10$, $I = 1406$.

(d)
$I_l = 235.86$, $I_\varepsilon = 26.89$,
$I_p = 0.10$, $I = 2627$.

Figure 5.5. Measured brightness value (I) in example concert video frames along with luminance (I_l), contrast (I_ε) and amount of burned pixels (I_p).

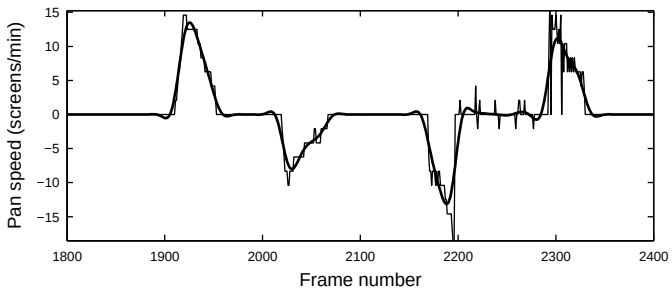


Figure 5.6. Camera panning speed in a video before filtering (p), represented by a thin line and after filtering (p_f), represented by a bold line).

direction, *tilt*, are calculated using a luminance projection method [Uehara, Amano, Ariti & Kumano, 2004]. In this method, the luminance values of every row are summed up in a vertical projection and of every column in a horizontal projection. If the camera is moved vertically or horizontally, the corresponding projections will also shift in the same direction. For example, if there is a panning in the right direction, the values of the horizontal projection will shift towards right. The camera motion, pan and tilt, is calculated by correlating the projections along the frames. The speed of the camera is measured in screens per minute, where one screen is equivalent to the horizontal dimension of the frame in case of pan and to the vertical dimension of the frame in case of tilt. Figure 5.6 shows the panning speed of a sequence of frames represented by a thin line.

2. A median filter is used on the calculated camera speed to remove outliers caused by imperfections in the luminance projection correlation. The high frequency components in the camera speed are considered as the effect of shakiness, while the low frequency components are considered as the intended camera motion. Therefore a low-pass filter (25 tabs FIR) is applied to extract the smoothed camera motion. Figure 5.6 shows the panning speed after low-pass filtering given by a bold line. A positive value corresponds to panning from left to right while a negative value corresponds to panning in the opposite direction.
3. The amount of shakiness is given by the difference in the pan and tilt speeds before and after filtering. If p and t represent pan and tilt speeds calculated from step 1 and p_f and t_f are the filtered value from step 2, then for each frame the shakiness measure J is given by:

$$J = \sqrt{(p - p_f)^2 + (t - t_f)^2} \; . \tag{5.15}$$

The values of pan and tilt speeds range from -187 to 187 screens per minute [Campanella, 2009]. The limits correspond to the shift of a pixel about three screens per second, which is perceived as noise. Therefore, the pan and tilt values outside the range are considered as noise and not a camera motion. The shakiness score can vary in between 0 and 264.5.

5.1.5 Discussion

The measured scores of the quality factors: blockiness (Equation 5.7), blurriness (Equation 5.10), brightness (Equation 5.14), and shakiness (Equation 5.15) are used in computing the image quality score of a video frame. Since the values corresponding to the different factors have different ranges, we need to normalize them before combining them to compute the image quality. We normalized them with respect to their corresponding maximum values, so that the values of the factors are in between 0 and 1. The maximum values were obtained from the test recordings captured in the concerts. Therefore, the algorithm is dependent on the maximum values of the test set.

There are different ways to combine the factors to estimate the image quality, such as linear addition [Li, 2002], linear multiplication [Wang, Sheikh & Bovik, 2002]. We tested the quality score computed by the two methods on our test frames. Since the quality score using multiplication has a wider range, it is more suitable for comparing the scores among the recordings of a multiple-camera recording. Therefore, we used the product of the quality scores of the different factors to compute the quality score. Since shakiness, blurriness and blockiness attribute a negative quality, the factor values are subtracted from one. The image quality score q of a video frame is given by:

$$q(f^v) = I' \, (1 - B') \, (1 - Z') \, (1 - J'), \tag{5.16}$$

where I', B', Z' and J' represent the normalized values of the factors brightness, blockiness, blur and shakiness, respectively.

Figure 5.7 shows the quality scores of three recordings of a multiple-camera recording (see Figure 6.9 at page 106: camera 4, 5 and 6 of concert RATM) in a common time-line represented by frame numbers. Typically, the quality scores of the recordings vary continuously due to many factors, for example, when a user moves his camera from the stage view towards the audience view the frames during transition become blurred, shaky or dark. The visualization of the frames at different points in the curve is presented in Figure 5.8. Figures 5.8a, b and c are synchronized frames, which correspond to the same time during the PinkPop-08 festival in Landgraaf, The Netherlands. Due to the large size of the concert arena, the different recordings capture very different videos. Figure 5.8a shows the back of the head of a person in the audience (dark and blurred image) with $q = 0.05$,

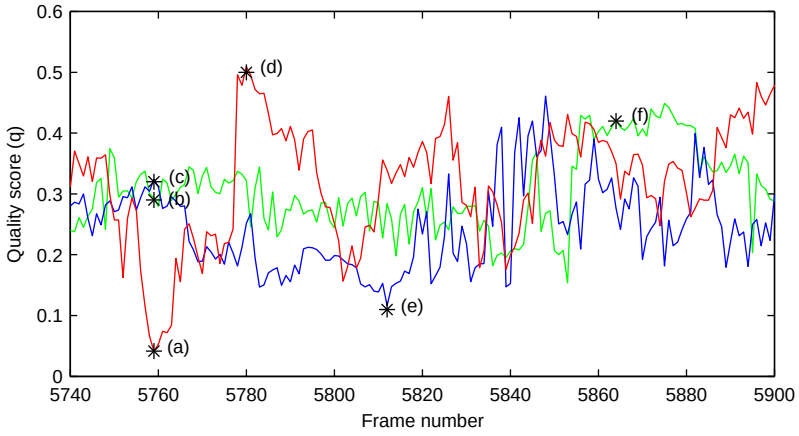
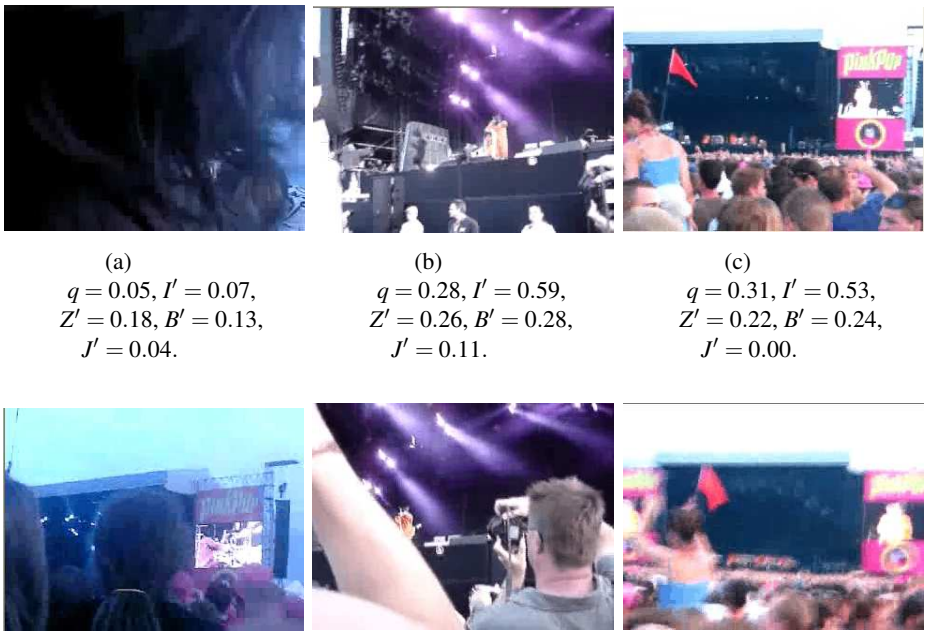


Figure 5.7. Quality score of recordings from a multiple-camera recording (see at page 106, camera number 4, 5 and 6 of concert RATM given in subfigure 2 of Figure 6.9) in a common time-line represented by the frame numbers. The frames indicated by '$*$' and their corresponding quality scores are given in the figure below.



(a) $q = 0.05$, $I' = 0.07$, $Z' = 0.18$, $B' = 0.13$, $J' = 0.04$.

(b) $q = 0.28$, $I' = 0.59$, $Z' = 0.26$, $B' = 0.28$, $J' = 0.11$.

(c) $q = 0.31$, $I' = 0.53$, $Z' = 0.22$, $B' = 0.24$, $J' = 0.00$.

(d) $q = 0.50$, $I' = 0.75$, $Z' = 0.17$, $B' = 0.16$, $J' = 0.02$.

(e) $q = 0.42$, $I' = 0.72$, $Z' = 0.26$, $B' = 0.20$, $J' = 0.00$.

(f) $q = 0.11$, $I' = 0.29$, $Z' = 0.47$, $B' = 0.24$, $J' = 0.01$.

Figure 5.8. The quality score and normalized values of brightness (I'), blurriness (Z'), blockiness (B') and shakiness (J') of the frames selected in Figure 5.7.

while Figures 5.8b and c show views toward the stage from different angles with medium quality, where $q = 0.28$ and $q = 0.31$, respectively. Figure 5.8d shows a frame captured after about 1 sec later than the frame shown in Figure 5.8a, where the brightness and quality has been highly improved with $q = 0.50$. Figure 5.8e has slightly higher quality, $q = 0.42$, than Figure 5.8b mainly due to the appearance of a bright hand in the frame. Figure 5.8f is more blurred, darker and shakier than Figure 5.8c which was captured about 2 sec before by the same camera.

5.2 Cut-point suitability estimation

According to Requirement 2.5 (suitable cut-point), formalized in Section 2.5, the quality of a mashup is dependent on the suitability of the of cut-points of its clips. As a measure of the degree of fulfillment of this requirement, we analyzed different audio and visual features of a recording and assigned a cut-point suitability score. The following sections describe the computation of the cut-point suitability.

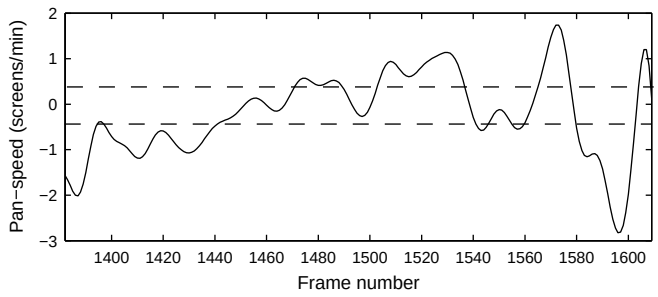
5.2.1 Video cut-points

In general, cut-points in a video indicate a beginning or an end of a sequence of visually similar frames. In the context of a mashup, cut-points indicate switching points among the clips from a multiple-camera recording. In order to find the cut-point suitability score based on video, we used the method described in [Campanella, Weda & Barbieri, 2007], which has been successfully applied in segmenting home-videos for editing. The method is based on camera speed and change in brightness. The estimation of camera speed and brightness are described in Sections 5.1.4 and 5.1.3, respectively. The following sections describe the computation of the video cut-point suitability score.

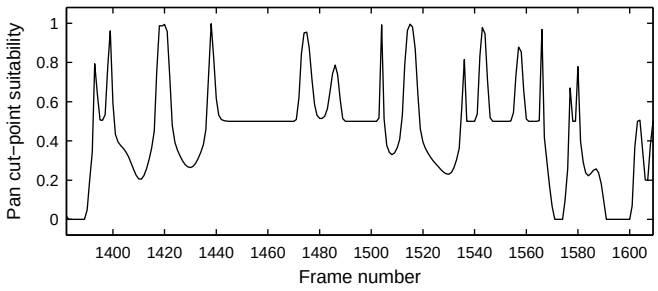
Camera motion

The camera motions, pan and tilt, represent visual changes in the horizontal and vertical directions. Since a cut during a motion is perceived as an abrupt change [Zettl, 2004], video frames corresponding to fast panning and tilting are considered less suitable as cut-points. The most suitable instant for a cut is at the beginning or at the end of the camera motion as a new sequence is going to appear.

In order to compute the cut-point suitability score, two thresholds T^+ and T^- are applied on the positive and negative values, respectively of the camera-speed. The thresholds represent shift of a pixel by the speed equivalent to two screens per minute, where a screen represents the horizontal or vertical dimension a frame in pan or tilt, respectively. For example, if the camera-speed is between the two thresholds, it represents a stable camera and the corresponding frames are assigned a suitability score of 0.5. If the camera-speed is at the thresholds, the corresponding frames are assigned the highest score 1 as they indicate the beginning of a new



(a)



(b)

Figure 5.9. (a) A pan-speed from a recording. The dotted lines indicate the thresholds, such that the frames corresponding above the threshold speeds are considered not suitable for a cut-point. (b) The derived cut-point suitability score (p_c) corresponding to the panning speed shown in (a).

camera pan. To avoid an abrupt high score, a blur-filter is used such that it takes 5 frames to increase to score 1 from 0 and 5 frames to decrease. As the camera speed increases above T^+ or decreases below T^-, the cut-point suitability of the corresponding frames become 0. Based on our qualitative analysis on the test videos, the scores 1 and 0 are spread gradually over 10 frames to allow gradual progression of the score. Figure 5.9 shows a panning curve and a corresponding cut-point suitability score. The computation of the cut-point suitability of the frames based on panning is given by:

$$p_c = \begin{cases} 1 & : \text{ if } \quad p = T^+ \text{ or } p = T^-, \\ 0 & : \text{ if } \quad p > T^+ \text{ or } p < T^-, \\ 0.5 & : \text{ otherwise.} \end{cases} \tag{5.17}$$

The same procedure is applied to compute the cut-point suitability based on the tilt speed represented by t_c. The video cut-point suitability score is computed from the pan, tilt and change in brightness, described in the next section.

Change in brightness

Abrupt changes in brightness are commonly used for shot-cut detection in videos. We computed the change in brightness by subtracting brightness values of consecutive frames in a video. The calculation is given by:

$$\Delta_I = \begin{cases} \left|\frac{I_{i+1}-I_i}{I_{i+1}}\right| & : \text{ if } \quad I_{i+1} \neq 0\,, \\ \\ 0 & : \text{ otherwise.} \end{cases} \tag{5.18}$$

We applied two clipping points T_h and T_l for the brightness difference, such that if the brightness difference is less than T_l, it is considered as no change occurred in the frame and assigned the score 0. If the difference is equivalent of T_l, it is considered as a slight change and assigned the score 0.5. If the difference is higher than T_h we consider the change is significant in the frame and assign the maximum score 1. Based on our observation of the test frames, we selected the values of T_h and T_l as 0.05 and 0.2, respectively, corresponding to 5% and 20% difference in brightness between the frames. In case of frames whose brightness difference is between the two thresholds, the cut-point suitability scores are calculated to be in the range from 0.5 to 1. The calculation can be represented as:

$$I_c = 0.5 + \frac{0.5(\Delta_I - T_l)}{T_h - T_l},$$

where $T_h = 0.2$, $T_l = 0.05$ and $0 \leq \Delta_I \leq 1$.

The factors for computing the video cut-point suitability score: pan-speed, tilt-speed and change in brightness are measured in the range 0 to 1. Based on our experimental evaluation of concerts, the video cut-point suitability is calculated as:

$$c_v = \frac{1}{3}(p_c + t_c + I_c). \tag{5.19}$$

5.2.2 Audio cut-points

In professional music-videos, changes in the visuals are accompanied by changes in the music, see Section 2.1.3 of Chapter 2. The changes may be caused due to changes in speed, instruments, or vocals. The basic time unit of music is the beat, which is an impact sound generally produced by a percussion instrument. The number of beats in a period of time determines the tempo or the speed of the music such that a fast piece of music contains more beats per minute than a slower song.

We tested an available algorithm, described in [Schrader, 2003] to detect beats in our test recordings captured by multiple cameras during concerts. Since the algorithm was designed for studio-recorded high quality audio, it failed to provide satisfactory beat and tempo detection. An option is to use an official audio recording from the concert, which generally contains a high-quality audio.

Based on the results of tempo detection, the video frames corresponding to the tempo changes are assigned the score 1, while other video frames are assigned the score 0:

$$c_a = \{0, 1\}.$$

The video cut-point suitability score range from 0 to 1 and the audio cut-point suitability score is either 0 or 1. Since all the recordings of a multiple-camera recording contain the same audio as the main source, audio provides a measure of cut-point suitability valid to all the recordings, where a video cut-point is dependent on individual cameras. Therefore, to include all the audio cut-points, we computed cut-point suitability score as a maximum of audio and video cut-point suitability score as:

$$\theta(f^v) = \max(c_a, c_v). \tag{5.20}$$

5.3 Diversity estimation

According to Requirement 2.3 (diversity), it is desirable to have variety of content in a mashup. As a measure of the degree of fulfillment of this requirement, we computed a diversity score between two clips based on their visual difference. The visual difference is measured in terms of image distance between two frames corresponding to two clips. We used the method described in [Peters & Fonseca, 2007] as the method had been successfully applied for clustering images and the code was readily accessible.

The method measures the distance between two images based on the differences in their corresponding features. The following six image features are used in calculating the distance:

1. The *Luma* represents the brightness of an image. It is the weighted sum of linear RGB components.
2. The *Color Hue* is given by the H component in the HSV color space, which represents the global distribution of color in the image.
3. The *Dominant color* descriptor is defined by MPEG7 in [MPEG, 2002]. The pixels in an image are clustered in at most 8 colors, representing the dominant colors of the image.
4. The *Color structure* descriptor is also defined by MPEG7 in [MPEG, 2002]. It expresses the local spatial distribution of colors in an image according to a structuring element based on several image samples.
5. The *Color layout* descriptor is also defined by MPEG7. It expresses global spatial distribution of colors in an image based on most significant values for each component Y', Cb and Cr.

6. The *Edges* in 4 directions: horizontal (0 degree), vertical (90 degree), diagonal (45 degree), and anti-diagonal (135 degree) are computed from an image.

The image distance between two images is computed as a linear combination of the distances between the features. If $\alpha(f^v)$ represents an image feature of a frame and $\psi(\alpha(f^v),\alpha(f^{v'}))$ represents the distance between two corresponding features from two frames, then the image distance is given by:

$$\psi(f_x^v, f_y^{v'}) = \sum_{i=1}^{m} w_i \, \psi\left(\alpha_i(f_x^v), \alpha_i(f_y^{v'})\right), \tag{5.21}$$

$$\text{where,} \quad \sum_{i=1}^{m} w_i = 1 \,,$$

and m is the total number of features. The weights w of the features are selected based on the experimental results on a large set of test images, described in [Peters & Fonseca, 2007].

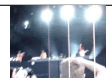
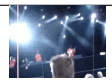
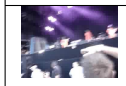
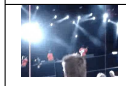
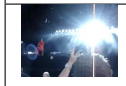
The diversity between two consecutive clips in a mashup is given by the image distance between the last frame of the first clip and the first frame of the second clip. Table 5.1 shows the test frames and the computed image difference between the images. The 3rd and 4th frames have similar color and brightness resulting in the image distance 0.09. However, the 6th and 7th frames have different brightness and different colors, which result in an image distance of 0.40. The figure shows that the image distances calculated by the method correspond to the visual difference in the frames.

5.4 Conclusions

In this chapter, we measured the degrees of fulfillment of Requirements 2.2 (image quality), 2.3 (diversity) and 2.5 (suitable cut-point) in the recordings. We applied different content analysis techniques to evaluate the audio and video features and assign a numeric *score* for each requirement, where the higher score indicates higher degree of fulfillment. The performance of different content analysis techniques in the context of our concert video recordings, have been validated by subjective evaluation.

The measured scores are used to select the most suitable clip for a mashup composition and to evaluate the objective mashup quality. In Chapter 6, we describe algorithm for mashup composition and objective evaluation of the mashup quality. Furthermore, in Chapter 7 we evaluate the perceptual quality of the composed mashups.

Table 5.1. Visualization of image distance in test images. The first column and the first row contain frames from a concert recording. Each numerical values in the table represents image distance between the corresponding orthogonal frames.

										
	0	0.280	0.234	0.177	0.321	0.349	0.297	0.283	0.195	0.324
	0.280	0	0.285	0.286	0.279	0.313	0.326	0.331	0.308	0.199
	0.234	0.285	0	0.090	0.204	0.236	0.382	0.102	0.124	0.341
	0.177	0.286	0.090	0	0.217	0.251	0.329	0.158	0.153	0.349
	0.321	0.279	0.204	0.217	0	0.061	0.377	0.252	0.291	0.349
	0.349	0.313	0.236	0.251	0.061	0	0.403	0.258	0.310	0.359
	0.297	0.326	0.382	0.329	0.377	0.403	0	0.405	0.359	0.348
	0.283	0.331	0.102	0.158	0.252	0.258	0.405	0	0.119	0.366
	0.195	0.308	0.124	0.153	0.291	0.310	0.359	0.119	0	0.342
	0.324	0.199	0.341	0.349	0.349	0.359	0.348	0.366	0.342	0

6

Optimization

We proposed a solution approach for the mashup generation problem in Chapter 3. Then based on the proposed approach, we synchronized the recordings as described in Chapter 4, and analyzed audio and video features as described in Chapter 5. The analysis provides a quantitative measure, *score*, of the fulfillment of the Requirements 2.2 (image quality), 2.3 (diversity) and 2.5 (cut-point suitabil-

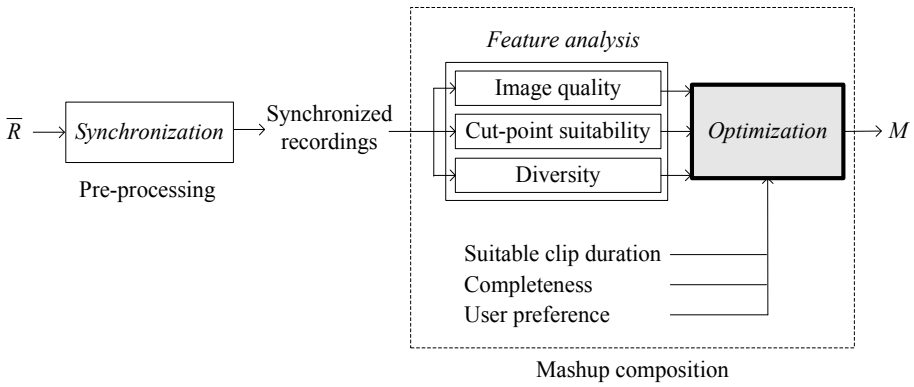


Figure 6.1. Schematic representation of the proposed mashup generation system, in which the optimization step is highlighted.

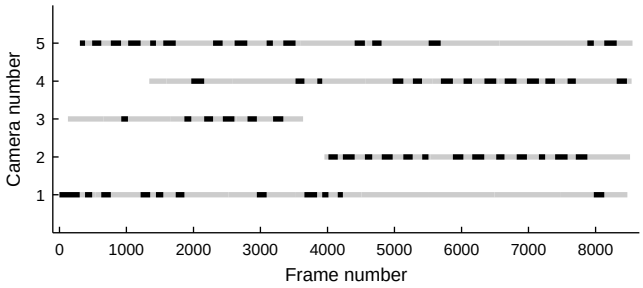


Figure 6.2. A synchronized multiple-camera recording containing five recordings (see Concert 4, Table 6.1 at page 108). The horizontal axis shows a common time-line represented by the frame numbers. The recordings are shown by gray lines. The clips selected for the mashup are shown by dark segments on the recordings.

ity). Figure 6.1 highlights the optimization step in the proposed solution approach for mashup generation. In this chapter, we compose a mashup from a given synchronized multiple-camera recording by selecting clips from different recordings along a common time-line. Figure 6.2 shows an example of the clips selected for a mashup in a multiple-camera recording containing five recordings.

As introduced in Section 3.4, we apply an optimization based approach to select the mashup clips. We developed an objective function according to Section 3.4.1, that provides a quality measure of a mashup by using different scores calculated from the audio-visual feature analysis. Since the scores contain different mean and standard deviation values, we normalize the scores before combining them together in a linear function. The method for normalizing the values obtained from the different feature analysis method is described in Section 6.1. The rest of the chapter is organized as follows. In Section 6.2, we develop the objective function for composing mashups. In Section 6.3, we discuss some known optimization problems similar to the mashup composition problem. Subsequently, in Section 6.4 we develop an algorithm, called *first-fit*, that searches for mashup clips such that the objective function is maximized and the mashup constraints are satisfied. In Section 6.5, we measure the performance of the algorithm in terms of mashup quality given by the mashup score (MS) and compare it with the mashups generated by two other methods, namely *naive* and *manual*. In the final Section 6.6, we discuss the behavior of the mashup quality when changing the weights of the parameters of the objective function.

6.1 Score normalization

The scores computed from the feature analysis in Chapter 5: image quality score, diversity score and cut-point suitability score provide a quantitative measure of fulfillment of the Requirements 2.2 (image quality), 2.3 (diversity) and 2.5 (cut-point suitability), respectively. The score values range between 0 and 1, where a higher score represents a higher degree of fulfillment of the corresponding requirement. However, the mean and standard deviation of the three scores are different. As a consequence, the contribution of the different scores in the objective function, which is a linear combination of the scores, is biased by the score type rather than the degree of fulfillment of the requirement. For example,if the mean value of the image quality score is always higher than that of the diversity score, while the standard deviation values of both scores are low, then combining the two scores will always lead to a bigger contribution of the image quality score. Therefore, we normalized the image quality score, diversity score and cut-point suitability score of a multiple-camera recording to a common average scale of zero and standard deviation given by *Z-score*. Figure 6.3 shows the image quality and cut-point suitability curves of a segment of a recording before and after normalization.

For the Z-score computation, we first calculated the population mean and standard deviation of the three scores from the recordings of a multiple-camera recording. Then the scores were normalized by subtracting from the corresponding population means and dividing by the corresponding standard deviations. For example, if the population mean and standard deviation of the image quality score for a multiple-camera recording is given by $\overline{q}$ and σ_q, respectively, then the computation of the Z-score, $q_z(f^v)$, from the image quality score $q(f^v)$ is given by:

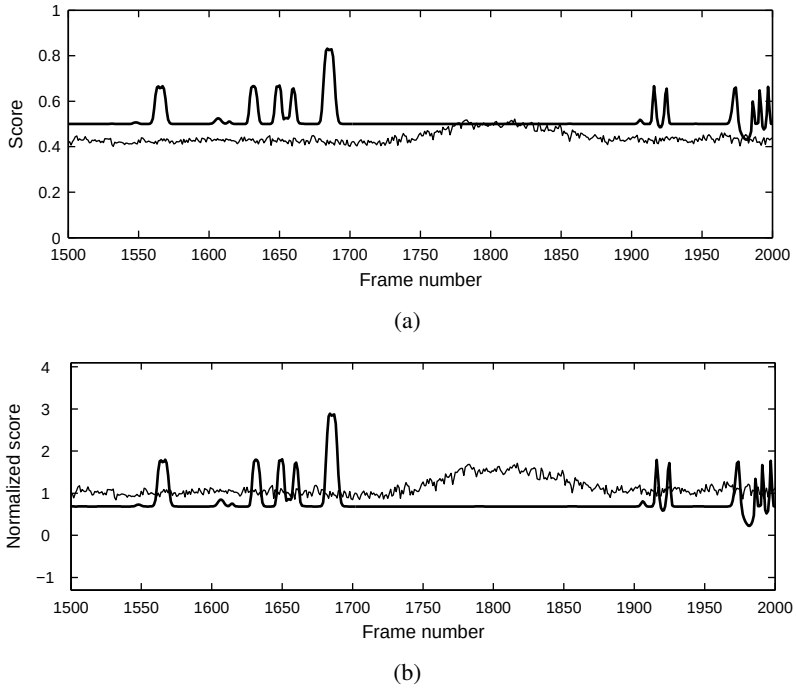
$$q_z(f_i^v) = \frac{q(f_i^v) - \overline{q}}{\sigma_q} , \tag{6.1}$$

$$\text{where} \quad \overline{q} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{j=1}^{n_i} q(f_j^{v_i}) , \tag{6.2}$$

$$\text{and} \quad \sigma_q = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{j=1}^{n_i}\left(q(f_j^{v_i}) - \overline{q}\right)^2} . \tag{6.3}$$

The same method is followed to compute the population mean, standard deviation and the normalized cut-point suitability score by using the scores $\theta(f^v)$, described in Section 5.2.

The diversity score is computed based on the image distance between two video frames corresponding to the clips from two recordings, as described in Chapter 5. The calculation of the population mean and standard deviation becomes very time consuming if we compute the image distance between all the possible pairs of video



(a)

(b)

Figure 6.3. Illustration of the image quality score (thin line) and cut-point suitability score (thin line) from a video segment. (a) Before and (b) after score normalization.

segments in a multiple-camera recording. Therefore, we sampled the synchronized frames and estimated the image distance among the pairs. The samples were uniformly distributed along the recording time. The population mean and standard deviation were calculated from the image distances of m pairs of sampled frames. The value of m was chosen such that the population mean and standard deviation values converge to the calculated values. Based on our observation in the test set, we assigned the value of $m = 500$.

6.2 Objective function

The objective function, as described in Equation 3.4, maximizes the mashup quality given by: image quality $Q(M)$, diversity $\delta(M)$, cut-point suitability $C(M)$, user preference $U(M)$ and semantics suitability $\lambda(M)$. In addition, the mashup M is subjected to satisfy a number of constraints given by the Requirements 2.6 (suitable clip duration), and 2.7 (completeness). In this section, we present our implementation of the objective function for the mashup composition.

The scores for image quality, diversity and cut-point suitability are obtained from the normalized scores as described in Section 6.1. The user preference score, described in Section 3.4.4, is required to be assigned by a user to each of the recordings. Similar to other scores, the preference score ranges between 0 and 1, where an increase in the score value represents a higher preference for the recording. If no user input is given, all the recordings are assigned, by default, the score value of 0.5 indicating that the user preferences for all the recordings are equal and neutral. The semantic suitability requirement is not implemented in our mashup composition solution due to the reasons described in Section 3.6.

In summary, the mathematical model of our mashup composition problem, including the optimization problem and constrains is to:

$$\text{maximize} \quad \text{MS}(M) = a_1 Q(M) + a_2 C(M) + a_3 \delta(M) + a_4 U(M) \ , \tag{6.4}$$

$$\text{subject to} \quad \forall S \in M : d_{\min} \leq d(S_i) \leq d_{\max} \ , \tag{6.5}$$

$$\forall j \in [1,\ldots,N], \exists S_i \in M : S_i \in R_j \ , \tag{6.6}$$

where coefficients a_1, a_2, a_3 and a_4 are used to weigh the contributions of the different requirements. Equations 6.5 and 6.6 represent the constraints suitable clip duration and completeness, respectively. The meanings of the symbols used in the equations are given in Table 3.1 at page 38.

6.3 Related optimization problems

We investigated some optimization problems, which are similar to our mashup composition. Since the problems have been solved with optimal results, we were interested in the used algorithms and if they can be applied to maximize our objective function. In the following paragraphs we present related optimization problems and their relevance in mashup composition.

The *knapsack problem* [Martello & Toth, 1990] is a maximization problem of the best choice of items that can fit into one bag. Given a set of items, each with a weight and a value, the problem is to determine which items to put in the bag so that the total weight is less than a given limit and the total value is as large as possible. The problem is addressed using optimization approaches, such as dynamic programming and greedy [Cormen, Leiserson, Rivest & Stein, 2001]. The greedy approach obtains a solution by making a sequence of choices, each of which seems the best at the point where the choice is made. The dynamic programming approach divides a problem into smaller subproblems and an optimal solution is computed by combining the solutions of the subproblems. In the case of the fractional knapsack problem, where the items can be taken in a fractional amount, the greedy approach can be used to select the items by sorting them according to the cost per a unit of weight. However, in the case of the 0-1 knapsack problem,

where each item must be either taken or left behind, the dynamic programming approach is applied as the greedy approach cannot produce an optimal solution. In the dynamic programming the values of different combinations of the items are calculated to find an optimal solution. In the mashup composition problem, finding the clips from a multiple-camera recording can be viewed as determining items to fit in the case of a fractional knapsack problem. However, the sequence of mashup clips should correspond to a continuous synchronized time-line, while there are no constraints on the order of items in the knapsack problem. Additionally, the mashup composition problem has more requirements than the knapsack problem such as suitable clip duration and completeness.

The *assembly-line scheduling problem* [Cormen, Leiserson, Rivest & Stein, 2001] is about minimizing the total time required to produce an automobile. An automobile should be processed sequentially at a number of stations available in different assembly lines before turning to a finished product. Given work delays at the stations and transportation delays between the stations at different assembly lines, the problem is to find the fastest route that covers all the necessary stations. The problem is solved optimally using dynamic programming. The assembly lines containing a number of stations are analogous to the recordings in a multiple-camera recording containing the clips. The mashup composition problem can be seen as finding the clips with the highest score similar to the stations with the least delay. However, to apply the dynamic programming approach, the duration of the overlapping clips from a multiple-camera recording should be the same. Since different recordings have different suitable cut-points, creating clips of equal duration in all the recordings does not allow maximizing the cut-point suitability score.

The *video summarization problem* involves selecting a set of segments from a given video to represent the video in a temporally condensed form. In [Campanella, 2009] and [Barbieri, 2007] videos are segmented based on criteria like camera motion and scene change. Each segment is represented by a score according to characteristics of the content, such as brightness and presence of faces. The selection of the suitable segments to be included in the summary is treated as a maximization problem. The total score of the summary is maximized while satisfying a number of given constraints like total duration of the segments and distribution of the segments along the time. The *simulated annealing* algorithm is used to search for suitable segments and to maximize an objective function. This algorithm starts with an initial summary and in successive iterations neighboring solutions are created and evaluated. A solution is accepted if it is a better solution than a neighboring solution or depending on the current value of a control parameter. The control parameter is set high at the beginning to accept all the generated solutions. However, at each iteration the control parameter value is decreased, which reduces the chance of accepting a solution worse than a neighboring solution. The

iterations are continued for a predefined number of times or until the control parameter is sufficiently low. The accepted solution is an approximate solution and there is no guarantee that it is optimal. The mashup composition problem can be considered as a multi-dimensional summarization, where the segments come not only from different times but also from multiple recordings. There are additional constraints involved in the mashup composition such as completeness and diversity. Furthermore, the search space becomes extremely large in a multiple-camera recording.

Other maximization problems such as the cutting and packing in production [Dyckhoff & Finke, 1992] and resource allocation in grid networks [Elmroth & Tordsson, 2008] also have some similar constraints as the mashup composition problem. However, in order to satisfy all the constraints in the mashup composition problem the described solutions are not directly applicable.

The mashup composition problem can be solved using different approaches such as greedy and simulated annealing. We choose a greedy approach as it is simple to design and implement with many different requirements. We developed an algorithm called as *first-fit*, which will be described in the next section.

6.4 First-fit algorithm

The First-fit algorithm is developed to select clips from the recordings of a multiple-camera recording such that the mashup score (MS) is maximized and the given constraints are satisfied as described in Section 6.2. The algorithm is based on a greedy approach, which obtains an optimal solution following a sequence of steps, with a set of choices at each step. For the selection of a mashup clip only the segments of the recordings available at that moment are considered. The idea is that if an optimal choice is made for every clip, the resulting mashup becomes optimal. However, the algorithm does not guarantee a globally optimal solution. To reduce the risk of achieving only a locally optimum mashup, we generated a number of mashups by repeating the algorithm starting at random frames. Mashup clips are selected by searching in the forward and backward direction from the starting frame. A mashup with the highest mashup score is taken as the final mashup.

For reasons of simplicity in explaining, we assume one camera-take per recording. Figure 6.4 shows the search method applied in selecting a mashup clip from three hypothetical recordings. Firstly, the candidate clips are determined by checking the availability of the recordings for duration $d_{\max}$ from the starting frame of the mashup clip. Figure 6.5 shows the algorithm for finding available recordings for a clip in forward direction. If there is more than one available recording for the candidate clips and the previous mashup clip belongs to one of the available recordings, then the recording corresponding to the previous clip does not qualify to be

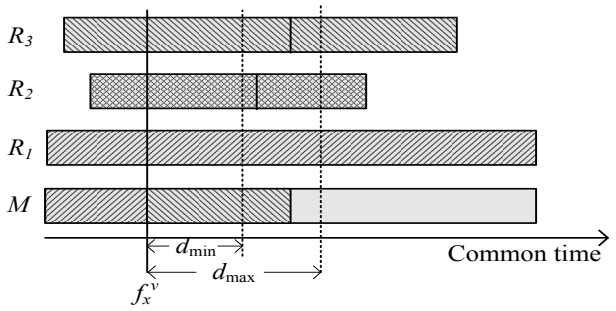


Figure 6.4. Illustration of the clip selection process in the first-fit algorithm. The three recordings R_1, R_2, R_3 and a mashup M to be composed are represented by rectangles on a common time-line. For a clip that begins with f_x^v, the candidate clips are selected from R_2 and R_3 as the previous clip belongs to R_1. The last frame of the candidate clips, shown by a solid line on the recordings, is selected according to the highest cut-point suitability score of the frames located in the interval shown by two dotted lines, which ensures that the candidate clips are longer than $d_{\min}$ and shorter than $d_{\max}$ (Equation 6.5). The candidate clips are evaluated according to the parameters given in the objective function. The highest scoring candidate clip, belonging to R_3, is selected to be included in the mashup.

---

algorithm FIND AVAILABLE RECORDINGS
Input: start frame f_x^v, maximum clip duration $d_{\max}$, multiple-camera recording $\overline{R}$
Output: set of available recordings A
$k \leftarrow 1$
$A \leftarrow \emptyset$
while $(k \leq N)$ **do**
  if $\left(\exists \overline{\tau^k} = (\overline{v}, \overline{a}) \in R_k\right)$ AND $\left(t_s(f_1^{\overline{v}}) \leq t_s(f_x^v)\right)$ AND $\left((t_s(f_x^v) + d_{\max}) < t_s(f_n^{\overline{v}})\right)$
    then $A \leftarrow A \cup R_k$
  end
  $k \leftarrow 1 + k$
end while
return A

Figure 6.5. Algorithm to find available recordings in a given duration to be selected as a clip in forward direction. The function $t_s(f^v)$ returns the time instant corresponding to the frame f^v in a common time-line, defined in Chapter 3.

included as a candidate clip. The length of a candidate clip is selected by choosing a frame with the highest cut-point suitability score, provided the candidate clip is shorter than $d_{\max}$ and longer than $d_{\min}$ (Equation 6.5).

When there is only one candidate clip, the clip is selected as a mashup clip without further calculation. If both the current and previous clips belong to the same recording, then we merge the two clips. In such cases when two clips are merged, the duration constraint (Equation 6.5) is violated. In practice, when no other recordings are available, we allow mashups to contain clips longer than $d_{\max}$. If there is more than one candidate clip, then each of them is evaluated according to the parameters given in the objective function given in Equation 6.4. The candidate clip corresponding to the highest score is selected as a mashup clip. The search process can be initiated from any point in the common time-line, which may require searching for the clips in both forward and backward directions. The algorithms for the backward search require searching for clips in the duration $d_{\max}$ before the starting frame f_x^v.

In order to satisfy the completeness constraint, given in Equation 6.6, there is an additional condition employed in the first-fit algorithm. During the initialization phase of the algorithm, a video frame from each of the recordings, located at least $d_{\max}$ before the last frame of the recording is set as flagged. It is insured that no two flags are within the distance given by $d_{\max}$ to avoid two candidate clips containing the flags. The flag is set first in the shortest recording, followed by the longer recordings to give priority to the shorter recordings. Figure 6.6 shows the algorithm for setting flags in the recordings. A flag is reset when the corresponding recording is included in the mashup. During the mashup clip search, if a candidate clip is encountered with a set flag, the clip is included in the mashup without any measurement. In this way, we insure that all the given recordings contribute at least one clip to the mashup.

There may be instances where it is impossible to fulfill this constraint due to the input recordings in a multiple-camera recording. For example, when there are two very short recordings of length $d_{\max}$ present at the same time such that both are available only for a single clip, or when there are too many clips that it is not possible to accommodate them all in the available time for the mashup. In practice, for all the multiple-camera recordings used in our test we were able to satisfy the completeness criteria. In the present implementation, if a constraint is not satisfied, the mashup is considered invalid. A better approach would be to involve users in the mashup generation process such that if the constraint is not met, the users are notified. Depending on the user response, the mashup is created ignoring the criteria or recalculated. The approach can be implemented in the post-processing step of mashup generation as described in Chapter 3. The first-fit algorithm for searching a mashup clip in the forward direction is explained in Figure 6.7.

algorithm SET FLAGS, assuming one camera-take per recording
Input: maximum clip duration $d_{\max}$, multiple-camera recording $\bar{R}$
Output: $F = (f_{x_1}^{v_1}, \ldots, f_{x_N}^{v_N})$
$F \leftarrow \emptyset, A \leftarrow \bar{R}$
while $|A| > 0$ **do**
    find $R_k \in A : d(R_k) \leq d(R_i) \, \forall R_i \in A$
    setFlag ←**false**
    if $|F| = 0$
        then setFlag ← **true**
    end if
    find $f_i^{v_k} : t_s(f_i^{v_k}) = t_s(f_n^{v_k}) - d_{\max}$ with $\tau_k = (v_k, a_k) \in R_k$
    do while NOT setFlag AND $t_s(f_i^{v_k}) \geq t_s(f_1^{v_k})$
        for each $f_{x_j}^{v_j} \in F$ **do**
            if $|t_s(f_{x_j}^{v_j}) - t_s(f_i^{v_k})| \leq d_{\max}$
                then setFlag ← **true**
            else $i \leftarrow i - 1$
            end if
        end for
    end while
    if NOT setFlag
        then QUIT, no solution
    end if
    $x_k \leftarrow i, f_{x_k}^{v_k} \leftarrow f_i^{v_k}$
    $F \leftarrow F \cup \{f_{x_k}^{v_k}\}$
    $A \leftarrow A \backslash \{R_k\}$
end do
return F

Figure 6.6. Algorithm to add flags to the recordings for satisfying completeness constraint.

algorithm FIRST-FIT MASHUP GENERATION, assuming one camera-take per recording
Input: minimum clip duration $d_{\min}$, maximum clip duration $d_{\max}$, multiple-camera recording $\bar{R}$, start frame f_x^v, a_1, a_2, a_3, a_4
Output: $M = (S_1, \ldots, S_l)$
$M \leftarrow \emptyset$, $S_p \leftarrow \emptyset$, $F \leftarrow$SET FLAGS$(\bar{R})$
$A \leftarrow$FIND AVAILABLE RECORDINGS$(f_x^v, d_{\max}, \bar{R})$
while $|A| > 0$ **do**
  if $M \neq \emptyset$ AND $|A| > 1$
    then $A \leftarrow A \backslash \{R_p\}$
  end if
  if $\exists f_{x_k}^{v_k} \in F : t_s(f_{x_k}^{v_k}) - t_s(f_x^v) \leq d_{\max}$
    then $A \leftarrow \{R_k\}$
  end if
  for each $R_k \in A$
    find frames $f_c^{v_k}$, $f_d^{v_k}$, $f_e^{v_k}$, $f_j^{v_k}$ with $\tau_k = (v_k, a_k) \in R_k$ such that:
      $t_s(f_c^{v_k}) = t_s(f_x^v), \quad t_s(f_d^{v_k}) = t_s(f_x^v) + d_{\min}$
      $t_s(f_e^{v_k}) = t_s(f_x^v) + d_{\max}, \quad \theta(f_j^{v_k}) = \max[\theta(f_j)], d \leq j \leq e$
    $S_k \leftarrow (s^a, s^v)$ with $s^v \leftarrow (f_c^{v_k}, f_j^{v_k})$
    and s^a is the corresponding audio segment
    $\mathrm{MS}_k \leftarrow a_1 C(S_k) + a_2 Q(S_k) + a_4(S_k)$
    if $S_p \neq \emptyset$ **then** $\mathrm{MS}_k = \mathrm{MS}_k + a_3 \delta(S_k, S_p)$ **end if**
  end for
  determine q such that $\mathrm{MS}_q = \max(\mathrm{MS}_i)$, $1 \leq i \leq |A|$
  determine y such that $t_x(f_y^v) = t_s(f_j^{v_q})$ with $S_z = (f_c^{v_q}, \ldots, f_j^{v_q})$
  if $q = p$
    then $S \leftarrow \mathrm{merge}(S, S_p)$, $M \leftarrow M \backslash \{S_p\}$
  end if
  if $\exists f_{x_q}^{v_q} \in F$
    then $F \leftarrow F \setminus \{f_{x_q}^{v_q}\}$
  end if
  $M \leftarrow M \cup S$, $S_p \leftarrow S$, $p \leftarrow q$, $x \leftarrow y + 1$
  $A \leftarrow$FIND AVAILABLE RECORDINGS$(f_x^v, d_{\max}, \bar{R})$
end while
return M

Figure 6.7. Algorithm for first-fit mashup generation in forward direction. The function $t_s(f^v)$ returns the time instant corresponding to the frame f^v in a common time-line, defined in Chapter 3.

6.5 Mashup quality: objective evaluation

The quality of a generated mashup is computed using the objective function given by Equation 6.4. We compared the quality of the mashups generated by the first-fit algorithm with two other methods: *naive* algorithm and *manual*. In the following sections we describe the methods and their mashup qualities compared to that of the first-fit algorithm.

6.5.1 Naive algorithm

The naive algorithm is designed to generate a mashup which fulfills the constraints given in Equations 6.5, and 6.6 derived from Requirements 2.6 (suitable-clip duration) and 2.7 (completeness), respectively. No other requirements are considered during the mashup generation.

The naive algorithm generates a mashup as follows. The starting point is always the very first frame on the common time-line. As described in Figure 6.5, the available recordings are searched within the given maximum clip duration $d_{\max}$. If there is more than one available recording for the candidate clips and the previous mashup clip belongs to one of the available recordings, then the recording corresponding to the previous clip does not qualify to be included as a candidate clip. If there is only one available recording, it is selected as the mashup clip. However, if multiple recordings are available, one of the recordings is selected randomly. Once a recording is selected, the last frame of the clip is selected randomly among the frames that are located between $d_{\min}$ and $d_{\max}$ from the starting frame of the clip, such that the clips are within the suitable clip duration (Equation 6.5). The algorithm assures that a mashup contains at least one clip from each of the recordings, to satisfy Equation 6.6 (completeness), by using flags as described in Section 6.4. Figure 6.8 explains the naive algorithm.

Test set

In order to compare the mashup quality generated by the naive algorithm and the first-fit algorithm, as a test-set, we used 10 multiple-camera recordings captured using non-professional cameras during concerts. Each of the multiple-camera recordings contained 3 to 9 recordings with both audio and video streams. The duration and the time overlap among the recordings in the test-set are shown in Figure 6.9. The recordings were synchronized using audio-fingerprints as described in Chapter 4. We could have used the audio-onsets, but when it was developed the naive mashups were already generated. All the recordings were longer than the duration $d_{\max}$ and had the video frame rate of 25 frames per second. The recordings were obtained from YouTube.

In order to generate mashups, both the first-fit and naive algorithms were given the following values for minimum and maximum clip duration: $d_{\min} = 3$ sec and

algorithm NAIVE MASHUP GENERATION, assuming one camera-take per recording
Input: minimum clip duration $d_{\min}$, maximum clip duration $d_{\max}$,
multiple-camera recording $\overline{R}$, start frame f_x^v
Output: $M = (S_1, ..., S_l)$
$M \leftarrow \emptyset$
$A \leftarrow$ FIND AVAILABLE RECORDINGS(f_x^v, $d_{\max}$, $\overline{R}$)
$F \leftarrow$SET FLAGS($\bar{R}$)
while $|A| > 0$ **do**
  if $M \neq \emptyset$ AND $|A| > 1$
    then $A \leftarrow A \setminus \{R_p\}$
  end if
  if $\exists f_{x_k}^{v_k} \in F : t_s(f_{x_k}^{v_k}) - t_s(f_x^v) \leq d_{\max}$
    then $A \leftarrow \{R_k\}$
  end if
  $R_z \leftarrow$ RANDOM(A)
  determine c and d such that:
    $t_s(f_c^{v_z}) = t_s(f_x^v)$ and
    $t_s(f_d^{v_z}) = t_s(f_x^v) + d_{\min} + \text{rnd}(d_{\max} - d_{\min})$
  $\tilde{S} \leftarrow (s^{a_z}, s^{v_z})$ with $s^{v_z} \leftarrow (f_c^{v_z}, \ldots, f_d^{v_z})$
  and s^{a_z} is the corresponding audio segment.
  $M \leftarrow M \cup \tilde{S}$
  $p \leftarrow z$
  determine y such that: $t_s(f_y^v) = t_s(f_d^{v_z})$
  if $\exists f_{x_z}^{v_z} \in F$
    then $F \leftarrow F \setminus \{f_{x_z}^{v_z}\}$
  end if
  $x \leftarrow y + 1$
  $A \leftarrow$ FIND AVAILABLE RECORDINGS(f_x^v, $d_{\max}$, $\overline{R}$)
end while
return M

Figure 6.8. Algorithm for naive mashup generation. The function RANDOM(A) returns an element contained in set A, with uniform distribution. The function rnd(x) returns a random integer between 0 and x, with uniform distribution and the function $t_s(f^v)$ returns the time instant corresponding to the frame f^v in a common time-line, defined in Chapter 3.

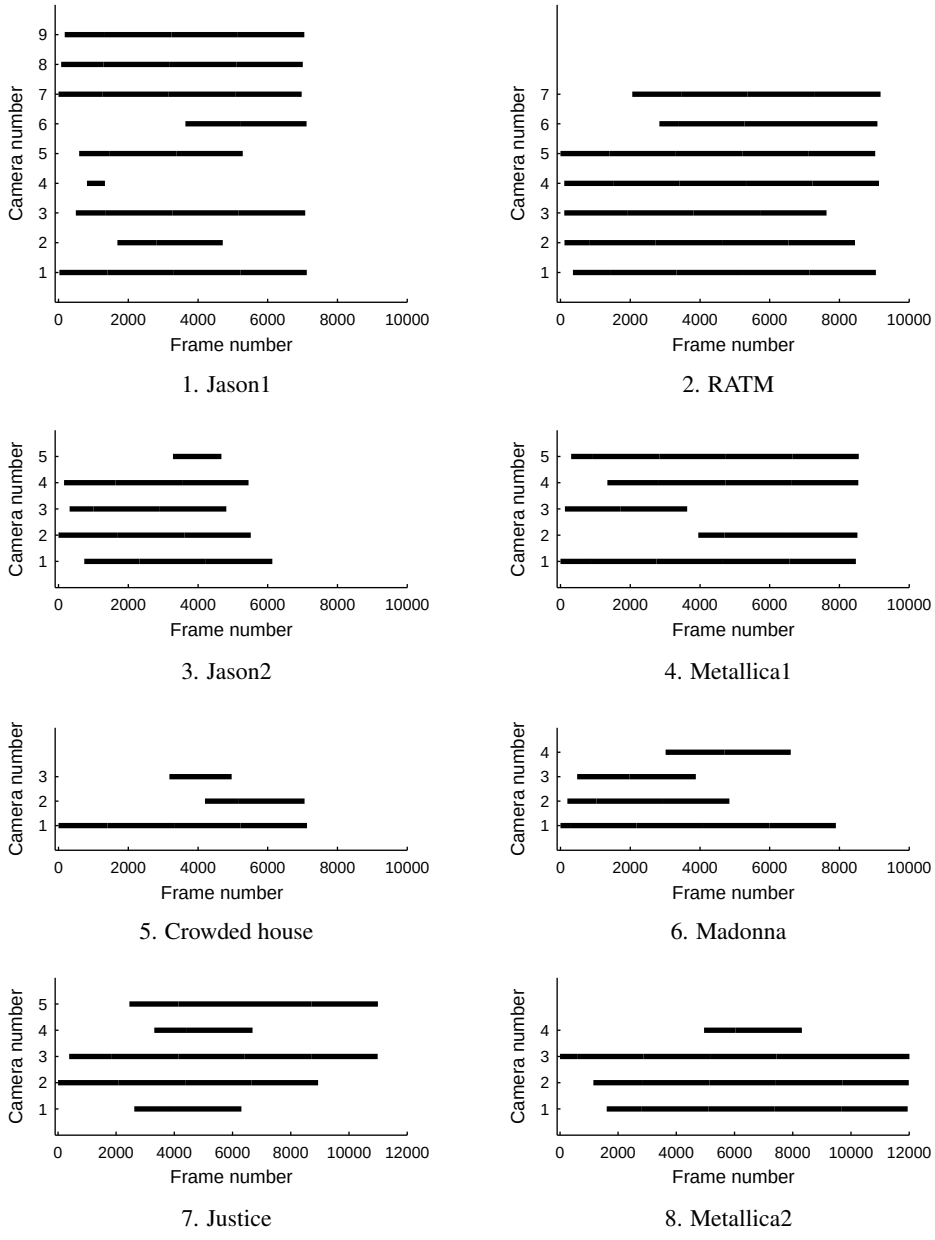
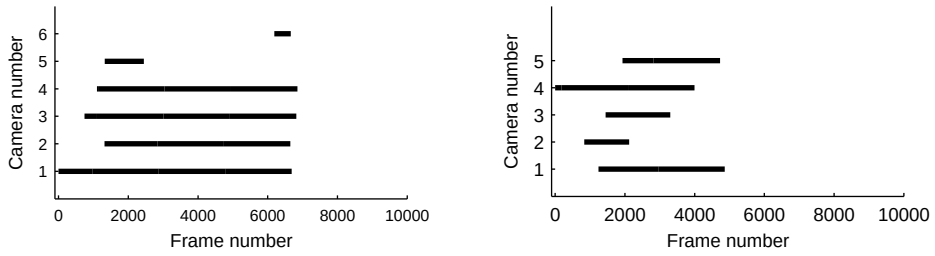


Figure 6.9. Multiple-camera recordings used as a test-set for evaluating *naive* and *first-fit* mashups. The recordings are represented by the bold dark horizontal lines on a common time-line. The camera numbers represent the index of the recordings. The remaining multiple-camera recordings are presented on the next page.



9. Foo fighters 10. Kaiser chiefs

Figure 6.9. Continuation of the figure from the previous page.

$d_{\max} = 7$ sec. The durations were chosen based on a common practice in amateur video-editing, which assumes that it takes at least 3 sec to understand an event and it becomes monotonous to watch a scene longer than 12 sec. Since the musical genre of our test concert recordings are of rock and pop, to match with their fast tempo we selected the maximum duration of 7 sec.

The recordings were analyzed and image quality and diversity scores were measured according to the methods described in Section 5.1 and 5.3. The video cut-point scores were calculated using the method described in Section 5.2, however, we could not produce reliable tempo detection using an available algorithm, described in [Schrader, 2003]. Since the algorithm was designed for studio-recorded high quality audio, it failed to provide satisfactory result in non-professional concert recordings. Therefore, we manually annotated the cut-points by listening to the recorded music. The manual annotations represent perceptual changes in music based on the opinion of a single subject. Among the synchronized multiple-camera recordings, we chose a recording with a higher quality audio for annotating the audio cut-points. The following rules were followed while annotating:

1. There should be a noticeable change in the audio, which lasts at least three seconds. If it is less than three seconds, it cannot be perceived. Examples of typical audio cut-points are transitions from solo to instrumental or chorus.
2. Beats are difficult to count and track accurately for a general music listener. Therefore, beats and tempo were not consciously considered for annotating the cut-points.
3. If there was a silence longer than three seconds, for example between chorus and solo, two cut-points were annotated at the beginning and end of the silence.

To have an objective comparison between the first-fit and naive algorithms, no user preference scores were provided to the recordings in the first-fit algorithm. Therefore, we ignored the preference score from our implementation of the objec-

Table 6.1. Evaluation of first-fit and naive mashups in the given test-set.

Concert	**# Cameras**	**Duration**	MS **First-fit**	MS **Naive**
1. Jason1	9	20 – 283 sec	1.202	-0.005
2. RATM	7	250 – 360 sec	0.893	-0.007
3. Jason2	5	51 – 179 sec	0.975	0.029
4. Metallica1	5	140 – 339 sec	0.829	-0.035
5. Crowded house	3	71 – 285 sec	0.680	0.014
6. Madonna	4	136 – 316 sec	0.720	-0.017
7. Justice	5	135 – 424 sec	0.706	-0.006
8. Metallica2	4	134 – 494 sec	0.639	-0.014
9. Foo fighters	6	19 – 268 sec	0.867	-0.012
10. Kaiser chiefs	5	51 – 179 sec	0.611	-0.025

tive function given in Equation 6.4. The coefficients a_1, a_2 and a_3 in the objective function were set to be equal to $\frac{1}{3}$. The algorithm was repeated 20 times, each time starting from a random location and the solution corresponding to the maximum MS was accepted as the mashup.

Results

The mashup score (MS) of the mashups generated by the first-fit and naive algorithms are shown in Table 6.1. The mashup scores were calculated using the objective function given in Equation 6.4. The results show that the quality of the mashups generated by the first-fit algorithm is consistently, at least 10 times higher than that of the mashups produced by the naive algorithm. The difference in the mashup quality score between the first-fit and naive mashups was mainly caused by the cut-point score followed by the diversity score and the quality score.

6.5.2 Manual

The manual mashups were created by a professional video editor. The synchronized multiple-camera recordings were provided to the editor and he was asked to create a mashup without adding any special effects and following the time-line of the content. It took approximately 16 hours to create 3 mashups using the editing software Adobe Premiere Pro compared to a couple of seconds for generating naive and first-fit mashups.

Table 6.2 shows the details of the concerts used as test-sets for creating the manual mashups. Considering the complex and time consuming process of creating manual mashups, we chose only three concerts containing 4 – 5 cameras from the concerts, given in Table 6.1. For example, in concert C1 we selected only 5 out of 10 recordings from concert Jason1, given in Table 6.1. The number of cameras in all the concerts were chosen to be similar to have a fair comparison among the different concerts. The test-sets represent typical concert recordings covering different physical settings, light-conditions and size of the audience.

Table 6.2. Multiple-camera recordings used for evaluating first-fit, naive and manual mashups. The camera numbers correspond to the concert representations in Figure 6.9.

Concert	Camera #	Duration	Audience	Venue	Genre
C1. (Jason1)	1, 3, 7, 8, 9	20 – 283 sec	<500	Indoor	Rap
C2. (RATM)	1, 2, 3, 4, 5	250 – 360 sec	>1000	Outdoor	Metal
C3. (Madonna)	1, 2, 3, 4	136 – 316 sec	>1000	Indoor	Pop

In order to compare the quality of the mashups generated by the different methods, we generated first-fit and naive mashups from the test-set as described in Section 6.5.1. The multiple-camera recordings and the naive, first-fit and manual mashups are made publicly available in YouTube, which can be accessed via the link given in [YouTube, 2009a].

The quality of the mashups generated by the first-fit, naive and manual methods were evaluated using Equation 6.4. Table 6.3 shows the mashup quality score (MS) from the given test-set, which increases in the order: naive, manual and first-fit. The lowest score for a naive mashup is expected as it does not satisfy the image quality, cut-point suitability and diversity requirements. We also expected that the quality of a manual mashup should be the highest since the requirements for a mashup can be better understood by a human editor than a model. However, the results show the quality of a manual mashup is lower than a first-fit mashup.

The fact that the scores of the first-fit mashups are higher than that of the manual mashups may be due to several reasons. Firstly, the requirements compiled for the mashup generation may not be complete. Furthermore the automatic mashup addresses only a number of these requirement, and it may have missed some important ones. Secondly, the techniques used in feature analysis may have failed to accurately compute the image quality, cut-point suitability and diversity scores of the test-set. Thirdly, the objective function maybe non-linear or the coefficients a_1, a_2 and a_3 should not be assigned equal values. Lastly, the manual mashups may not fulfill the user requirements as they may be too difficult to fulfill manually. Since both the manual and first-fit algorithms may have addressed different requirements or with different importance, it is difficult to compare the mashup quality between the two algorithms by means of an objective evaluation. Therefore, we conducted a user study that provides a perceptual evaluation of the naive, first-fit and manual mashups. The study will also validate if the feature analysis results correspond to human perception. The design of the user study and the analysis of the results is presented in Chapter 7. In order to check the behavior of MS on changing the coefficient values, we analyzed the values of the coefficients used in the manual mashup. The analysis method and the results are discussed in the following Section 6.6.

Table 6.3. Evaluation results of the naive, manual and first-fit mashups.

Concert	MS **Naive**	MS **Manual**	MS **First-fit**
C1	0.077	0.087	1.09
C2	-0.003	0.13	0.83
C3	-0.017	0.054	0.72

6.6 Discussion

In the present implementation of the first-fit algorithm, we used equal weights for the three mashup requirements: image quality, cut-point suitability and diversity represented by the coefficients a_1, a_2 and a_3, respectively. In order to check if we can tune the weights, so that the MS of both the manual and the first-fit mashups can be maximized, we started by analyzing the distribution of weights in the manual mashups. For every clip included in the manual mashups, we computed the image quality, cut-point suitability and diversity scores from the corresponding recordings. These average scores in the manual mashups from C1, C2 and C3 contribute to the objective function as follows:

$$a_1 0.16 - a_2 0.18 + a_3 0.29 = \text{MS}_{\text{C1}}\ , \tag{6.7}$$

$$a_1 0.21 + a_2 0.27 - a_3 0.09 = \text{MS}_{\text{C2}}\ , \tag{6.8}$$

$$-a_1 0.002 + a_2 0.05 + a_3 0.11 = \text{MS}_{\text{C3}}, \tag{6.9}$$

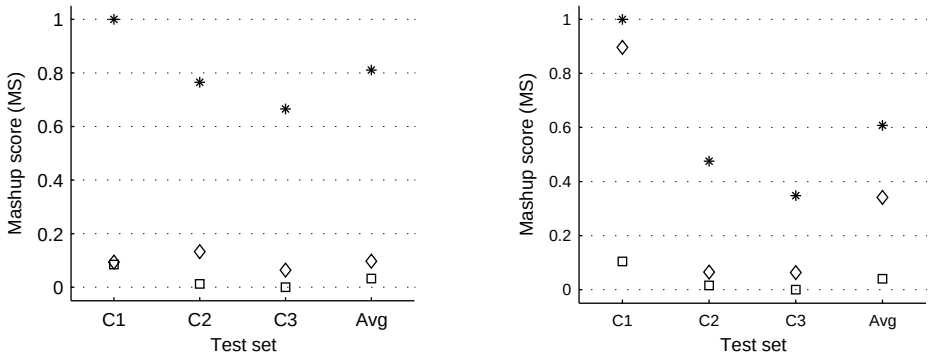
where $a_1 + a_2 + a_3 = 1$.

We would like to find the values of the coefficients a_1, a_2 and a_3 that can be used to maximize the MS in all three mashups. In Equations 6.7 and 6.9, the MS will be maximized if the weight of the diversity score is maximum and the weights of the image quality and cut-point suitability score are minimum, $(a_1 = 0, a_2 = 0, a_3 = 1)$. However, in Equation 6.8 the MS will be maximized if the image quality score is weighed with the highest value, $(a_2 = 1, a_1 = 0, a_3 = 0)$. There is no consistency in the order of importance among the requirements because diversity is the most important requirement in Equations 6.7 and 6.9, while image quality is the most important in Equation 6.8.

In order to further analyze the behavior of the mashup scores, we generated the naive and first-fit mashups in the test-set, given in Table 6.2, using different sets of coefficients. The same sets of coefficients were also applied to the manual mashups given in Equations 6.7–6.9. Figure 6.10a shows the mashup scores of the test-set in terms of naive, manual and first-fit mashups, when $a_1 = a_2 = a_3 = \frac{1}{3}$. In all three concerts, the mashup score increases in the order: naive, manual and first-fit. The first-fit algorithm scores the highest with the minimum and maximum score of 0.6 and 1. The score of the manual algorithm ranges between 0.05 and 0.13, and the scores of the naive ranges between -0.003 and 0.07. Figure 6.10b shows the MS values when $a_2 = 0$ and $a_1 = a_3 = \frac{1}{2}$. The MS value increases in

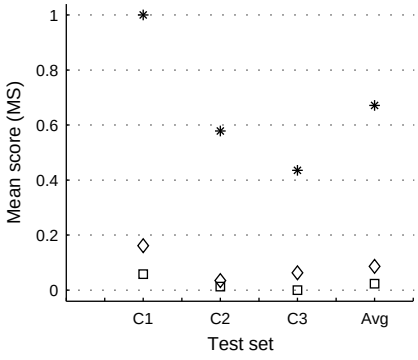
the order: naive, manual and first-fit, similar to the previous set of coefficients. However, the difference between the manual and first-fit is reduced in all three test-sets. For example, in C1 the MS of the first-fit and manual mashups are 1 and 0.9, respectively. In C2 and C3, the first-fit algorithm has scores 0.5 and 0.4 respectively. Figure 6.10c shows the MS values when $a_2 = \frac{1}{9}$ and $a_1 = a_3 = \frac{4}{9}$. In all three concerts, the mashup score increases in the order: naive, manual and first-fit as in the previous sets of coefficients. The first-fit algorithm scores the highest with the minimum and maximum score of 0.4 and 1. The score of the manual algorithm ranges between 0.06 and 0.16, and the scores of the naive ranges between 0 and 0.05. In all the cases, the average score of the mashups was in the order: naive, manual and first-fit.

The behavior of the MS in different sets of coefficients is dependent not only on the methods of generating the mashups but also on the test-sets. A set of coefficients may increase the mashup score of a mashup in one test-set, while in another it may cause the opposite effect. Using different sets of coefficients did not change the order of the mashup scores and there is not an a-priori reason for which one of the features should be more important. For a more precise determination of the coefficients from manual mashups, a larger number of test-sets would be required. Therefore, based on this analysis, we used the equal value for all the coefficients in the objective function.



(a) $a_1 = a_2 = a_3 = \frac{1}{3}$

(b) $a_2 = 0$ and $a_1 = a_3 = \frac{1}{2}$



(c) $a_2 = \frac{1}{9}$ and $a_1 = a_3 = \frac{4}{9}$

Figure 6.10. The MS value of the mashups generated for the test-sets given in Table 6.2, with different coefficient values. The symbols □, ◇ and ∗ represent naive, manual and first-fit mashups, respectively. The horizontal axis shows the three test-sets and their average score.

7

Mashup quality: subjective evaluation

In the previous chapter we generated an automatic mashup using the *first-fit* algorithm, which is based on fulfilling the requirements for generating mashups elicited in Chapter 2. Then we evaluated the objective quality of the first-fit mashups with respect to the *naive* and *manual* mashups. A naive mashup is a random selection of synchronized video clips within a given duration and a manual mashup is a creation of a professional video-editor. In this chapter, we describe a subjective test conducted to measure the perceived mashup qualities by end-users.

The rest of the chapter is organized as follows. In Section 7.1, we present the hypotheses and their operationalizations upon which the subjective evaluation is based. In Sections 7.2 and 7.3 we describe the design and procedure of the subjective test, respectively. In Section 7.4 we present the results of the test and then discuss the perception of the mashup qualities based on the test results in Section 7.5. We conclude the subjective evaluation in Section 7.6.

7.1 Hypotheses and operationalizations

The goal of the test is to compare the perceived quality of mashups generated by different methods: first-fit algorithm, naive algorithm and manual. Ideally, we expect the perceived quality of a mashup to be in the ascending order: naive, first-fit, and manual. According to the objective evaluation as described in Section 6.5.2,

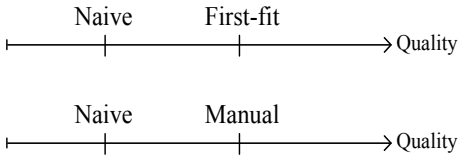


Figure 7.1. Hypotheses on the perceived quality of the mashups generated by naive, first-fit and manual mashups on a one-dimensional scale.

the naive mashups score the lowest, however, the first-fit mashups score higher than the manual mashups. Therefore, from the user test we want to check if the quality of the naive mashups are perceived as the lowest and how similar are the perceived qualities of the first-fit and manual mashups. The formulated hypotheses for the test are:

H1. The perceived quality of a mashup generated by the first-fit algorithm is higher than that of one generated by the naive algorithm.

H2. The perceived quality of a manually made mashup is higher than that of one generated by the naive algorithm.

H3. The perceived quality of a manually made mashup is equal to that of one generated by the first-fit algorithm.

On a one-dimensional scale, the hypotheses on the perceived quality of the naive mashups against the first-fit and manual mashups are visualized in Figure 7.1. The *perceived quality* is an abstract concept and we need to define it in terms of measurable factors (operationalization). According to the mashup requirements elicited in Chapter 2, we operationalized the perceived mashup quality into three factors: *diversity, visual quality,* and *pleasantness*. The factors are further divided into different parameters, which are easy for the users to judge, for example image quality, entertaining, variety. The user response on the perception of these parameters provides a quality measure on the corresponding factors, which will lead to the verification of the hypotheses. The following paragraphs describe the factors and their measuring parameters.

7.1.1 Diversity

In a concert recording, users desire to watch not only the artists at stage but experience the concert atmosphere as a whole (Requirement 2.3). A *high quality* mashup should be able to provide a rich coverage of different aspects of a concert. Based on the keywords used to associate diversity by the participants of our focus group study, described in Chapter 2, we represent diversity in terms of the following parameters: *atmosphere, overview*, and *content variety*. The atmosphere signifies the mood during the concert such as dull, enjoyable and wild. The

overview signifies the physical settings, such as location, audience size and the content variety signifies richness in the content. A *high quality* mashup should score high on atmosphere, overview and content variety.

7.1.2 Visual quality

Visual quality is a desirable feature of any video for clarity and understanding of content (Requirement 2.2). It is an obvious and easily perceivable feature. However, visual quality depends on multiple criteria such as brightness, edge blur, and noise, which are difficult to differentiate and evaluate for a general user. We selected two parameters: *image quality* and *camera stability* because they were the keywords used to associate visual quality by the participants of our focus group study, described in Chapter 2. The image quality signifies the perception of spatial features like brightness, blockiness and blur. The camera stability signifies the perception of spatial and temporal feature such as motion along the frames. A *high quality* mashup should measure high on both image quality and camera stability.

7.1.3 Pleasantness

One of the main purposes of watching a mashup is to have a pleasant experience, as given in the objective of creating a mashup in Section 1.1 of Chapter 1. There are several ways to express this experience, which is very subjective. Based on the keywords used to associate pleasantness by the participants of our focus group study, described in Chapter 2, we represent pleasantness in terms of the following parameters: *boring, overall goodness*, and *entertaining*. A boring mashup is considered the opposite of entertaining, which could be due to the failure of many requirements such as image quality, diversity, suitable cut-point and suitable clip-duration. An overall goodness signifies that the mashup fulfills the requirements in a satisfactory level. A *high quality* mashup should measure low in boring and high in overall goodness and entertaining.

7.2 Test Design

7.2.1 Test variables

The test contained two independent variables. The first independent variable was *algorithm*, which refers to the *naive, manual,* and *first-fit* methods for generating mashups. The second independent variable was *content*, which were the concerts **C1, C2** and **C3** shown in Table 6.2 at page 109. In our observation, the perceived quality of a mashup also depends on factors such as length of the recordings, num-

ber of cameras, and image resolution. For example, a mashup is more likely to contain more variety in content and high image quality if there are large number of recordings. The influence of these factors were minimized in the test by using *content* with similar factor values. In all concerts the duration of the recordings was between 2.4 min to 5.6 min, the number of cameras were 4 or 5 and the screen resolution was 320×240 pixels.

7.2.2 Material

The test-set used in the subjective test is the same as the one used in the objective test. The description of the multiple-camera recordings, size of the audience and musical genre are presented in Section 6.5.2 and Table 6.2 in Chapter 6.

7.2.3 Full-factorial within-subject test design

The test was designed as full-factorial within-subject such that all the nine mashups were evaluated by every participant. The advantages of this design are that all the main effects of the variables and their interactions can be estimated with a limited number of participants and the interpersonal differences do not influence the evaluation. A drawback of this design is that the test run time per participant increases with the number of test variables. In our case, since we had nine mashups, each about 5 min long, for each participant the test run would take about 45 min to play the mashups and about 20 min for answering questionnaire, which was an accepted duration for a perception test.

In such a test design, participants may be biased in their perception by the order of the mashups they watch (*order effect*), or by the memory mix-ups from the contents of previous mashups (*carry-over effect*). In order to minimize the influence of these effects, the presentation order of the mashups was *balanced*.

For every participant a balanced mashup presentation order was generated at the beginning of the test. First, the order of *content* and *algorithm* was chosen randomly from six corresponding permutations. Then a balanced mashup presentation order is generated according to the chosen order of *algorithm* and *content*. For example, if the chosen algorithm and content are in order *manual, first-fit, naive* and **C3, C1, C2** respectively, the balanced presentation order consists of mashups: manual-C3, manual-C1, manual-C2, first-fit-C3, first-fit-C1, first-fit-C2, naive-C3, naive-C1, naive-C2. In this mashup presentation order the distance between the same concerts was always maximum to reduce the carry-over effect, and different participants viewed the mashups in different order, to reduce the order effect.

7.2.4 Participants

A statistical estimation of the required number of participants was not possible because the population distribution with respect to the test hypothesis was not known.

However, in a similar user test conducted to evaluate quality of video summaries generated by three different methods the hypotheses were tested successfully using 40 participants, described in [Barbieri, 2007]. Therefore, we set the number of participants to 40.

The mashup perception might be influenced by the age of a participant due to the popularity of an artist among a certain age group. Therefore, we selected a target group of the participants with age in between 20 and 30. The participants were required not to be involved in developing any of the algorithms for mashup generation to avoid bias in the evaluation due to prior knowledge.

7.2.5 Scaling

The test involved *scaling* the mashup, which was conducted by showing participants one mashup at a time, followed by a questionnaire that rates different parameters of mashup quality perception. The advantage of this design is that the participants do no require to recall and compare an other mashup. A test design involving direct comparison among the mashups, such as ranking and paired comparison, was impractical because that would require a participant to watch at least two mashups before answering. If the two mashups were shown in parallel in a split screen, it would be difficult to focus on two different mashups at a time. If they were shown one after the other it would take about 10 min to watch, which would make it difficult to remember specific parameters from the two mashups.

A disadvantage of *scaling* is that the quality scale is unknown at the beginning of the test. To give an idea of the visual quality of the content, we showed four recordings of a concert in a split screen whose quality vary along the time. Figure 7.2 shows a screen shot containing four recordings in a split screen. However, showing an example of the highest or lowest quality mashup is difficult due to the subjective perception of a mashup. The effect is reduced with enough participants and a balanced ordering of the mashup presentation.

7.2.6 Questionnaire

A questionnaire was designed to measure the effect of the different parameters described in Section 7.1 on the different mashups. The parameters represent the mashup requirements elicited in Chapter 2. In total, nine statements were presented to the participants and for each statement the participants had to indicate the level of agreement or disagreement. The level of agreement (from strongly agree to strongly disagree) was given by a popular seven point Likert scale used extensively in perception tests. The statements presented were as follows:

S1. I got a good impression of the concert atmosphere from the mashup video.

S2. The different viewpoints shown in the video gave me a rich overview of the concert.

Figure 7.2. Screen-shot of a concert video recording with four cameras shown in a split screen.

S3. I got disturbed by the lack of camera stability.

S4. I found this video entertaining.

S5. I think there was enough variety in the content.

S6. I found the video boring to watch.

S7. The cameras in the video were stable.

S8. The image quality in the video was very bad.

S9. Overall, I think the video was good.

Statements **S1, S2** and **S5** were aimed at measuring diversity (Section 7.1.1). Statements **S7** and **S8** were aimed at measuring visual quality (Section 7.1.2). Statement **S3** was asked as a control measure to check if the perception of the parameter measured in **S7** reflects a subjective measure. Statements **S4, S6** and **S9** were aimed at measuring pleasantness (Section 7.1.3). The order of the statements were arranged so that the questions, which can be interpreted to be very similar, are not asked consecutively to avoid answering on memory.

7.2.7 Control variables

In addition to *algorithm* and *content* the perception of a mashup quality might be influenced by other factors, such as age, gender, artist, genre, frequency of concert visits. In order to check the effect of these factors, some additional questions were asked to the participants.

The age and gender of every participants were registered. At the beginning of the questionnaire, described in paragraph 7.2.6, three statements were added to in-

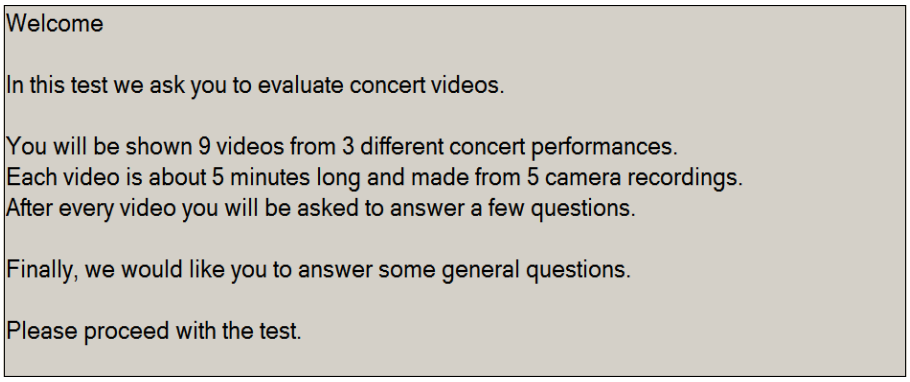


Figure 7.3. Welcome window grabbed from the interface used in the test.

quire the participant's opinion about the artist and the genre for each of the concert recordings. The level of agreement (from strongly agree to strongly disagree) on the statement was measured in the seven point Likert scale.

- I know this artist.
- I like this genre.
- I like this artist.

Participants who are regular concert goers or a concert-video viewers might have different expectations in a mashup than participants with less concert or concert video experience. However, it is difficult to accurately provide an average number of concert visits or views per year. Therefore we formulated the following questions along with the possible answers, given within brackets, one of which the participant had to choose.

- Concert visits in last two years: (0-1, 2-3, 4-5, 6-10, $>$ 10).
- Frequency of watching concert videos: (daily, weekly, monthly, yearly, rarely, never).

7.3 Procedure

The participants were briefly introduced to the research on mashup generation. They were informed that the mashups were generated by three different methods, but they were not aware of the algorithms or the *manual* made mashup to avoid any pre-conception. An example of concert recording from four cameras was shown in split screen as shown in Figure 7.2, which gave an impression on the type and quality of content used in the test.

The test began with a welcome window shown in Figure 7.3. Then the nine mashups were shown according to the balanced order each followed by the ques-

tionnaire and a free comment box. Participants could use the comment box to give any remark about the last mashup they saw. The questions on knowing an artist, liking an artist and liking a genre was asked for every concert after showing it for the first time. Figure 7.4 and 7.5 show the presentation of a control question and a statement, respectively, grabbed from the interface used in the test. Lastly, questions were asked about age, frequency of watching concert videos and number of concerts visited in past two years. Additionally, they were also given a final comment box to report or remark anything about the test or the mashups. The presentation of these questions and comment box is shown in Figure 7.6.

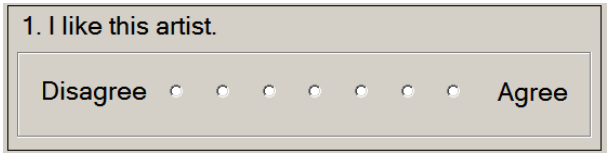


Figure 7.4. A control question grabbed from the interface used in the test.

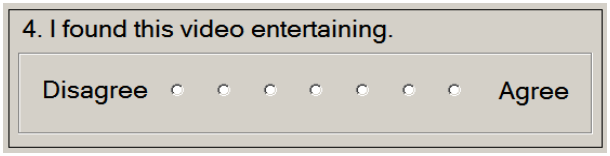


Figure 7.5. A statement from the questionnaire grabbed from the interface used in the test.

No control over the video player was given to the participant, to make sure that no mashups were skipped or watched in fast-forward speed. Participants were asked to continue watching during the play and answer the questionnaire right after

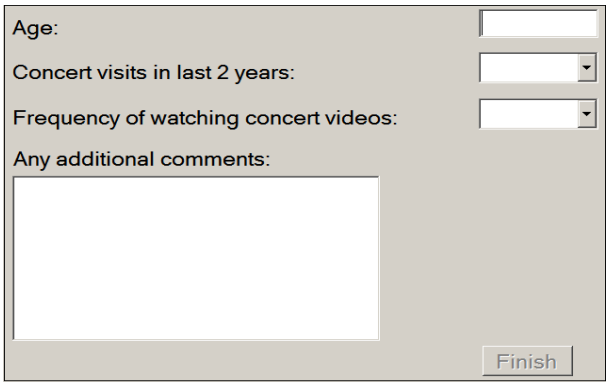


Figure 7.6. The final window used in the test, grabbed from the interface.

Figure 7.7. User study setup.

the video stopped. Short breaks, like visiting toilets, were allowed while answering the questions. The experimenter was available for clarifying questions or other technical problems throughout the test, however, the participants were not being observed.

The tests were conducted using personal computer and head phones in an laboratory during working hours. A photo of the test setup is shown in Figure 7.7.

7.4 Test results

7.4.1 Participants

Forty participants (17 female, 23 male) volunteered in the test. Most of them were student interns and some were employees at High Tech Campus, Eindhoven, The Netherlands. The average age of the participants was 27 (min: 22, max: 34). The average number of times they had been to a concert in the last two years was 3 (min: 1, max: 6). Among the participants, the frequency of watching concert videos was distributed as: 10% never, 45% rarely (less often than once a year), 32.5% monthly, 10% weekly, and 2.5% daily.

7.4.2 Analysis techniques

An average response value to a given statement for each algorithm is computed as a *mean* score across all the participants. Reliability of a mean score is estimated by *confidence interval*, such that if the test is repeated with other participants from the same target group, there is 95% probability that the mean score will remain within the interval. The confidence intervals are presented graphically as an error bar on the mean score.

To verify whether the differences between the mean scores are statistically significant, we conducted a two-way analysis of variance (ANOVA) with repeated measures. The *algorithm* and *content* variables were treated as within-subject independent variables and the response of the participants was treated as a dependent variable. The results are presented in terms of *F-statistic* and *p-value* such that if $p < 0.05$ there is 95% confidence that the means are significantly different. The ratio of the mean squares is given by F.

ANOVA indicates if the means are significantly different, but it does not distinguish which means are different. Therefore, an additional *Tukey post-hoc* test was performed on both independent variables. The results provide pairwise comparisons of the means and their confidence intervals.

7.4.3 S1: Atmosphere

The mean scores for statement **S1**: *I got a good impression of the concert atmosphere from the mashup video*, for the three algorithms and the three concerts are shown in Figure 7.8. In concerts **C1** and **C2**, the scores for *algorithm* seem to increase in the order: *naive, manual* and *first-fit*. However, in **C3** there seem to be little difference among the scores of the three algorithms. The effect of the algorithm is most obvious in **C2**, where *naive* scores considerably lower compared to the other two algorithms. The mean across all the content shows that *first-fit* scores higher than *manual* and *manual* scores higher than *naive*.

From the ANOVA analysis, a significant main effect was found for *algorithm* ($F = 3.119,\ p = 0.045$) and also for *content* ($F = 6.398,\ p = 0.002$). A Tukey test on *algorithm* shows that there is a significant difference between the scores of *naive* and *first-fit*, while the score of *manual* is not significantly different from both *naive* and *first-fit*. Similarly, a Tukey test on *content* shows that **C2** scores significantly higher than **C1** and **C3**.

All the recordings of concert **C3** contain similar views, mostly of the stage. The clips containing audiences are mostly dark due to the light setting in the concert hall, which was focused only towards the stage. Since the recordings provided limited content, the algorithms made little difference in conveying the mashup atmosphere. Since **C2** was held on a large open space during daylight hours, the recordings contain different views like stage, audience, and display boards. The mean scores show that *first-fit* and *manual* mashups are able to convey the concert atmosphere with high values, in contrary to a *naive* mashup. In **C1**, as in the case of **C3**, the audience views suffer from insufficient light. However, the recordings contain stage views from very different angles including a top view from the theater balcony. Since the concert was held in a small hall, it is relatively easy to capture the atmosphere in the mashups. Therefore, we see only a slight improvement in the mean scores of *first-fit* and *manual* algorithms over *naive*.

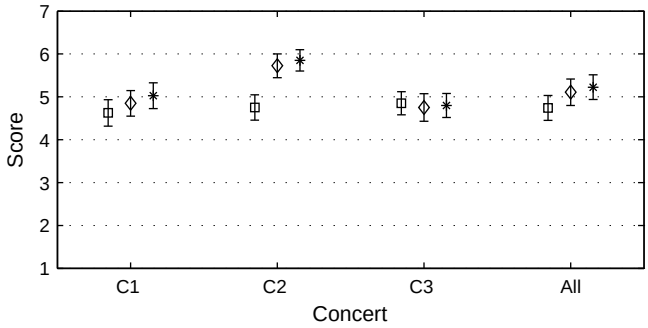


Figure 7.8. Mean scores for statement **S1**: *I got a good impression of the concert atmosphere from the mashup video*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

The score of *first-fit* is not statistically significant but slightly higher than *manual* in **C1** and **C2**. According to the comments written by some of the participants, they preferred *first-fit* because the focus of the *manual* mashups were mainly towards the artists, which limited their perception of the concert atmosphere.

7.4.4 S2: Overview

The mean scores for statement **S2**: *the different viewpoints shown in the video gave me a rich overview of the concert*, for the three algorithms and the three concerts are shown in Figure 7.9. In concert **C1**, the score of *first-fit* seems slightly higher than *naive* and slightly lower than *manual*. In concert **C2**, there seems to be a distinctly higher score for *first-fit* followed by *manual* and *naive*. In concert **C3**, there appears little difference among the mean scores of the algorithms. The mean across *content* shows that the score of *first-fit* and *manual* are about the same, which is higher than that of *naive*.

From the ANOVA analysis, a significant main effect was found for *algorithm* ($F = 3.085,\ p = 0.037$) and also for *content* ($F = 11.214,\ p < 0.001$). A Tukey test on *algorithm* shows that *naive* is significantly different from *manual* and *first-fit*. Similarly, a Tukey test on *content* shows that **C2** is significantly different than **C1** and **C3**.

The results are similar to that of the statement **S1**, as described in Section 7.4.3. The views captured by the cameras and lighting conditions in the venue plays an important role in the perception of the concert overview. Due to the limited variety in their content, the algorithms generate similar mashups for the concerts **C1** and **C3**, while the availability of rich content provides a clear distinction in concert **C2**.

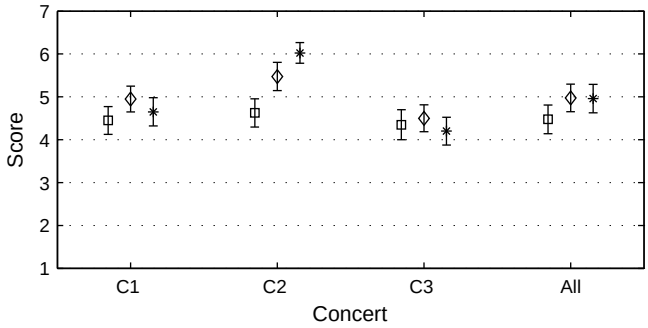


Figure 7.9. Mean scores for statement **S2**: *the different viewpoints shown in the video gave me a rich overview of the concert*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

7.4.5 S3 and S7: Camera stability

The mean scores for statement **S3**: *I got disturbed by the lack of camera stability*, for the three algorithms and the three concerts are shown in Figure 7.10. In concerts **C1** and **C2**, the scores for *algorithm* seem to increase in the order: *first-fit, manual* and *naive*. However, in **C3** the mean score for *first-fit* seems to be the highest and closely followed by *naive* and then *manual*. The effect of the algorithm seems most obvious in **C2**, where *naive* scores considerably higher than the other two algorithms. The mean across all the three concerts shows that the *naive* algorithm scores higher than *manual* and *first-fit*.

From the ANOVA analysis, a significant main effect was found for *algorithm* ($F = 4.356,\ p = 0.0126$) and also for *content* ($F = 25.005,\ p < 0.001$). A Tukey test on *algorithm* shows that *naive* is significantly different from *manual* and *first-fit*. Similarly, a Tukey test on *content* shows that the mean score of **C1** is significantly different from **C2** and **C3**.

The mean scores for statement **S7**: *the cameras in the video were stable*, for the three algorithms and the three concerts are shown in Figure 7.11. In all the concerts the *manual* algorithm scores higher than the other two algorithms. In concerts **C1** and **C3**, the two algorithms *first-fit* and *naive* seems to score almost the same. However, in concert **C2**, the *naive* score seems lower than the *first-fit* algorithm. The mean across all the three concerts shows increase in the order *naive*, *first-fit*, and *manual*.

From the ANOVA analysis, a significant main effect was found for *algorithm* ($F = 4.593,\ p = 0.010$) and for *content* ($F = 31.853,\ p < 0.001$). A Tukey test on

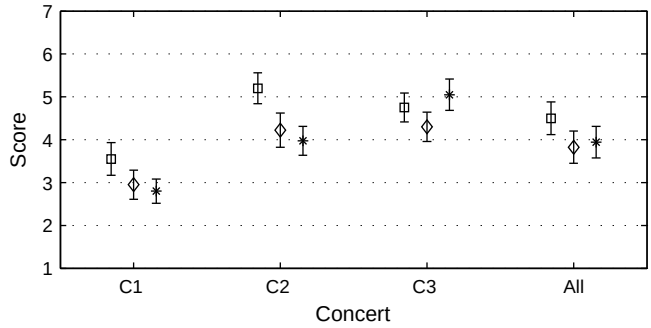


Figure 7.10. Mean scores for statement **S3**: *I got disturbed by the lack of camera stability*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

algorithm shows that *naive* is significantly different from the other two algorithms. Similarly, a Tukey test on *content* shows that **C1** is significantly different from **C2** and **C3**.

It is expected that the *naive* mashups are perceived as shaky and disturbing since they do not take into account the camera stability. In **C1** and **C2**, *manual* mashups are perceived about as stable as the *first-fit* mashups. However, in **C3**, the *first-fit* mashup is perceived as the most shaky and disturbing among the three algorithms. This could be due to the low visual quality of the camera recordings of concert **C3**, containing objects in fast motion (dancing) and dynamic lights, which caused errors in feature analysis, in particular in the shakiness detection algorithm. The scores of the statements **S3** and **S7** show that there is a direct relationship between the measure of camera stability (**S7**) and the measure of feeling disturbed (**S3**).

7.4.6 S4: Entertaining

The mean scores for statement **S4**: *I found this video entertaining*, for the three algorithms and the three concerts are shown in Figure 7.12. In **C1** and **C2**, the scores for *algorithm* seem to increase in the order: *naive, manual* and *first-fit*. However, in **C3** the mean scores of *manual* and *naive* seem to be about equal and higher than *first-fit*. The mean across all the three concerts shows that *manual* and *first-fit* score the same, while *naive* scores lower.

From the ANOVA analysis, no significant main effect was found for *algorithm* ($F = 2.209,\ p = 0.113$). However, a significant main effect was found for *content* ($F = 8.086,\ p < 0.001$). A Tukey test on *content* shows that there is no difference

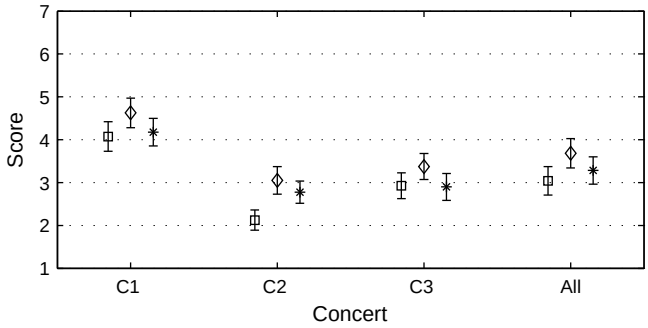


Figure 7.11. Mean scores for statement **S7**: *the cameras in the video were stable*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

between the mean scores of **C2** and **C3**, while the score of **C1** is significantly higher.

Although there was no effect found for *algorithm*, **C1** and **C2** show a consistent decrease of mean score in order: *first-fit*, *manual* and *naive*. According to the comments written by some of the participants, *manual* mashup are too predictable as every camera switch is accompanied by a music beat. The trend is absent in **C3** and *first-fit* scores even lower than the other two algorithms. As explained in Section 7.4.5, it might be due to the low visual quality of the camera recordings of the concert **C3**, which caused wrong estimation of the visual features.

7.4.7 S5: Variety

The mean scores for statement **S5**: *I think there was enough variety in the content*, for the three algorithms and the three concerts are shown in Figure 7.13. In concert **C1**, the *first-fit* score seems to be slightly higher than *naive* and slightly lower than *manual*. In concert **C2**, the score seems to increase in the order: *naive*, *manual* and *first-fit*. In concert **C3** *naive* seems to score slightly higher than *first-fit* and *manual*. The mean across all the three concerts also shows that the scores for *manual* and *first-fit* are very similar, which are slightly higher than *naive*.

From the ANOVA analysis, no significant main effect was found for *algorithm* ($F = 1.168,\ p = 0.312$). However, a significant main effect was found for *content* ($F = 7.266,\ p < 0.001$). A Tukey test on *content* shows that there is no difference between the mean scores of **C1** and **C3**, while the score of **C2** is significantly higher.

The results show that the perception of variety is largely dependent on the cam-

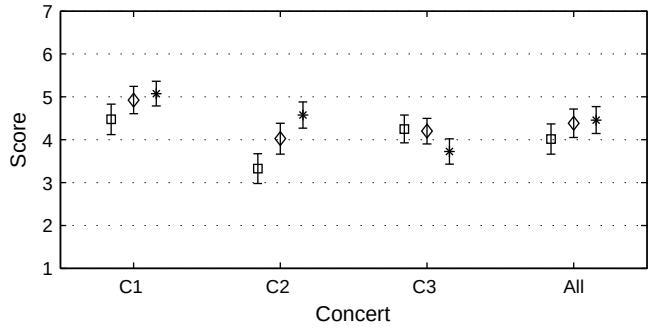


Figure 7.12. Mean scores for statement **S4**: *I found this video entertaining*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

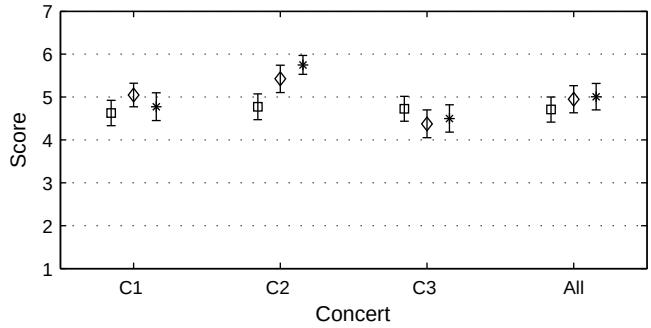


Figure 7.13. Mean scores for statement **S5**: *I think there was enough variety in the content*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

era recordings. For **C1** and **C3**, the algorithms cannot contribute to the perception of variety in a mashup since the available recordings have limited variety. The rich recording contents of **C2**, described in Section 7.4.3, provide an opportunity for the *manual* and *first-fit* algorithms to create a mashup with better diversity.

7.4.8 S6: Boring

The mean scores for statement **S6**: *I found the video boring to watch*, for the three algorithms and the three concerts are shown in Figure 7.14. In **C1**, *manual* and

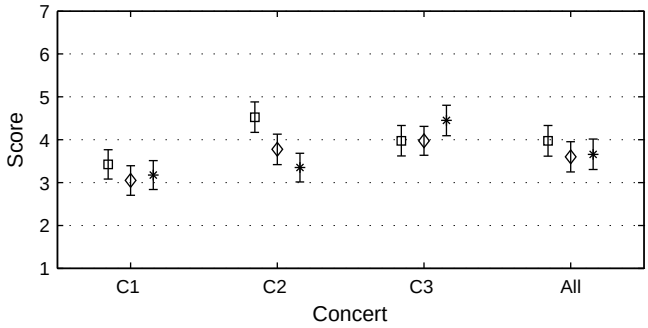


Figure 7.14. Mean scores for statement **S6**: *I found the video boring to watch*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

first-fit seems to score about similar, which looks slightly lower than *naive*. In **C2**, *naive* score seems to be higher than *manual* followed by *first-fit*. In **C3**, both *naive* and *manual* score seems the same, while *first-fit* looks slightly higher. The mean across all the three concerts shows that both *manual* and *first-fit* score about the same, which is lower than *naive*.

From the ANOVA analysis, no significant main effect was found for *algorithm* ($F = 1.397,\ p = 0.248$). However, a significant main effect was found for *content* ($F = 7.705,\ p = 0.002$). A Tukey test on *content* shows that **C1** is significantly different than the other two concerts. Although there was no effect found for *algorithm*, the *naive* mashup scores consistently high as boring in concerts **C1** and **C2**. However, in **C3** the *manual* algorithm scores the highest. As explained in Section 7.4.6, this might be due to the low visual quality of the camera recordings that caused poor feature analysis.

7.4.9 S8: Image quality

The mean scores for statement **S8**: *the image quality in the video was very bad*, for the three algorithms and the three concerts are shown in Figure 7.15. In all the concerts the *naive* score seems to be higher than the other two algorithms. In **C1** and **C2**, the two algorithms *first-fit* and *manual* scores seem to be about the same. In concert **C3**, both the *first-fit* and *manual* seem to be close to *naive*. The mean across all the three concerts shows *naive* scores higher than the other two algorithms, which score about the same.

From the ANOVA analysis, a significant main effect was found for *algorithm* ($F = 7.833,\ p < 0.001$) and for *content* ($F = 16.051,\ p < 0.001$). A Tukey test on

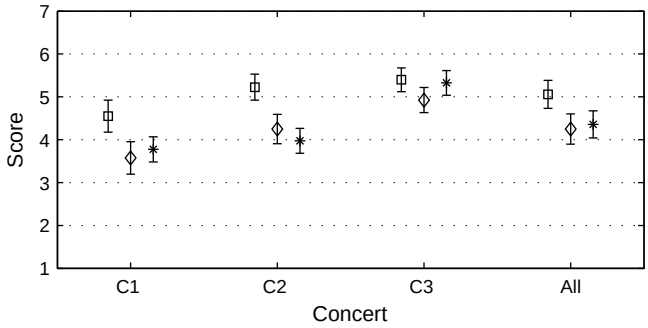


Figure 7.15. Mean scores for statement **S8**: *the image quality in the video was very bad*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

algorithm shows that *naive* is significantly different from the other two algorithms. Similarly, a Tukey test on *content* shows that **C3** is significantly different from **C1** and **C2**.

The results in **C1** and **C2** show that the mashups generated by the *naive* algorithm, which does not take into account image quality, are perceived as containing bad image quality. However, in concert **C3** the mean scores of all the algorithms are about the same because of the low quality of the concert recordings.

7.4.10 S9: Overall good

The mean scores for statement **S9**: *overall, I think the video was good*, for the three algorithms and the three concerts are shown in Figure 7.16. In **C1** the mean score seems to increase in the order: *naive*, *manual* and *first-fit*. A similar trend is seen in **C2**, however, the mean score of the *naive* algorithm appears very low. In **C3**, the *naive* score looks slightly higher than *first-fit* and slightly lower than *manual*. The mean across all the three concerts shows that *manual* and *first-fit* score about the same, which is higher than that of *naive*.

From the ANOVA analysis, no significant main effect was found for *algorithm* ($F = 2.271,\ p = 0.071$). However, a significant main effect was found for *content* ($F = 8.993,\ p < 0.001$). A Tukey test on *content* shows that concert **C1** is significantly different than **C3**.

The Tukey test results are the same as in the case of statement **S4**, described in Section 7.4.6, and **S6**, described in Section 7.4.8. The perception of overall goodness in a mashup is mainly dependent on *content* rather than on *algorithm*. However, in concerts **C1** and **C2** there is a consistent trend that the score increases

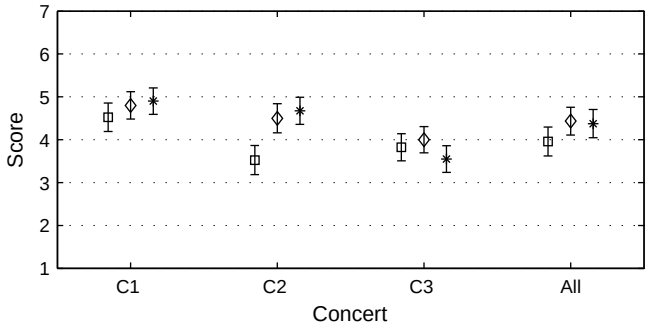


Figure 7.16. Mean scores for statement **S9**: *overall, I think the video was good*. The algorithms are represented as: naive (□), manual (◇), and first-fit (∗). The horizontal axis represents the mashups corresponding to the three concerts. "All" represents the mean across all the concerts. Error bars show confidence intervals of 95% of the mean value.

in the order: *naive, manual* and *first-fit*. In concert **C3**, the *first-fit* scores the lowest maybe due to the poor visual quality of the camera recordings that caused wrong visual feature analysis.

7.4.11 Effects of control variables

We included some control questions in the test, as described in Section 7.2.7, to check the influence of factors such as age, gender, liking the artist. The analysis involved one-way ANOVA in which the users' response to the statements, described in Section 7.2.6, is divided into groups according to the users' response to a control question. The ANOVA results indicate how significantly the group means are different (*p-value*) and the ratio of the mean squares, (*F-statistic*). An additional *Tukey post-hoc* test was performed for pairwise comparison of the group means and their confidence intervals. The following sections present the results for different control variables.

Age

To simplify the analysis, we mapped the age of the participants into three groups: 25 and younger (13 participants), between 26 to 30 (18 participants), 31 and older (9 participants). A significant main effect of age was found on statements **S2**: *the different viewpoints shown in the video gave me a rich overview of the concert*, ($F = 3.31,\ p = 0.037$) and **S5**: *I think there was enough variety in the content*, ($F = 5.180,\ p = 0.006$). Tukey tests on both statements show that the perception of variety and overview in a mashup is significantly lower in the age group 25 and younger than that of the age group 31 and older. The result can be explained if

we observe the trend in professionally produced music videos where the density of shot-cuts and fast transitions has increased tremendously in recent years [Reeves & Nass, 1996]. The results show higher expectations of the younger generation for variety and overview in a mashup.

Knowing an artist

No significant main effect was found for knowing the artist for any of the statements. So the participants were not biased in the test because they knew an artist.

Liking an artist or musical genre

There was a strong similarity in the user response to the control statements, *I like this genre* and *I like this artist*, which was measured in the scale of one to seven, as described in 7.2.7. The estimation of correlation between the two statements using standard Cronbach's α, results in $\alpha = 0.946$.

In the ANOVA analysis, both control statements show a significant main effect for statements **S4**: *I found this video entertaining*, **S6**:*I found the video boring to watch*, and **S9**: *Overall, I think the video was good* with results $F > 6$ and $p < 0.001$. The participants who did not like the musical genre in the mashup perceived it as significantly boring to watch than the people who were neutral and who liked the genre. The more the participants like an artist or a genre, the more the mashups were perceived as entertaining, overall good and not boring. Therefore, the pleasantness of a mashup is influenced by liking an artist or musical genre.

Number of concert visits

According to the user response on *the number of concert visits in the last two years*, participants were mapped into three groups: two or less visits (17 participants), three to five visits (15 participants), six or more (8 participants). No significant main effect was found on the statements due to the number of concert visits. However, a positive trend was found for statements **S2**: *The different viewpoints shown in the video gave me a rich overview of the concert*, **S9**: *Overall, I think the video was good*, and a negative trend was found for statement **S3**: *I got disturbed by the lack of camera stability*. It is an interesting observation that people who like going to concerts more often are less disturbed by the camera stability and appreciate a mashup more than people who visit concerts less often.

Frequency of concert-video watching

According to the user response on *frequency of watching concert videos*, participants were mapped into three groups: daily or weekly (5 participants), monthly or yearly (13 participants), rarely or never (22 participants). A significant effect was found for statement **S8**: *The image quality in the video was very bad*,

($F = 5.82,\ p < 0.001$). Tukey tests showed that the first group of participants perceives the mashup image quality significantly lower than the third group of participants. The result shows that people who are concert video fans are more critical towards the image quality of a mashup.

7.4.12 Comments

The comment-boxes were provided after every mashup and at the end of the test. The participants filled-in total 160 boxes (on average 4 boxes per participants), where each box contained comments on a number of topics. We clustered the comments into six categories according to their topics and they appeared at least twice in each of the algorithms: *naive, manual, first-fit*. Table 7.1 presents the six categories and the number of positive and negative comments on each category per algorithm. The comments that are not relevant for the comparison of the algorithms or appeared only once are not shown in the table.

The highest number of comments were made on the shot-cuts which included shot duration and timing of the cut. Since the naive algorithm did not consider the timing of the cut, it received the highest number of negative comments, such as 'shaky scenes too long and good scenes too short', 'I'm distracted with the abrupt cuts'. Other interesting comments were that the manual mashups had very regular cuts accompanied with the music beats, which made them predictable and boring. The first-fit mashups received most of the positive comments in this category, however, it was also mentioned in some of the comments that there were obvious signs of an automatic mashup generation such as 'there was a cut right in the middle of the most interesting moment'.

The comments in the content category include user opinions on the artist, genre and the concert itself. The concert **C3** got most of the comments as the participants had more strong opinions on the artist. Typical remarks were: 'I can't watch this video I hate her' and 'she is the best'. The brightness category includes comments referring to black shots and overall brightness level in the mashup shots. As expected, the naive algorithm got most of the negative comments, such as 'too many dark scenes'. Similarly, in the other comment categories: stability, quality, variety, the naive algorithm got most of the negative comments. In general, the comments reflect a similar opinion towards the *algorithms* as found in the user response to the questionnaire. The *naive* mashup is perceived as the lowest in quality, while *first-fit* and *manual* mashups are similar to each other.

Another very common comment, almost from every participant, was on the inconsistent audio quality of the mashup. We did not present the topic in Table 7.1 because it was not a differentiating factor among the *algorithms*. All the mashups contained audio along with the corresponding video as recorded by the cameras, which were different in quality and volume due to different ambient noises and

Table 7.1. Comment categories and number of positive and negative comments per algorithm

	Naive		**Manual**		**First-fit**	
	+	-	+	-	+	-
Shot-cuts	1	12	4	3	7	5
Content	0	8	2	5	2	6
Brightness	0	7	0	2	2	0
Stability	1	7	1	1	1	3
Quality	1	5	4	5	3	3
Variety	3	2	6	2	5	4

microphone settings of the individual cameras. Choosing audio stream from one of the recordings was not an option as one recording might not always be available throughout a mashup and even if it was available it might not contain high quality audio. Since we did not have access to the original high-quality audio recording, we could not analyze and improve the audio quality of the mashups. The test participants were informed about the state of the audio in the mashups during introduction of the test. However, many commented that 'I was disturbed by the low-quality audio' and 'I would have enjoyed the mashups a lot more if the audio would have been better'.

7.5 Discussion

The test results provide a measure of quality perception in terms of the criteria *diversity, visual quality* and *pleasantness*, described in Section 7.1. The following sections discuss the test results on these criteria.

7.5.1 Diversity

The diversity criteria, described in Section 7.1.1, were tested in terms of **S1**: *I got a good impression of the concert atmosphere from the mashup video*, **S2**: *I got disturbed by the lack of camera stability* and **S5**: *I think there was enough variety in the content*. A high score of these statements corresponds to better perception of diversity. The mean scores across the *content* for the statements are shown in Figure 7.17. In all the three statements *manual* and *first-fit* score about the same, which is always higher than *naive*. An estimation of correlation among the scores of the statements using standard Cronbach's α results in $\alpha = 0.881$. The high value of α suggests that all three statements agree on the same opinion on the measurement of diversity.

According to the results presented in Sections 7.4.3, 7.4.4, 7.4.7 and Figure 7.17, the mashups generated by the *naive* algorithm are perceived as the least diverse, while the ones generated by *manual* and *first-fit* are perceived as comparable.

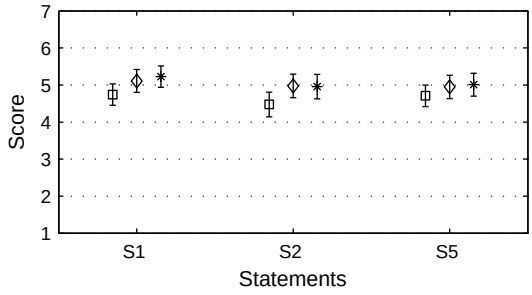


Figure 7.17. Mean scores across the *content* for naive (□), manual (◇) and first-fit (∗) algorithms on the statements measuring diversity. The horizonal axis represents the three statements. Error bars show confidence intervals of 95% of the mean value.

The perception of diversity in a mashup is found to be influenced by the age factor as described in Section 7.4.11. The younger participants (25 and younger) found that the mashups contained less overview and less variety in content than the older participants (31 and older).

7.5.2 Visual quality

The visual quality criteria, described in Section 7.1.2, was tested by statements **S3**: *I got disturbed by the lack of camera stability*, **S7**: *The cameras in the video were stable* and **S8**: *The image quality in the video was very bad*. Statement **S7** provides a measure of camera stability, while statement **S3** provides a measure of feeling disturbed due camera instability. Statement **S8** provides perceptual measure of the image quality. Since the statements **S3** and **S8** have negative connotations, the lower scores indicate higher visual quality. For the clarity in illustration, the scores of these statements were subtracted from 8, such that in all the three statements higher scores correspond to a better visual quality. The mean scores across *content* are shown in Figure 7.18.

In all the three statements, *manual* scores the highest, closely followed by *first-fit*, while *naive* scores the lowest. The correlation measurement among the scores of the statements using standard Cronbach's α results in $\alpha = 0.811$. The high value of α suggests that all three statements agree on the same opinion about visual quality. According to the results presented in Sections 7.4.5, 7.4.9 and Figure 7.18 it is verified that *naive* algorithm generates a mashup with lower visual quality than the other two algorithms. In the case of *manual* and *first-fit* there is not a big difference in mean scores, however, *manual* mashups are consistently perceived as having higher visual quality.

The evaluation of visual quality is found to be influenced by the frequency

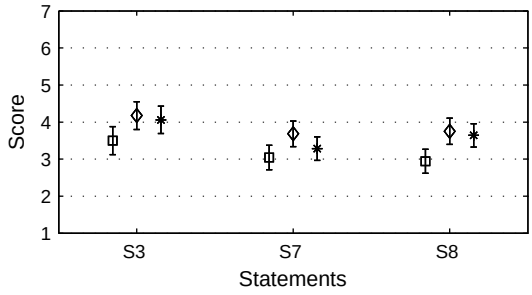


Figure 7.18. Mean scores across the *content* for naive (□), manual (◇) and first-fit (∗) algorithms on the statements measuring visual quality. The horizonal axis represents the three statements. Error bars show confidence intervals of 95% of the mean value.

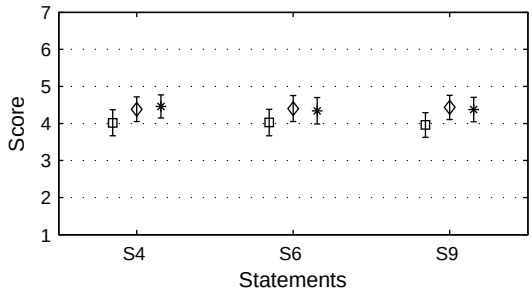


Figure 7.19. Mean scores across the *content* for naive (□), manual (◇) and first-fit (∗) algorithms on the statements measuring pleasantness. The horizonal axis represents the three statements. Error bars show confidence intervals of 95% of the mean value.

of watching concert videos. As described in Section 7.4.11, people who watch concert videos on weekly or daily basis were more critical towards mashup image quality (**S8**) than people who rarely watch concert video.

7.5.3 Pleasantness

The pleasantness criteria, described in Section 7.1.3, is tested in terms of **S4**: *I found this video entertaining*, **S6**: *I found the video boring to watch*, and **S9**: *Overall, I think the video was good.* A pleasant mashup is indicated by higher scores of statements **S4**, **S9** and lower score of statement **S6**. For the clarity in illustration, the scores of **S6** is subtracted from 8, such that in all the three statements higher scores correspond to a more pleasant mashup. The mean scores across *content* are shown in Figure 7.19. In all the three statements, the *naive* algorithm scores the

lowest. The *manual* and *first-fit* score about the same. The correlation measurement among the scores of the statements using standard Cronbach's α results in $\alpha = 0.893$. The high value of α suggests that all the three statements agree on the same opinion on the measure of pleasantness. According to the results presented in Sections 7.4.6, 7.4.8 and 7.4.10. It is verified that the *naive* algorithm generates mashups that are less pleasant than the ones generated by *manual* and *first-fit*.

The evaluation of pleasantness is found to be dependent on liking an artist (or genre) present in a mashup. As described in Section 7.4.11, people who like an artist (or genre) found the mashups containing the artist (or genre) more pleasant to watch than people who do not like them.

7.6 Conclusions

From the user study we can conclude that the perceived quality, in terms of diversity, visual quality and pleasantness, of a mashup generated by the naive algorithm is consistently lower than that of the ones generated by the manual and first-fit algorithms. Therefore the hypothesis **H1** and **H2**, described in Section 7.1 are confirmed.

Between the first-fit and manual mashups, the first-fit scores slightly higher in diversity but slightly lower in visual quality. The pleasantness scores of both algorithms are very similar. Therefore, hypothesis **H3** is confirmed.

The perception of a mashup quality is highly dependent on the content. The camera recordings with multiple view angles, variety in content and good visual quality allow the manual and first-fit algorithms to generate mashups that are perceived as significantly higher in quality than that of the ones generated by the naive algorithm.

The feature analysis techniques, described in Chapter 5, used in measuring the degree of fulfillment of the mashup requirements provided fairly satisfactory results. On average, the perceived image quality and diversity of the first-fit mashups were close to the manual mashups and higher than the naive mashups. The cut-points in the first-fit mashups were preferred even more than that of the manual mashups, however, this may be caused by the manually annotated audio cut-points. So we cannot comment on the performance of feature analysis on the cut-point suitability. The feature analysis, however, failed in case of concert C3 where the recordings were of bad visual-quality.

The effects of control variables such as knowing an artist and liking an artist on the perception of the mashup quality show some interesting findings. The mashup evaluation was found to be not influenced by knowing an artist present in the mashup. However, the perception of pleasantness was biased by liking an artist or a genre. A significant difference was found between people who watch concert

videos very often (daily or weekly) and people who watch concert videos rarely or never. The former group was found to be more critical towards the image quality of the mashups than the latter group. Another significant difference was found on the demands for content diversity between the younger group of participants (25 and younger) and the older group (31 and older). The results show higher expectation of the younger group for more diversity in content than that of a slightly higher age group. The results match with the recent trend in video-production of increasing shot-cut density and fast transitions [Reeves & Nass, 1996].

8

Conclusions

In this thesis, we presented an automated mashup generation system for multiple-camera recordings captured during musical concerts by non-professionals. We described the research methodologies and techniques followed in designing, developing and evaluating the system. In this final chapter, we present the main contributions of this thesis and suggest some future work.

8.1 Conclusions

We started our research by selecting a target application area and eliciting requirements for generating mashups. We interviewed experts on video-editing and conducted focus group meetings involving multimedia researchers, industrial design students and amateur video-editors. Based on the opinions of the focus groups and the rising popularity of concert videos in web-sites like YouTube, we decided to focus on concert recordings captured by non-professionals. We also compiled a list of 12 requirements for concert video mashups, such as image quality, diversity, special effects.

We proposed a system consisting of three steps: pre-processing, mashup-composition and post-processing, where each step addresses certain requirements that help addressing other requirements in successive steps. This thesis focuses on the first two steps and post-processing is left as future work. We introduced

a formal model for an automatic mashup generation system by translating the requirements into mathematical terms.

In the pre-processing step, recordings from multiple cameras are aligned in a common time-line to address the requirement *synchronization*. We proposed an automated synchronization approach based on detecting and matching audio and video features extracted from the recorded content. We developed three realizations of the approach using different features: still-camera flashes in video, audio-fingerprints and audio-onsets. In each realization, features of a recording are compared with the corresponding features of another recording using techniques such as cross-correlation, longest subsequence matching and Hamming distance. We evaluated the performance of the different realizations in a common test-set. The audio-fingerprint and audio-onset based realizations were found to be the most suitable for synchronizing multiple-camera recordings from musical concerts.

In the mashup-composition step, we proposed an optimization based approach to select the mashup clips. We modeled the requirements into *score functions* and *constraints*. The score functions provide a numerical representation of the degree of fulfillment of the requirements: *image quality*, *diversity* and *cut-point suitability*. Different audio-visual feature extraction and analysis techniques were employed to compute the scores of the recordings, where a higher score indicated better fulfillment of a requirement. The *constraints* represent the requirements: *suitable-clip duration* and *completeness*, which must be fulfilled in a mashup. A global *objective function* was defined to combine all the score functions and provide a measure of the mashup quality. We developed an algorithm, *first-fit*, based on a greedy optimization technique, which composes a mashup by satisfying the constraints and maximizing the objective function.

We performed an objective evaluation of the quality of the mashups, generated by the first-fit algorithm with respect to the ones generated by two other methods: *naive* and *manual*. In the naive method, a mashup was generated by satisfying only the requirements given as constraints. The comparison between naive and first-fit mashups on ten multiple-camera recordings containing 3–9 recordings shows that the quality score of the first-fit mashups are at least ten times higher than the naive mashups. In the manual method, mashups were created by a professional video-editor. The comparison among naive, manual and first-fit mashups on three multiple-camera recordings containing 4–5 recordings shows that the quality score of the first-fit mashups are the highest and that of the naive mashups are the lowest, while the manual mashups score slightly higher than the naive mashups.

We performed a subjective evaluation to measure the end-user satisfaction of the first-fit, naive and manual mashups. We used the test-set containing three multiple-camera recordings, the same as used in the objective evaluation of the three mashups. The subjects were asked to provide their ratings on the mashup

quality in terms of three aspects *diversity*, *visual quality* and *pleasantness* after watching each of the mashups. The results showed that the naive mashups score consistently and significantly lower than the other mashups in all the aspects. The first-fit mashups scored slightly higher in diversity but slightly lower in visual quality. The pleasantness scores of the mashups generated by the first-fit and manual methods were very similar. Therefore, we conclude that the perceived quality of a mashup generated by the naive method is lower than the first-fit and manual while the perceived quality of mashups generated by the first-fit and manual methods are similar.

In Section 1.2 of Introduction, we presented four research questions. To conclude the thesis, here we revisit the questions and present our answers:

1. *What are the requirements for generating a mashup?*
   We conducted interviews with experts and focus group meetings involving multimedia researchers, design students and amateur video-editors to understand the usage of multiple-camera recordings. Based on the discussions, we selected musical concerts as the target application of our research. We also compiled a list of 12 requirements for generating a mashup from non-professional multiple-camera concert recordings. The requirements contain desired characteristics in a mashup and user control during the mashup generation process. We addressed a set of these requirements in our automatically generated mashup. In both objective and subjective evaluation, these mashups scored higher than the mashups created by a naive method, which addresses only two of the requirements. This shows that the compiled requirements are applicable in generating a higher quality mashup. In comparison to the mashups made by a professional editor, our automated mashups scored higher in objective evaluation, which indicates the set of requirements addressed by the professional are different from our set of requirements. We analyzed the manual mashups to understand how different requirements were weighed by the expert, but the results show no consistency on the weights of the requirements in different mashups. Since we had manual mashups from only three sets of multiple-camera recordings made by one expert, a larger number of test set would be required to verify the differences in requirements addressed. In the subjective evaluation, both the manual and our automated mashups were perceived as similar. This shows that the requirements represent end-user needs. However, a thorough analysis with a mashup that satisfies the complete set of requirements would be needed to further answer this question.
2. *How can the requirements be addressed by a mashup generating system?*
   To address the requirement for synchronizing multiple-camera recordings,

we proposed synchronization techniques based on audio-visual features: still camera flashes in video, audio-fingerprints and audio-onsets. The proposed techniques are applicable independent of the number of recordings and their frame rates. The flash based technique requires at least 2 common flashes in the recordings and provides synchronization accuracy of $\pm$ 40 msec, assuming videos are captured at the rate of 25 frames per sec. The audio-fingerprint and audio-onset based technique requires 3 sec of common audio in the recordings and provide synchronization accuracy of $\pm$ 11.6 msec. In our test-set the audio-onset performed the best by synchronizing 29 out of 30 test recordings. We conclude that the audio-onset and audio-fingerprint based method are suitable to apply in synchronizing concert video recordings. To address other requirements for generating a mashup, we modeled the mashup generation problem as an optimization problem. We defined an objective function, which maximizes the degrees of fulfillment the requirements while satisfying the constraints. The model allows to address multiple requirements and provides a mashup quality score corresponding to the degree of fulfillment of the requirements.

3. *How can a mashup that satisfies the requirements be generated?*
   We proposed an algorithm, *first-fit*, based on a greedy optimization approach to select the clips that best satisfy the requirements. According to the formal model for automatically generating mashups, the algorithm maximizes the fulfillment of some requirements and satisfies the constraints. The advantage of the approach is that it efficiently addresses multiple mashup requirements, however, it does not guarantee a globally optimal solution. The mashups generated by the first-fit algorithm were compared with the mashups created by a naive algorithm, which satisfies only the constraints and manually made by a professional video-editor. In our objective evaluation on 10 sets of multiple-camera recordings containing 3–9 recordings, the first-fit mashups scored at least 10 times higher than the naive mashups. This shows that the first-fit algorithm addresses the mashup requirements better than the naive method. The manual mashups score slightly higher than the naive mashups but lower than the first-fit mashups. Since we had manual mashups only on three sets of multiple-camera recording made by one individual, and there are many factors involved in creating both mashups, we cannot conclude if the first-fit or manual mashups better address the requirements.
4. *How can the generated mashups be evaluated?*
   To measure the end-user satisfaction offered by the first-fit, manual and naive mashups, we conducted a subjective evaluation. The results show that the naive mashups score consistently and significantly lower than the other

mashups in all the aspects. The first-fit mashups score slightly higher in diversity but slightly lower in visual quality than the manual mashups. The pleasantness scores of both algorithms are very similar. Therefore, we conclude that the perceived quality of a mashup generated by the naive method is lower than the first-fit and manual while the perceived quality of the mashups generated by the first-fit and manual methods are similar. Additionally, perception of a mashup quality is found to be highly dependent on the content. The recordings that contain multiple view angles, variety in content and good visual quality allow manual and first-fit methods to generate mashups, which are perceived as significantly higher quality than that of the ones generated by the naive method.

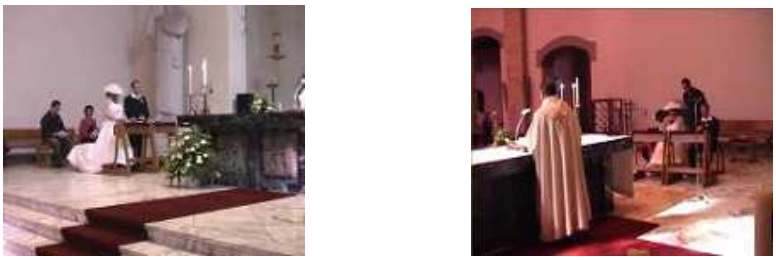
8.2 Future work

The thesis focuses on the generation of an automatic mashup. Therefore, the requirements for a mashup generation system compiled in Chapter 2 have not been completely addressed. The remaining requirements can be applied in the post-processing step, described in Section 3.1. Below we describe some research ideas that can be helpful in addressing these requirements and extending the mashup generation to other application areas.

8.2.1 Color balance

Recordings from different cameras may look different in color, even if they contain the same object at the same time, due to camera quality and settings. Figure 8.1 shows example frames from two wedding recordings. Although both frames contain the same couple in the same location around the same time, the overall color effect is very different. Available editing tools such as Adobe Premiere Pro and Final Cut Pro offer tools to correct the color discrepancies between clips by manually selecting a color of a clip as a reference and adjusting the corresponding color of the another clip. In order to achieve a desirable match, a user has to adjust several parameters such as highlight, mid-tones, shadows, brightness and contrast in these editing tools. The process is very time consuming and in many cases it is impossible to find the right color balance. Therefore, a research challenge is to provide an easy solution to normalize the color of the clips contained in a mashup.

When there are multiple-camera videos, all with non-professional sources, a challenge is to automatically find a reference video that can be used to normalize the other recordings. A solution could be to use a professional recording, which is most likely to contain a high quality color, or use a color model that is perceptually most pleasing, or allow users to provide a reference. In [Shrestha, Sekulovski, Weda, Barbieri & Clout, 2008], we propose an algorithm to normalize colors be-



(a) (b)

Figure 8.1. Two images captured at the same event from different cameras. However, the same objects in the two images show different colors.

tween two images using dominant colors at reference points given by users. The algorithm could be extended to balance the colors of the clips from different recordings in a mashup.

8.2.2 User interface

Since user-generated videos are personal recordings, users would like to be in charge of the decisions while editing them. We can assume that automatic mashups cannot satisfy the requirements of all users. However, as we described in Chapter 1, current tools for editing multiple-camera recordings are extremely complex and technical, which makes their use very time consuming and difficult. The research challenge is to design an intuitive user interface for editing multiple-camera recordings, which provides balance between user control and automation.

[Campanella, 2009] proposes a semi-automatic editing interface for home-videos recorded by a single camera, in which an automatic summary is generated based on video features and users can apply their editing decisions on the automatic summary. A similar approach can be applied in the context of multiple-camera videos, where a user can personalize the automatic summary rather than synchronizing the videos and edit on frame level. Figure 8.2 shows a concept of a multiple camera editing interface, designed by a multimedia design student, Ilona Hein, as a part of her study assignment. The design allows visualization of videos, generate summaries and apply editing functions intuitively in a single window. The users can import multiple-camera recording and assign a preference score to each of the recordings using the number of stars. The recordings are automatically synchronized and represented in a time-line. The users can specify an audio stream from one of the recordings or import another stream. In order to generate a summary, the user provides a preferred duration. The summary is represented by a collection of key frames from the recordings in a three dimensional elliptical plane. A player is located at the front of the ellipse, where the key frames can be dragged individually

Figure 8.2. Concept design of a web based multiple camera interface prepared by Ilona Hein. The synchronized videos and audio stream are presented in a time-line at the left side. The mashup summary from different recordings are presented in an elliptical array of frames, which represent clips from the recordings. A player is located at the front of the ellipse. The buttons for simple editing functions such as adding texts on the video, special effects are displayed at the lower right side.

or play the summary in sequence. On clicking a key frame on the ellipse, the corresponding synchronized segments in all the multiple-camera recordings become highlighted and can be swapped by the user. The interface also provides simple editing functions like adding texts on the video, transitions and effects.

8.2.3 Other applications

Since the use of multiple cameras is common in many types of events such as sports, surveillance and lectures, the idea of generating mashups can be applied in many other applications. Some of the application scenarios for home videos are presented in Chapter 2. In events such as sports, the events are captured not only by amateurs but also by professionals. The stationary cameras at lecture-halls, meeting-rooms and super-markets also produce multiple-camera recordings. The recordings from amateurs, professionals and stationary cameras are most likely to have different characteristics, however they capture the same event. Similarly, the purpose of a mashup of the recordings from a meeting should be very different than that of a sport. The research challenge lies in developing a method for generating mashups that is general enough to be applicable to different applications by employing different sources of recordings and easily accommodate different requirements.

Bibliography

BARBIERI, M. [2007], *Automatic summarization of narrative video*, Ph. D. Thesis, Eindhoven University.

BELLO, J. P., L. DAUDET, S. ABDALLAH, ET AL. [2005], A tutorial on onset detection in music signals, *IEEE Transactions on Speech and Audio Processing, Volume 13, Issue 5, Part 2*.

BREEBAART, J., AND M. MCKINNEY [2003], Features for audio and music classification, *Proceedings of Int. Symposium on Music Information Retrieval*.

BT.1359-1, RECOMMENDATION ITU-R [1998], Relative timing of sound and vision for broadcasting.

CAMPANELLA, M. [2009], *Balancing automation and user control in a home video editing system*, Ph. D. Thesis, Eindhoven University.

CAMPANELLA, M., H. WEDA, AND M. BARBIERI [2007], Edit while watching: home video editing made easy, *Proceedings of the IS&T/SPIE Conference on Multimedia Content Access: Algorithms and Systems, Volume 6506*, 65060L–1 – 65060L–10.

CARLETTA, J. [2007], Unleashing the killer corpus: experiences in creating the multi-everything AMI meeting corpus, *Language Resources and Evaluation Journal, Volume 41(2)*.

CASPI, Y., AND M. IRANI [2002], Aligning non-overlapping sequences., *International Journal of Computer Vision, Volume 48, Number 1*, 39–51.

CASPI, Y., D. SIMAKOV, AND M. IRANI [2004], Feature based sequence-to-sequence matching, *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*.

CORMEN, H., C. H. LEISERSON, R. L. RIVEST, AND C. STEIN [2001], *Introduction To Algorithms*, MIT Press.

CREMER, M., AND R. COOK [2009], Machine-assisted editing of user generated content, *Proceedings of SPIE-IS&T Electronic Imaging, SPIE Vol 7254*.

DYCKHOFF, H., AND U. FINKE [1992], *Cutting and Packing in Production and Distribution: A Typology and Bibliography*, Springer.

ELMROTH, E., AND J. TORDSSON [2008], Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions, *Future Generation Computer System, Volume 24, Number 6*.

ENGSTRÖM, A., M. ESBJÖRNSSON, AND O. JUHLIN [2008], Mobile collaborative live video mixing, *MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, ACM, 157–166.

GAO, W., C. MERMER, AND Y. KIM [2002], A de-blocking algorithm and a blockiness metric for highly compressed images, *IEEE Transactions on Circuits and Systems for Video Technology, Volume 12*, 1150 – 1159.

GREENE, K. [2009], Photosynth for video and other techfest treats. `http://www.technologyreview.com/blog/editors/23023/?a=f`.

GUIMARAES, S. J. F., M. COUPRIE, A. A. ARAÚJO, AND N. J. LEITE [2001], A method for cut detection based on visual rhythm, *Proceedings of the 14th Brazilian Symposium on Computer Graphics and Image Processing*, 297.

HAITSMA, J., AND T. KALKER [2002], A highly robust audio fingerprinting system, *Proceedings of the International Symposium on Music Information Retrieval*.

HATA, T., T. HIROSE, AND K. TANAKA [2000], Skimming multiple perspective video using tempo-spatial importance measures, *VDB 5: Proceedings of the Fifth Working Conference on Visual Database Systems*, The Netherlands, Kluwer, B.V., 219–238.

HIRANO, M., K. SHINTANI, H. NAKAO, S. OHTA, S. KANEDA, AND H. HAGA [2007], Children observation support system using multiple cameras COSS-MC, *WBED'07: Proceedings of the sixth conference on IASTED International Conference Web-Based Education*, Anaheim, CA, USA, ACTA Press, 627–630.

JTC1/SC29/WG11-N9163, ISO/IEC [2007], Requirements on multi-view video coding v.8 , MPEG document.

KENNEDY, L., AND M. NAAMAN [2009], Less talk, more rock: automated organization of community-contributed collections of concert videos, *WWW '09: Proceedings of the 18th international conference on World wide web*, ACM, 311–320.

KLAPURI, A. [2003], Musical meter estimation and music transcription, *Cambridge Music Processing Colloquium*.

LAMPI, F., S. KOPF, M. BENZ, AND W. EFFELSBERG [2008], A virtual camera team for lecture recording, *IEEE Multimedia, Volume 15, Number 3*, IEEE Computer Society, 58–61.

LEI, C., AND Y. H. YANG [2005], Tri-focal tensor based multiple video synchronization with sub-frame optimization, *IEEE Trans. on Image Processing*.

LEVEAU, P., AND L. DAUDET [2004], Methodology and tools for the evaluation of automatic onset detection algorithms in music, *Peoceedings of the International Conference on Music Information Retrieval (ISMIR)*, 72–75.

LI, X. [2002], Blind measurement of blocking artifacts in images, *International Conference on Image Processing, Volume 1*, I–449 – I–452.

LIENHART, R. [1999], Abstracting home video automatically, *Proceedings of the 7th ACM International Conference on Multimedia (Part 2)*, 37–40.

MARTELLO, S., AND P. TOTH [1990], *Knapsack problems: algorithms and computer implementations*, J. Wiley and Sons.

MICHEL, M., AND V. STANFORD [2006], Synchronizing multimodal data streams acquired using commodity hardware, *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, 3–8.

MPEG [2002], ISO/IEC 15938-8 multimedia content description interface-part 8: extraction and use of MPEG-7 descriptors.

MPEG [2008], ISO/IEC JTC 1/SC 29/WG 11 N9965: MPEG multiplies views of video.

NACI, U. S., AND A. HANJALIC [2007], Intelligent browsing of concert videos, *Proceedings of the 15th international conference on Multimedia*, ACM, 150–151.

ONG, E., AND ET. AL. [2003], A no-reference quality metric for measuring image blur, *Proceedings. Seventh International Symposium on Signal Processing and Its Applications, Volume 1*, 469 – 472.

O'SHAUGHNESSY, D., P. KABAL, D. BERNARDI, ET AL. [1990], Applying speech enhancement to audio surveillance, *Journal of Forensic Sciences, Volume 35, Number 5*.

PETERS, M. A., AND P. M. F. FONSECA [2007], Photo management applications, *Nat.Lab. Report TN-2007/00698*, Philips Research, Eindhoven, The Netherlands.

REEVES, B., AND C. NASS [1996], *The media equation: How people treat computers, television and new media like real people and places*, CSLI Publications, Cambridge University Press,.

RUSSELL, S., AND P. NORVIG [2002], *Artificial Intelligence: A Modern Approach*, Prentice Hall.

SAWHNEY, H. S., S. HSU, AND R KUMAR [1998], Robust video mosaicing through topology inference and local to global alignment, *Book Series Lecture Notes in Computer Science., Volume 1407/1998*.

SCHRADER, J. E. [2003], *Detecting and Interpreting Musical Note Onsets in Polyphonic Music*, M. Sc. Thesis, Eindhoven University of Technology, The Netherlands.

SHRESTHA, P., D. SEKULOVSKI, H. WEDA, M. BARBIERI, AND R. CLOUT [2008], Color correction using dominant colorscolor correction using dominant colors, *010809, Patent application*.

SHRESTHA, P., H. WEDA, AND M. BARBIERI [2007a], Application domains for

mixing and summarizing multiple videos, *Nat.Lab. Report TN-00140*, Philips Research, Eindhoven, The Netherlands.

SHRESTHA, P., H. WEDA, AND M. BARBIERI [2007b], Synchronization of multi-camera video recordings based on audio, *Proceedings of the 15th annual ACM International Conference on Multimedia*, 545–548.

SHRESTHA, P., H. WEDA, M. BARBIERI, AND D. SEKULOVSKI [2006], Synchronization of multiple videos using still camera flashes, *Proceedings of the 14th ACM International Conference on Multimedia*, 137–140.

SINHA, S. N., AND M. POLLEFEYS [2004], Visual-hull reconstruction from uncalibrated and unsynchronized video streams, *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*.

SNOEK, C. G. M., M. WORRING, A. W. M. SMEULDERS, AND B. FREIBURG [2007], The role of visual content and style for concert video indexing, *Proceedings of the International Conference on Multimedia & Expo (ICME)*.

STANFORD, V., J. GAROFOLO, O. GALIBERT, M. MICHEL, AND C. LAPRUN [2003], The NIST smart space and meeting room projects: signals, acquisition annotation, and metrics, *IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 4*, 736–9.

STANISLAV, S. [2004], Multi camera automatic video editing, *Proceedings of ICCVG 2004*, Kluwer Verlag, 935–945.

STEIN, G. P. [1998], Tracking from multiple view points: Self calibration of space and time, *Proceedings of the DARPA IU Workshop*, 521–527.

TAKIMOTO, M., S. SATOH, AND M. SAKAUCHI [2006], Identification and detection of the same scene based on flashlight patterns, *IEEE International Conference on Multimedia & Expo (ICME)*.

TUYTELAARS, T., AND L. VAN GOOL [2004], Synchronizing video sequences, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

UEHARA, K., M. AMANO, Y. ARITI, AND M. KUMANO [2004], Video shooting navigation system by real-time useful shot discrimination based on video grammar, *Proceedings of the International Conference on Multimedia & Expo*, 583–586.

VLACHOS, T. [2000], Cut detection in video sequences using phase correlation, *IEEE signal processing letters, Volume 7, Number 7*, 173–175.

WAIBEL, A., AND ET. AL. [2004], Computers in the human interaction loop, *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*.

WANG, Z., A. C. BOVIK, AND B. L. EVANS [2000], Blind measurement of

blocking artifacts in images, *Proceedings of International Conference on Image Processing, Volume 3*, 981–984.

WANG, Z., H. R. SHEIKH, AND A. C. BOVIK [2002], No-reference perceptual quality assessment of jpeg compressed images, *Proceedings of International Conference on Image Processing, Volume 1*, 477–480.

WHITEHEAD, A., R. LAGANIRE, AND P. BOSE [2005], Temporal synchronization of video sequences in theory and in practice, *Proceedings of the 14th Brazilian Symposium on Computer Graphics and Image Processing*, 132–137.

YAN, W., AND M. S. KANKANHALLI [2002], Detection and removal of lighting & shaking artifacts in home videos, *Proceedings of the tenth ACM international conference on Multimedia*, 107–116.

YANG, F., S. WAN, Y. CHANG, AND H. R. WU [2005], A novel objective no-reference metric for digital video quality assessment, *IEEE Signal Processing Letters, Volume 4, Issue 10*, 685–688.

YEO, B.-L., AND B. LIU [1995], A unified approach to temporal segmentation of motion jpeg and mpeg compressed video, *Proceedings of the International Conference on Multimedia Computing and Systems*, 81–88.

YOUTUBE [2009a], Test data-set used in mashup quality evaluation. `http://www.youtube.com/VirtualDirectorData#play/uploads`.

YOUTUBE [2009b], Test data-set used in synchronization method evaluation. `http://www.youtube.com/AutomaticMashup#play/uploads`.

ZETTL, H. [2004], *Sight Sound Motion : Applied Media Aesthetics*, Wadsworth Publishing.

Publications

The following publications are related to this thesis:

Technical Papers

- P. Shrestha, H. Weda, M. Barbieri, D. Sekulovski, "Synchronization Techniques for Multiple-Camera Videos Using Audio-Visual Features", To be published in *IEEE Transactions on Multimedia.*
- P. Shrestha, H. Weda, M. Barbieri, "Synchronization of multi-camera video recordings using audio", *Proc. ACM Int. Conf. on Multimedia*, pp. 545-548, October 2007.
- P. Shrestha, H. Weda, M. Barbieri, "Application domains for mixing and summarizing multiple videos", Philips Research Technical Note *PR-TN 2007/00140*, June 2007.
- P. Shrestha, H. Weda, M. Barbieri, D. Sekulovski, "Synchronization of multiple videos using flash patterns", *Proc. of the 3rd Philips Symposium on Intelligent Algorithms (SOIA 2006)*, December 2006.
- P. Shrestha, H. Weda, M. Barbieri, D. Sekulovski, "Synchronization of multiple video recordings based on still-camera flashes", *Proc. ACM Int. Conf. on Multimedia*, pp. 137-140, October 2006.
- P. Shrestha, H. Weda, M. Barbieri, D. Sekulovski, "Synchronization of multiple video recordings based on audio-visual content", Philips Research Technical Note *PR-TN 2006/00659*, September 2006.

Patent applications

- P. Shrestha, D. Sekulovski, H. Weda, M. Barbieri, R. Clout, "Color adjustment", PCT/IB2009/052557, June 2008.
- P. Shrestha, H. Weda, M. Barbieri, "System and method for generating a summary from a plurality of multimedia items", WO/2009/001278, June 2008.

- H. Weda, P. Shrestha, M. Campanella, M. Barbieri, “Method and apparatus for automatically generating summaries of a multimedia file”, WO/2008/152556, June 2008.
- M. Barbieri, H. Weda, L. Agnihotri, M. Campanella, P. Shrestha, “Method and apparatus for classifying a person”, WO/2008/047315, October 2007.
- H. Weda, M. Barbieri, M. Campanella, P. Shrestha,, “Method of creating a summary”, WO/2008/0382308, September 2007.

Automatic mashup generation of multiple-camera videos

Summary

The amount of user generated video content is growing enormously with the increase in availability and affordability of technologies for video capturing (e.g. camcorders, mobile-phones), storing (e.g. magnetic and optical devices, online storage services), and sharing (e.g. broadband internet, social networks). It has become a common sight at social occasions like parties, concerts, weddings, vacations that many people are shooting videos at approximately the same time. Such concurrent recordings provide multiple views of the same event. In professional video production, the use of multiple cameras is very common. In order to compose an interesting video to watch, audio and video segments from different recordings are mixed into a single video stream. However, in case of non-professional recordings, mixing different camera recordings is not common as the process is considered very time consuming and requires expertise to do. In this thesis, we research on how to automatically combine multiple-camera recordings in a single video stream, called as a *mashup*. Since non-professional recordings, in general, are characterized by low signal quality and lack of artistic appeal, our objective is to use mashups to enrich the viewing experience of such recordings.

In order to define a target application and collect requirements for a mashup, we conducted a study by involving experts on video editing and general camera users by means of interviews and focus groups. Based on the study results, we decided to work on the domain of concert video. We listed the requirements for concert video mashups such as image quality, diversity, and synchronization.

According to the requirements, we proposed a solution approach for mashup generation and introduced a *formal model* consisting of *pre-processing*, *mashup-composition* and *post-processing* steps. This thesis describes the pre-processing and mashup-composition steps, which result in the automatic generation of a mashup satisfying a set of the elicited requirements. At the pre-processing step, we synchronized multiple-camera recordings to be represented in a common time-

line. We proposed and developed synchronization methods based on detecting and matching audio and video features extracted from the recorded content. We developed three realizations of the approach using different features: still-camera flashes in video, audio-fingerprints and audio-onsets. The realizations are independent of the frame rate of the recordings, the number of cameras and provide the synchronization offset accuracy at frame level. Based on their performance in a common data-set, audio-fingerprint and audio-onset were found as the most suitable to apply in generating mashups of concert videos. In the mashup-composition step, we proposed an optimization based solution to compose a mashup from the synchronized recordings. The solution is based on maximizing an *objective function* containing a number of parameters, which represent the requirements that influence the mashup quality. The function is subjected to a number of *constraints*, which represent the requirements that must be fulfilled in a mashup. Different audio-visual feature extraction and analysis techniques were employed to measure the degree of fulfillment of the requirements represented in the objective function. We developed an algorithm, *first-fit*, to compose a mashup satisfying the *constraints* and maximizing the objective function.

Finally, to validate our solution approach, we evaluated the mashups generated by the first-fit algorithm with the ones generated by two other methods. In the first method, *naive*, a mashup was generated by satisfying only the requirements given as constraints and in the second method, *manual*, a mashup was created by a professional. In the objective evaluation, first-fit mashups scored higher than both the manual and naive mashups. To assess the end-user satisfaction, we also conducted a user study where we measured user preferences on the mashups generated by the three methods on different aspects of mashup quality. In all the aspects, the naive mashup scored significantly low, while the manual and first-fit mashups scored similarly. We can conclude that the perceived quality of a mashup generated by the naive method is lower than first-fit and manual while the perceived quality of the mashups generated by first-fit and manual methods are similar.

Acknowledgements

First of all, I would like to thank my promoter Emile Aarts for providing me the opportunity to conduct my Ph. D. research at Philips Research and for continuously guiding me along the way. In addition to his excellent supervision, I've been impressed by his vision, creativity and enthusiasm. After each of our regular meetings, together with Hans and Mauro, I felt more determined and inspired.

Very special thanks to my co-promoters Hans Weda and Mauro Barbieri, without them this thesis would not have been possible. I will always be indebted to their coaching, support and tireless reviews despite changes in their research area. I admire Hans's attention to details and his sincere efforts for perfection; and Mauro's command in the research topic and his dedication. Since I started in the MACS project 4 years ago, I have enjoyed our team work, meetings, lunches and dinners, both work related and casual.

I would like to thank all the members of the reading committee, Professors Jan Biemond, Arnold Smeulders and Berry Eggen for reviewing this thesis. The elaborate feedbacks from Professor Eggen helped a lot to improve the quality of the thesis.

I would like to express my gratitude to many people from Philips Research, who helped me during the research: Jaap Haitsma for the audio-fingerprinting code; Marco Campanella for the camera motion detection code; Martin McKinney, Janto Skowronek and Mun Hum Park for the audio-classification and audio-onset extraction code; Pedro Fonseca and Marc Peeters for the photo clustering library; Merlijn Sevenster for his suggestions on the optimization algorithms; Gerard Bloemen for making manual mashups; Jettie Hoonhout for helping on user studies and statistics, Dragan Sekulovski for the dynamic programming code and many other useful suggestions. Their expertise and timely help has made this thesis a lot easier to complete. Special thanks to Jan Korst, who excellently coordinated the research with the Dutch government project MultiMediaN.

I am grateful to all the members of the Experience Processing group for providing an excellent working atmosphere and co-operation. Many thanks to my group leader Hans van Gageldonk for supporting and motivating me throughout the research period. I am fortunate to have had roommates Marco Campanella and Bart Kroon, with whom I not only shared the office space but also a part of my life.

They deserve a special thank for providing technical tips, lively company and every possible help. I am also thankful to Tsvetomira Tsoneva and Francesco Scalori for their friendship and support, without which, working in the group would have been less fun.

I would also like to thank Ilona Hein for the concept design of a web-based interface of multiple-camera recordings and Frank Crienen for allowing us to use his personal multiple-camera recordings.

Finally, I would like to thank my parents Krishna Bhakta and Rajeshowri and the rest of my family especially Prashanta, Babak, Bahman and Suyash for their constant support, encouragement and being there whenever I needed them.

Curriculum Vitae

Prarthana Shrestha was born in 1975 on September 1st, in Kathmandu, Nepal. She completed her secondary education at the Mahendra Bhawan (1991) and senior secondary education at the Modern Indian School (1993), both in Kathmandu. She started her undergraduate study in Electrical and Electronics Engineering at the Bangladesh Institute of Technology, Rajshahi, Bangladesh in 1993 and graduated in 1998. She worked at the Kathmandu University as a lecturer at the department of Electrical and Electronics Engineering from 1999 till the summer of 2000. She started her postgraduate study from September 2000 at the faculty of Information Technology and Systems in the Delft University of Technology, Delft, The Netherlands. Her Master's Thesis, supervised by Late Stamatis Vassiliadis, was entitled 'MIPS Augmented with Wavelet Transform: Performance Analysis'. After graduating from the Delft University of Technology in 2002, she started the Professional Doctorate in Engineering (PDEng) program of the Stan Ackermans Institute in the Eindhoven University of Technology, Eindhoven, The Netherlands. The research entitled 'Audio-fingerprinting in P2P networks' was supervised by Ton Kalker and Tjalling Tjalkens of the Signal Processing Systems group. After receiving the PDEng degree in 2005, she started her Ph. D. research as a *van der Pol Junior* at the Experience Processing group, Philips Research, Eindhoven, The Netherlands. The research was financially supported by the Dutch BSIK research program *MultimediaN*. She worked on synchronizing and mixing recordings captured by non-professional users during musical concerts, supervised by Emile Aarts, Mauro Barbieri and Hans Weda. The work resulted in five technical publications and five patent applications. Her research interests include signal processing, algorithm development and system design for multimedia and biomedical applications.