Design and Evaluation of mProducer: a Mobile Authoring Tool for Personal Experience Computing

Chao-Ming Teng, Chon-In Wu, Yi-Chao Chen, Hao-hua Chu, Jane Yung-jen Hsu
National Taiwan University, Taipei, Taiwan
{jt, r92079, b89066, hchu, yjhsu}@csie.ntu.edu.tw

ABSTRACT

Personal experience computing is about computing support for recording, storing, retrieving, editing, analyzing, and sharing of personal experiences. In this paper, we present our design, implementation and evaluation of a mobile authoring tool called mProducer. MProducer enables a user to generate personal experience content using a mobile device anytime, anywhere. To address challenges in both limited system resources and user interface constraints on a mobile device, mProducer provides several innovative system techniques and UI designs. (1) The *Storage Constrained Uploading (SCU) algorithm* uploads large multimedia data to remote servers, in order to alleviate the problem of limited storage on a mobile device. (2) *Sensor-Assisted Automated Editing* utilizes a tilt sensor on the mobile device to automate the detection and *removal of blurry frames* resulting from excessive amount of camera shaking. This sensor-based solution requires small processing overhead, and it is considered a good alternative to computational-expensive image processing techniques for detecting shaking artifacts. (3) *Map-based content management interface* incorporates a GPS receiver on a mobile device to record location meta-data for each recording captured by a user, and enables easy, intuitive content navigation on a small screen. (4) *Keyframe-based editing* enables a user to edit content using only keyframes. We have conducted user studies to evaluate overall editing experience, user satisfaction in the editing quality, task performance time, ease-of-use, and learnability. The results of user studies have shown that keyframe-based editing works best with a storyboard interface. In general, users have found mProducer to be both fun and easy to use on a mobile device.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous; H.5.1 [Information Interface and Presentation]: Multimedia Information Systems; H.5.2 [User Interfaces]: Graphical user interfaces (GUI)

General Terms

Algorithms, Design, Experimentation, Human Factors.

Keywords

Personal Experiences, Multimedia Editing Tools, Sensors, Mobile User Interfaces

1. INTRODUCTION

The proliferation of camera-equipped phones and PDAs comes as a result of consumers' demand *not only* to be mass media consumers, but also content producers of their own personal experiences anytime, anywhere: where they go, what they do, and what they see and hear. The ability to record, edit and share footage of users' daily activities can be a strong selling point for these mobile devices with content producing capability.

Given the popularity of camera phones, it is expected that mobile content will be dominated by *personal experiences* produced by casual users. This is in contrast to the desktop computing world that targets professional content providers creating mass media content.

We believe that users are motivated to edit personal experiences directly on a mobile device, rather than to transfer content to a PC for editing. The motivations are that (1) they want to share their personal experiences *anytime, anywhere from a mobile device* – but prior to sharing them, they may want to perform simple editing functions to remove non-essential content or to add text or audio annotations; (2) they want to record important events as keepsakes – but given limited mobile storage, they want to keep only the essential content on a mobile device by removing unwanted recordings; and (3) typical users with little or no prior computing experience prefer to use a simple and intuitive user interface designed specifically for the mobile environment, rather than sophisticated PC-based tools that require a higher level of computer skills.

We would like to differentiate mProducer from many of the well-known desktop video editing tools, such as Adobe Premiere^{TM}, Apple's Final Cut Pro^{TM}, Cyberlink PowerProducer^{TM}, Ulead VideoStudio^{TM}, and others. These desktop tools target professional video editors who are skillful with video editing and work in a focus, resource-abundant desktop computing environment. These tools are not intended for casual consumers who may be unfamiliar with web technologies, nor are they designed for a resource-poor, input-limited mobile device environment. Specifically, the design of mProducer considers the following mobile challenges:

1. **Limited Storage**: Mobile devices have limited storage that restricts the length of recordings a user can capture. For example, the Toshiba T08 mobile phone, a mobile device released in 2003, provides only 8MB of storage which is

sufficient to record about 3 minutes of video at 5 frames per second and 240x320 resolution (without audio). Three minutes of recording time is too short for many of the personal experiences a user may wish to record.

2. **Limited Computing Resource**: Most image/video processing techniques for media editing are computationally intensive and demand the high computational power of PCs. They are beyond the limited computing resources on a mobile device.
3. **Specialized User Interface**: Small screens, inconvenient input methods, limited mobile user attention, and typical consumers with little computing experience require a different interaction model and user interface design, where simplicity, ease-of-use, and good learnability are as important as the final quality of edited contents.

Although the idea of a multimedia authoring tool for mobile devices has been raised in [1][9], we have yet to find a tool that address these challenges. In this paper, we describe our design, implementation and evaluation of mProducer, a mobile authoring tool which successfully addresses the challenges outlined. Our contributions include the following novel solutions:

- **Storage Constrained Uploading (SCU)**: In order to overcome limited storage, video frames in content captured by a user are prioritized based on whether they will be needed during the user editing phase. When a mobile device runs low on local storage space, lower priority frames not needed for later editing will be uploaded to a server, allowing more contents to be captured on a mobile device. This technique can increase the size of contents captured on a mobile device by 14 times.

- **Sensor-assisted Automated Editing**: Existing desktop video editing tools use image-based processing methods to semi-automate editing on raw contents so that the amount of user effort can be reduced. Examples of these techniques include object recognition, location determination [11], lighting detection, and shaking artifacts removal [14]. Although these techniques are generally too computationally intensive to run on a resource-poor mobile device, sensors attached to the device can automatically achieve a similar result with relatively small computing cost. We describe how to use GPS and tilt sensor to automate editing for users.

- **Location-based Content Management:** When mProducer is used for capturing personal experiences at multiple locations (e.g., a trip covering multiple sightseeing venues), our studies have found that users mentally organize personal experiences based on the locations where the video clips were captured. To match users' *location-based mental model*, a simple, intuitive, map-based content management interface is designed to enable easy navigation and browsing of video clips. A GPS receiver on a mobile device is used to record location meta-data for each recording captured by a user, so that a user does not have to input it.

- **Keyframe-based Editing**: A keyframe is defined as a video frame that best represents a shot or a scene, i.e., a user can get a good understanding of what a shot is about by simply looking at its keyframe. Our user studies have shown that casual users can edit using only keyframes and produce satisfactory editing quality for the purposes of sharing and recording personal experiences. In addition, keyframe-based editing allows casual users to perform basic editing functions (merging or deletion of video content, text annotation, etc.) on average twice more efficiently than traditional frame-by-frame editing.

The rest of this paper is organized as follows. Section 2 proposes the design of mProducer. Section 3 describes the storage constraint uploading algorithm. Section 4 explains how tilt-sensor is used in sensor-assisted automated editing. Section 5 explores the design of mProducer's user interface (UI), including GPS-enabled, location-based content management and keyframe-based editing. It also describes user studies to evaluate mProducer's UI. Section 6 discusses related work. Section 7 presents our conclusion and future work.

2. DESIGN

We assume mProducer runs on a camera-equipped mobile device with limited storage space. The mobile device communicates with a server through a wireless network such as 3G or 802.11. When the mobile storage is full, an uploading mechanism transfers parts of the content not needed for editing to the server. When a user finishes editing a video clip and is ready to share it, the remaining parts of contents are also uploaded to the server and removed from the mobile device. This allows mobile storage to reclaim space that can be used to record additional contents.

The current mProducer prototype covers two phases: *capturing phase* and *editing phas*e. Typical usage of mProducer involves repeating patterns of capturing one or more clips, editing these clips (which frees up space in the mobile storage), then refilling the freed space with newly captured clips.

2.1 The Capturing Phase

Figure 1 shows the execution flow within the capturing phase, starting with data captured from a mobile device and finishing with either storing the data on the mobile device or the server. In the 1st step, the camera and microphone on a mobile phone capture video and audio data in a buffer. The 2nd step applies the Shot Boundary Detection (SBD) algorithm[1] to divide a clip into disjoint shots[2] or scenes. In the 3rd step, data from a tilt sensor is used to automatically detect and remove blurred frames resulting from excessive amount of camera shaking (more details are described in section 4). In the 4th step, motion-JPEG encoding compresses incoming bitmap frames. In the 5th step, the Keyframe Selection Algorithm (KSA) [16] finds a representative keyframe for each shot, and keyframes are assigned higher priority than non-keyframes. In the 6th step, the SCU algorithm (described in details in section 3) uses the frame priority to either upload frames to the server or store them in the mobile device.

2.2 The Editing Phase

The editing phase consists of the three steps shown in Figure 1. In the 1st step, location-based content management organizes video

[1] We implemented SBD algorithm based on color histograms described in [6].

[2] A shot is defined as one or more frames generated and recorded contiguously and representing continuous action in time and space [7].

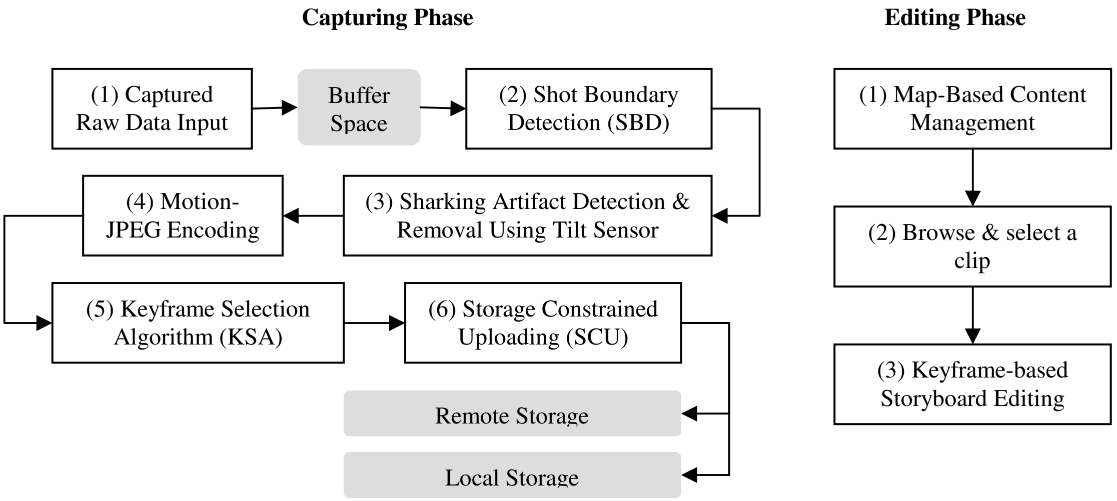

Capturing Phase
Editing Phase
(1) Captured Raw Data Input
Buffer Space
(2) Shot Boundary Detection (SBD)
(3) Sharking Artifact Detection & Removal Using Tilt Sensor
(4) Motion-JPEG Encoding
(5) Keyframe Selection Algorithm (KSA)
(6) Storage Constrained Uploading (SCU)
Remote Storage
Local Storage
(1) Map-Based Content Management
(2) Browse & select a clip
(3) Keyframe-based Storyboard Editing

Figure 1: The capturing and editing phases

clips based on their recording locations. A user starts editing video clips by first selecting a point on a map which represents recordings made there. In the 2nd step, when the user clicks on a map point, a list of clips is displayed to the user. The user then chooses a clip to edit. In the 3rd step, the user can edit the chosen clip using the keyframe-based editing interface. This interface design is described in more details in Section 5.

3. STORAGE CONSTRAINED UPLOAD

The storage constrained uploading (SCU) algorithm minimizes network communication (including both uploading and downloading) for content capturing and editing from a mobile device. We first describe the SCU algorithm in details, and then generalize this algorithm for different priority schemes.

The limited storage on a mobile devices' is barely sufficient for a user to record one complete experience. One solution to this problem is for a mobile device to upload recorded content to a server so that the amount of captured content is not limited by the mobile device's local storage. A naive approach would be to upload every piece of content to the server immediately after it is recorded, then download it back to the device whenever a user needs to edit it. The first problem with this approach is that transferring content that will later be deleted by the user is a waste of network bandwidth. The second problem is that limited wireless bandwidth is likely to result in slow content transfers, leading to a frustrating user editing experience. Therefore, we need a more intelligent mechanism to determine when to upload, and what portions of the contents to upload, from mobile storage to the server.

SCU will not upload contents to the server until the local storage space is nearly full. The reason for this is that we can avoid uploading frames that will later be cut by the user. SCU chooses frames for uploading based on the observation that there is a difference in quality requirements between personal experience editing tools targeting casual consumers, and editing tools targeting professional content providers. We believe that there is no need to provide a mobile authoring tool that can produce professional quality content. Fine-grain editing (e.g. frame-by-frame) used in a professional PC-based authoring tool for professional quality content is, in fact, unsuitable for a mobile authoring tool. This style of editing requires a significant amount of user effort, training and attention, high resolution screens, and high computational power.

When applying SCU in mProducer, frames were prioritized into two levels of importance: *keyframe* and *non-keyframe*. This prioritization is useful due to our observation that typical consumers are satisfied with editing using only keyframes. This allows a mobile device to provide editing functionality using only a subset of the total content being modified.

We define *editing granularity* to be the subset of frames used during editing. The finest editing granularity possible is frame-by-frame. The system may also only present keyframes or I-frames (for MPEG-encoded video) to users for editing. The editing granularity then becomes keyframes or I-frames. We have performed a user study to find out the granularity requirements of casual consumers. Our results have shown that typical consumers can edit and produce satisfactory quality (delete unwanted portions of video clips and add text to shots) when they were presented with keyframes only. This suggests that, for casual consumers, non-keyframes can be uploaded without degrading the editing experience. Further details about this user study are presented in Section 5.2.

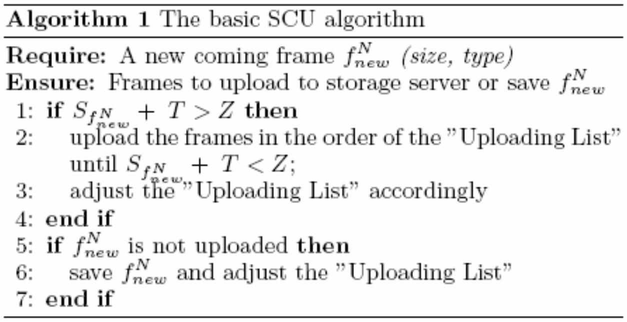
We define *storage granularity* as the type of frames that a mobile device stores locally during the capturing phase. This is a necessary parameter in the SCU algorithm given that different storage granularities dictate which type of frames to upload. For example, the algorithm may have two levels of storage granularity: *high* preserves all frames, and *low* preserves only higher priority frames.

3.1 SCU Algorithm

Initially, when mobile storage is empty, the SCU algorithm will store all frames including both keyframes and non-keyframes. The mobile storage is said to be at high storage granularity when it can store both types of frames. As a mobile user captures new frames, mobile storage may eventually run out of free space. The SCU algorithm then enters low storage granularity when a new captured frame fills the remaining space in mobile storage. While mobile storage remains low, it will start uploading non-keyframes to the server in order to make room for incoming frames. In situations where network bandwidth (upload rate) is less than the content capture rate, a buffer on the mobile device is needed to temporarily store data. Eventually such a buffer would be filled and recording will be disabled on the mobile device. At this point, the user is notified to edit some clips, which are then uploaded to the server and removed from local storage. A mobile device re-enters high storage granularity again after local storage is cleared.

If mobile storage contains multiple clips, SCU uploads frames in round robin fashion among the stored clips, in order to maintain fairness among clips. When the storage granularity drops from high to low, the uploading of frames is done on an as-needed basis. SCU does not upload all non-keyframes at once to the server. The reason for as-needed uploading is to avoid unnecessary uploading of frames that will later be cut by users, as mentioned earlier.

Consider an uploading list that tracks the order of frames to be uploaded from the mobile storage. It sorts frames based on priority first then applies round-robin scheduling across the clips. Using this uploading list, mProducer can simply look at the head of the list to choose which frames to upload next. Note that the current policy in mProducer is to never upload keyframes, even when the storage is full. The main body of SCU algorithm is shown below. We denote the reserved space for mProducer in the storage as *Z*, the size of total frames in the storage is *T*, the *i*-th frame of clip *#j* as f_i^j, its size as $S_{f_i^j}$, the newly coming frame as f_{new}^N, and *N* is the number of clips in the mobile storage. For more details for the SCU algorithm, please refer to our previous paper [2].



```
Algorithm 1 The basic SCU algorithm
Require: A new coming frame $f_{new}^N$ (size, type)
Ensure: Frames to upload to storage server or save $f_{new}^N$
if $S_{f_{new}^N} + T > Z$ then
    upload the frames in the order of the "Uploading List" until $S_{f_{new}^N} + T < Z$;
    adjust the "Uploading List" accordingly
end if
if $f_{new}^N$ is not uploaded then
    save $f_{new}^N$ and adjust the "Uploading List"
end if
```

3.2 Generalization of the SCU Algorithm

We can generalize the SCU algorithm for different priority schemes. For example, if we have MPEG encoded frames as input, I-frames can be set to 2nd highest priority, followed by P and B-frames. We then have four different levels of storage granularity, including the keyframe level as the highest priority. Uploading first removes B-frames. After all B-frames are uploaded, P-frames are uploaded, followed by I-frames. Only keyframes remain on the mobile device for editing. Another example is scalable video coding such as MPEG-4 Fine Granularity Scalability (FGS) [14].

The enhancement layers can be set at lower priority and be considered for uploading first. The base layer is assigned a higher priority and kept on the device for editing.

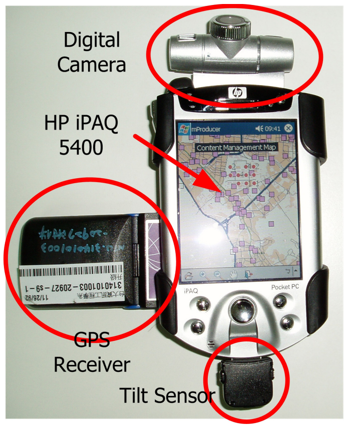


Figure 2: HP iPAQ 5450 with a digital camera, a GPS receiver and a tilt sensor

4. SENSOR-ASSISTED AUTOMATED EDITING

Existing video editing uses image processing to identify and extract meta-data *context information* at the time of production [8][11]. Sensors attached to a mobile device can achieve the same context information without high computational cost. This is ideal for a mobile device that has limited computing capability.

The current version of mProducer incorporates two sensors to automatically annotate captured contents with meta-data context information: (1) global positioning system (GPS) receiver detects location meta-data, and (2) a tilt sensor detects the amount of camera shaking. Note that excessive amounts of camera shaking results in blurry, unusable video, which can then be automatically detected and removed. A common example of unwanted video clips that can be detected by camera shaking is when a user forgets to hit the stop button after recording, leaving the device (while walking) in a pocket or a bag to continue capturing unwanted video clips. Figure 2 shows the hardware component of the prototyped system together with GPS receiver and tilt sensor.

GPS Receiver: it is the GPS-CF card from CHIPCOM Electronics. Each time a user records a video clip, mProducer will probe the GPS receiver for current location information. The GPS receiver has approximately 5 meters of accuracy outdoor. This clip will be annotated with location information. Our user studies have shown that typical consumers are more likely to merge video clips taken at the same location. This observation leads to the design of a location-based content management (described in details in section 5.1), which organizes and groups contents based on their recording locations shown on a map. This enables users to easily and quickly navigate multiple video clips.

Tilt Sensor: it is TiltControl CF card made by ECER Technology as shown in Figure 3. It contains an accelerometer that measures the horizontal and vertical tilt of the device. Changes in the tilt are used to compute the magnitude of camera shaking and predict its impact on video quality. The sensor measures both direction and magnitude of tilt.

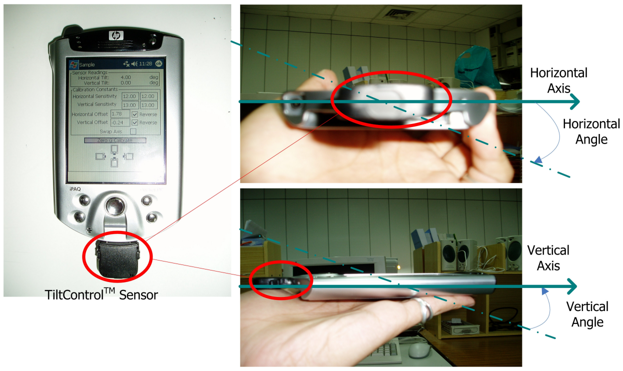


Figure 3: The TiltCONTROL Sensor

Experiment to Identify Camera Shaking Pattern

Tilt sensors can be used to detect camera shaking and automate the process of shaking artifact detection and removal. This is an ideal alternative to computationally intensive video analysis on a resource-poor mobile device. To determine the signature of camera shaking, an experiment was conducted to distinguish between excessive amount of shaking (e.g., resulting from putting the device in a pocket during walking) from moderate shaking that comes naturally with unstable hands when walking while filming. Our experiment is described below.

Data Acquisition: The TiltCONTROL sensor monitors vertical and horizontal tilt of the device throughout the experiment. A series of readings are recorded and analyzed to determine if camera shaking occurs. The sample rate of tilt sensing is set to be 200 milliseconds. The standard deviation of changes of device angles is computed for each sliding window of the most recent 10 readings.

Shaking Detection: Device shaking can be detected when changes in a device's tilt angles *oscillates* between two opposite directions. The intensity of shaking can be measured by calculating the *rate of change in device tilt angles* and the *oscillation rate*. Walking while holding a device by hands will create oscillations of smaller magnitude (see the middle graph of Figure 4). Walking with the device in a pocket will also create oscillations, but of larger magnitude (see the right graph of Figure 4). For the experimental setup, we measured three activities for each participant:

Table 1: Standard deviations on the magnitude of oscillations and frequency of oscillations for three activities

	Standard deviation on tilt angle degree changes		Frequency of oscillations (per second)	
Activities	Horizontal	Vertical	Horizontal	Vertical
Sitting	2.62°	3.00°	1.36	0.76
Walking	5.27°	7.13°	1.89	1.97
Pocketing	64.72°	75.96°	1.73	1.85

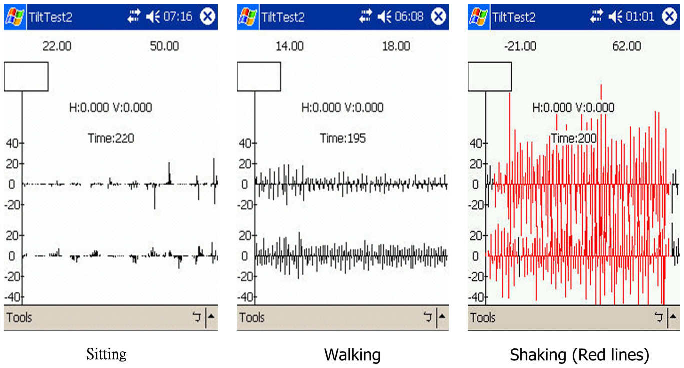


Figure 4: Measured oscillation magnitudes for three activities: (1) holding the mobile while sitting, (2) holding the device while walking, and (3) putting the device in a pocket. X-axis represents time. Y-axis represents the magnitude as change of degree per unit time.

(1) Holding the mobile device while sitting or standing still for 2 minutes (collecting 591 samples);

(2) Holding the mobile device while walking for 2 minutes (collecting 591 samples); and

(3) Putting the PDA in a pocket or a bag while walking for 2 minutes (collecting 591 samples).

Result: Based on empirical data shown in Table 1, we determine two conditions for excessive shaking: (1) the standard deviation of the tilt angles is larger than 20° (degrees) – it is calculated by 89.9% of actual shaking frames (externally observed) having higher standard deviation values than this threshold value, and (2) the frequency of oscillations in both directions exceeds 1.5 oscillations per second – again, it is calculated by 76.5% of actual shaking frames having higher value than this threshold value. In Figure 4, we depict a partial result of one participant's experiment. We can see from this figure, under the normal case, that the standard deviation is small, and the vibration is moderate. Walking introduces constant vibration, but the standard deviation is below 20°. When shaking, we can see that the standard deviation is high and the vibration is frequent. This pattern helps the system to detect camera shaking with a simple computation of standard deviation, which demonstrates how sensor measurements may assist in processing video content using simple computation.

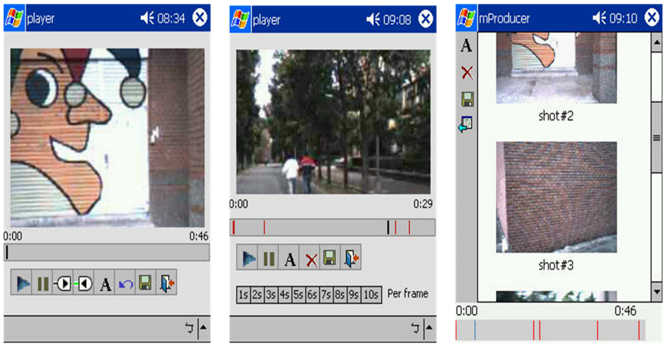
5. USER INTERFACE DESIGN

The design of a mobile user interface needs to consider small screen size, inconvenient input methods, limited user attention, and limited user computing experience. A key design challenge is to understand the tradeoff between simplicity (ease-of-use, short learning curve, and reduced user effort) and *quality of edited production* (which provides a rich feature set but comes at a cost of increased user effort). In addition, the UI design needs to accommodate the system storage constraints on a mobile device.

The mProducer UI consists of two parts: location-based content management and keyframe-based editing. They are described in the following subsections.

5.1 Location-based Content Management

We have conducted an informal user study to find out the preferred manner, of casual users, to navigate or browse video



(a) Frame-by-frame Playback (b) Keyframe-based Playback (c) Keyframe-based Storyboard

Figure 5: Screen shots for three editing user interface (frame-by-frame editing UI on the left, keyframe-only editing with slideshow player in the middle, and keyframe-only editing with storyboard player)

clips. In general, there are two ways they mentally group clips: by *recording time* or by *recording location*. They reported that, in general, they prefer to navigate based on location instead of time. Users told us that they can make stronger mental associations between video clips and visual locations rather than times, i.e., they can better remember specific locations where they recorded video clips, rather than the specific times when they recorded video clips. We believe that the reason is location information is more visual (hence easier to remember and make associations), whereas time information is more abstract. With the help of the GPS receiver, we were able to automatically annotate each video clip with its recording location. This removed the need for a user to manually input the location meta-data. With location information, clips are organized and grouped by points on a map, rather than in directories for a file browser.

5.2 Keyframe-based Editing

There have been several applications that use keyframes extracted from video clips. One of these uses keyframes to expedite video browsing [5][10]. It has been shown that users can get a good understanding of video clip content by browsing only their keyframes [12]. We would expand on understanding to investigate keyframe editing, i.e., we would like to know if users can edit using only keyframes and still produce satisfactory quality for sharing personal experiences. We realized that editing content is more demanding on a user in comparison to simply getting general understanding of contents. We also admitted the possibility that keyframe-based editing may reduce the editing quality in exchange for decreased user effort; therefore, it provides an alternative point on the design trade-off curve shown in Figure 6, between simplicity (the amount of user efforts) and quality of editing. To better understand this design trade-off, we have performed a user study to investigate the effectiveness of keyframe-based editing, specifically:

- The reduction in user-perceived quality and whether the produced contents were acceptable to them;
- The reduction in user efforts or improvement in task performance;

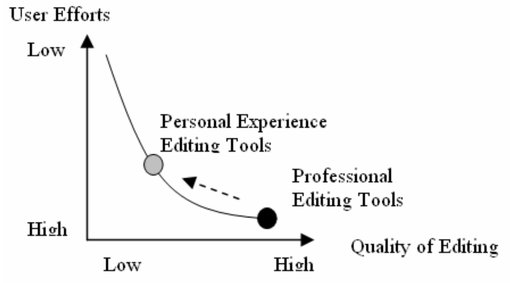


Figure 6: Tradeoffs between user efforts and quality of editing

- Keyframe-based editing's effectiveness when combining with either a slideshow player[3] or a storyboard player[4].

If the editing quality drops only slightly and the task performance improves significantly, we can say that keyframe-based editing offers a good design trade-off for mobile computing environment.

5.3 User Study #1

The user study consists of testing the following three user interfaces:

- **(UI-A)**: Frame-by-frame editing with a video player (the scaled-down version of conventional desktop editing interface);
- **(UI-B)**: Keyframe-only editing with slideshow player; and
- **(UI-C)**: Keyframe-only editing with storyboard player.

Participants were asked to capture and edit video clips using each of three editing interfaces shown in Figure 5.

Independent Variables: The three editing interfaces detailed above.

Dependent Variables: Task performance measures the amount of time to complete editing tasks using a selected editing interface. Subjective satisfaction ranks the interfaces in terms of overall editing experience, the user's perception of quality of editing, ease of use, and ease of learning.

Participants: We randomly chose eleven participants (eight males and three females) on campus for this user study. Their ages range from 20 to 41 years, with a mean of 24. Three of them (all male) have previous experiences in using a PDA. Five of them (four male and one female) have previous experiences in using PC video editing tools. None of them had previous experience in using mobile video editing tool. All participants have used cell phones.

[3] A slideshow player displays an image to a user, waits a short period of time, and then displays the next image in a sequence, which may be random or ordered.

[4] A storyboard player displays multiple still keyframe images at once, representing pivotal frames from a sequence, in order to understand a clip. The storyboard player differs from a slideshow player in that the storyboard player allows a user to see keyframes from adjacent shots the same time, whereas a slideshow player allows a user to see one keyframe (shot) at a time.

Software and Hardware Equipment: Each participant was provided with mProducer running on an HP iPAQ 5500 mounted with a GPS receiver and a digital camera.

Procedures: The evaluation is consisted of four sessions: introduction/training, capturing video clips, editing video clips, and filling out a questionnaire as part of a face-to-face interview.

1. Participants were briefed on the goal and the procedure of the user study. We demonstrated how to capture videos using the PDA and how to edit using each of the three interfaces.
2. Each participant was asked to record a total of 6 minutes of video containing three 2-minute clips. Examples of content captured included scene-recording, self-introduction of people in a group, and a specific event.
3. The participants were asked to edit three clips, each using one of the three different editing interfaces. The editing task involved removing unwanted content from the raw video clips. We measured the length of time it took to complete each editing task for each participant. Note that the assignment between clips and editing interfaces were chosen randomly to reduce the first clip bias[5].
4. Each participant filled out a questionnaire with demographic information including age, sex, and experience with video editing tools. The questionnaire also asked each participant to rate the three editing interfaces in four terms defined in Table 2.

Results in Task Performance: We recorded the time each participant took to complete editing a two-minute video clip for each of three clips. The results are shown in Figure 7. The mean task completion time for each UI is: (UI-A) 4 minutes and 32 seconds, (UI-B) 3 minutes & 58 seconds, and (UI-C) 2 minutes and 48 seconds. Ten out of eleven participants completed the editing task fastest using (UI-C). All participants finished editing sooner using (UI-B) in comparison to (UI-A). The result shows that users can perform editing tasks *more* efficiently using a keyframe-only editing interface. In addition, the keyframe-only storyboard editing interface provided the best task completion time.

Based on our interviews with participants, they reported that the storyboard UI helped them by enabling them to see several keyframes at the same time. They could quickly identify which frames or shots they did not like and remove them. Some participants also mentioned that their problem with frame-by-frame editing was that it required uninterrupted, focused attention on the screen. However, many elements in the mobile environment can be distracting and make it difficult for a user to maintain continuous attention for a long period of time. For examples, friends calling, people walking by, and surrounding noise can all temporarily distract user attention from the editing task. This makes frame-by-frame editing over a long clip difficult in a mobile environment.

Table 2: Ratings on three editing interfaces

	Questions (Rank three editing UIs)
1	Perceived quality of editing
2	Ease-of-use
3	Ease of learning
4	Overall editing experience

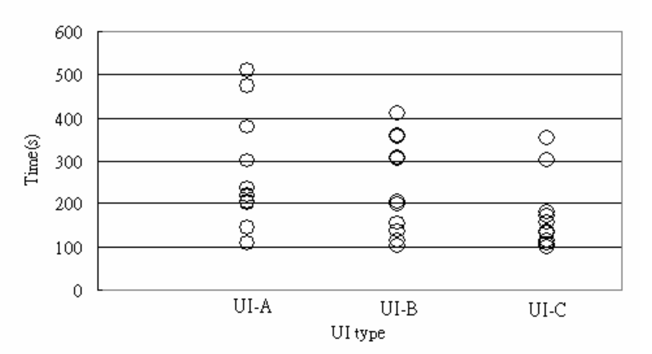


Figure 7: Task completion time

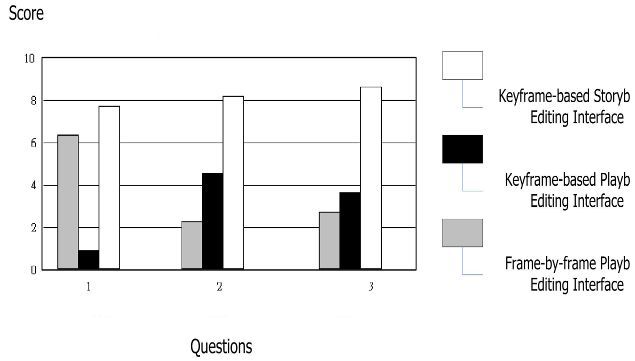


Figure 8: Response to questions in Table 2. The score is computed by giving the highest-ranked UI 10 points, the second highest-ranked one 5 points, and the worst one zero point. It shows the average score over 11 participants.

Results in Subjective Satisfaction:

Participants answered the questions listed in Table 2. Their responses to the first three questions are shown in Figure 8. The results show that users rated keyframe-only storyboard editing as producing superior editing quality. Our explanation is that when using frame-by-frame editing, casual users are not willing to spend time to find good mark-in and mark-out boundary points for unwanted content. Because of this, they find our SBD algorithm can find better boundary points for both wanted and unwanted shots. The results also showed that users rated keyframe-only storyboard editing to have the best ease-of-use and ease-of-learning. We were told that the advantages of the keyframe-only storyboard interface were that (1) it allows users to quickly move among shots, which is useful during editing, and (2) it allows users to quickly delete unwanted shots by a single-click on the keyframes corresponding to these shots.

The results for overall experiences in the three editing interfaces showed that UI-C (keyframe + storyboard) was consistently selected as most satisfying from all participants (100%), and 64%

[5] Participants may be least familiar with the first clip they recorded and might be less efficient in locating and removing unwanted portions of it.

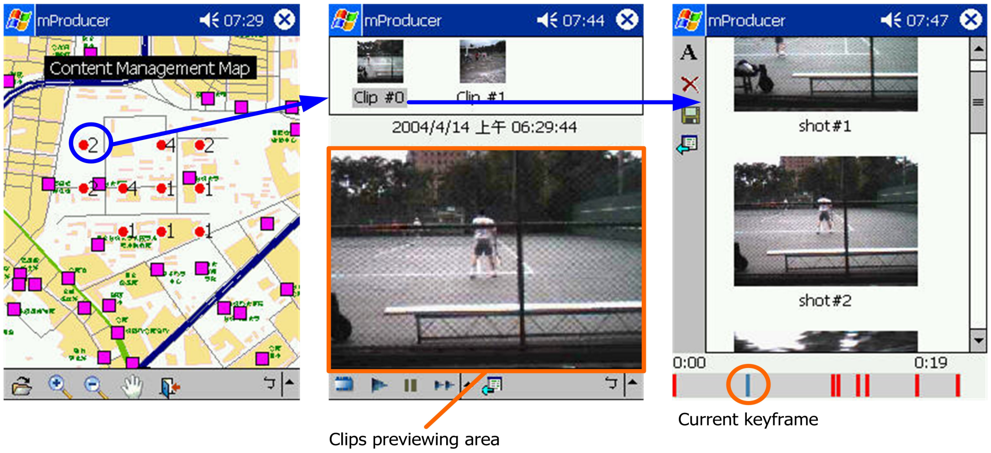


Figure 9: mProducer user interface showing the location-based content management (left screen), material pool (middle screen), and storyboard interface (right)

(seven) of the participants found UI-B to be more satisfying to use than UI-A.

5.4 User Study #2

We conducted user study #2 to evaluate the overall experience of mProducer due to location-based content management interface and keyframe-only storyboard editing interface. The left screen of Figure 9 shows what a user sees when starting to edit video clips. On the map, dots are used to represent locations where contents were captured. Initially, we tried to use thumbnails instead of dots on map, but the PDA's small screen became cluttered with only a small number of video clips. Users can use the map interface to navigate (zoom in, zoom out, or move the map) and find clips to edit based on the location information. The middle screen of Figure 9 shows a *material pool* containing all clips captured at a specific location. The material pool screen is shown after the user clicks on a dot on the map. We provide keyframe previews for users to quickly decide which clip to edit. On the list of clips, one can see the time, date and the duration of the recorded content. The right screen of Figure 9 shows the keyframe-only editing UI.

Participants: We observed seven participants using mProducer to record video clips. Five were male and two were female. The ages of users varied from 21 to 33 years old, with the average being 23.8 years. Three have had previous experiences using PDA, while all have used cell phones. Three had previous experiences with desktop PC video editing tools. One of them had previous experience with a mobile device's video editing tool. All were chosen randomly on campus.

Software and Hardware Equipment: Each participant was provided with mProducer running on an HP iPAQ 5450 mounted with a GPS receiver and a digital camera.

Procedure:

(1) Participants were briefed on the goal and the procedure of the user study. We demonstrated how to capture and edit video using the PDAs.

(2) Participants were asked to shoot any type of footage they wanted. They were encouraged to walk around campus, and record what they found interesting. We asked them to record about 10 minutes of footage with any number of clip(s).

(3) Participants used the editing component of mProducer immediately on the content they had produced. They were asked to edit two clips chosen randomly from the pool of clips they had recorded. During the editing sessions, participants were asked to "think aloud" in order to let us know their intentions and the cognitive process of using mProducer.

(4) After the editing session, participants were asked to fill out a questionnaire and discuss their overall experiences using mProducer. The questionnaire included questions about demographical information, participants' previous experiences with mobile devices and video editing tools, their impression of the mProducer tool (before and after using it), their experiences of navigating among different clips and editing the two clips they chose, and any other improvements they thought we could make.

Result in Overall Experience: In general, participants' feedbacks were very positive. One of the participants described mProducer as "*a pretty cool tool to use*." Another participant said that "*the keyframe-only storyboard is very helpful for me to delete contents that I do not like. Editing tools on desktop PCs should incorporate this feature too*!" "*Map based content management is very informative for choosing which clip to edit*", said the other.

All participants said that editing with a keyframe-only storyboard interface was fast and easy. Some of the participants mentioned that the slideshow interface was better for getting a rough idea about the clip, while the storyboard interface was better for editing. Therefore, they suggested that the UI gives users the option to switch between these two interfaces. One participant suggested that we allow for location tracking of indoor recordings where the GPS receiver does not work. Some participants said that the content management map sometimes responds slowly[6].

6. RELATED WORK

The Toshiba T-08 cell phone [13] is a commercial product that comes with its own video editing tool. Since it does not provide any uploading functionality, it only allows users to record 3

[6] This was due to the limited computation power on the PDA

minutes of video clips at five frames per second on its 8 MB storage. Its UI is a smaller version of a frame-by-frame editing interface, but for a 3 minute video clip at a low frame rate, frame-by-frame editing is probably manageable. However, for long video clips recorded at a higher frame rate, a frame-by-frame editing interface would be difficult to use in a mobile environment.

Jokela presents an overview of the key opportunities and challenges in developing tools for authoring multimedia content in mobile environments [9]. However, no solutions were provided. Lara et al. in [3] described a collaborative mobile authoring tool that allows authors to collaboratively download and edit content with different fidelity. They address the replica inconsistency problem occurring when revisions at different fidelities are merged. However, they do not address the limited storage and mobile UI issues.

Hitchcock [1] is a PC tool that uses keyframes to speed up editing of home videos. It displays keyframes in piles (based on color similarity of keyframes) for selection, and a storyboard to drag-and-drop keyframes (shots) according to the sequence of shots the user wants. Since mProducer runs on a PDA with a much smaller display, the idea of presenting shots in piles was not a workable solution. In addition, it is not possible to have both the keyframe presentation area and a storyboard on a small mobile screen at the same time.

7. CONCLUSION AND FUTURE WORK

We describe our design, implementation, and evaluation of a mobile authoring tool called mProducer that enables a mobile user to capture and edit personal experiences from a mobile device anytime, anywhere. MProducer addresses the challenges of both limited system resources and user interface constraints on a mobile device.

We have designed the Storage Constrained Uploading (SCU) algorithm, which uploads potentially large multimedia contents to servers, in order to alleviate the problem of limited storage on a mobile device. A GPS receiver was added to a mobile device to record location information for each piece of content captured by a user, and provide a map-based content management interface to enable easy, intuitive navigation from a small mobile screen. We incorporated a tilt sensor on a mobile device to automate the detection and removal of blurry frames resulting from excessive amount of shaking. This sensor-based solution requires small processing overhead, and is considered a reasonable alternative to computationally-expensive image processing techniques to detect shaking artifacts. We have designed a keyframe-only editing interface, and conducted user studies to evaluate the overall user editing experience (ease-of-use and learning curve), task performance time, and quality of the edited product. Overall, users found mProducer to be both easy and fun to use on a mobile device.

Since cell phones are more popular than PDAs, we are in the process of porting mProducer onto a cell phone platform. We are interested in finding out how well our UI design would work on a cell phone with an even smaller screen than a PDA.

Editing video clips is more meaningful if they can be shared with other people who are interested in viewing them. Our future work will exploit new methods to conveniently disseminate personal experience recordings.

8. REFERENCES

[1] A. Girgensohn *et al*. Home video editing made easy: Balancing automation and user control. In Human-Computer Interaction: INTERACT, 2001.

[2] Reference removed for double blind reviewing.

[3] E. de Lara *et al*. Collaboration and multimedia authoring on mobile devices. In Int'l Conf. on Mobile Systems, Applications and Services (MobiSys), 2003.

[4] R. Sarvas *et al*. Metadata creation system for mobile images. In Int'l Conf. on Mobile Systems, Applications and Services (MobiSys), Jun 2004.

[5] F. C. Li *et al*. Browsing digital video. In CHI 2000, pages 169–176, Apr 2000.

[6] U. Gargi and R. Kasturi. An evaluation of color histogram based methods in video indexing. In International Workshop on Image Databases and Multimedia Search, 1996.

[7] G. Davenport, T. Aguierre Smith, and N. Pincever. Cinematic primitives for multimeidia. IEEE Computer Graphics and Applications, pages 67–74, July 1991.

[8] H. Zhang *et al*. Video parsing, retrieval and browsing: An integrated and content-based solution. In ACM international conference on Multimedia, 1995.

[9] T. Jokela. Authoring tools for mobile multimedia content. In IEEE Int'l Conf. on Multimedia and Expo (ICME), pages 637–640, 2003.

[10] A. Komlodi and G. Marchionini. Key frame preview techniques for video browsing. In ACM International Conference on Digital Libraries, pages 118–125, 1998.

[11] C. Snoek and M. Worring. Multimodal video indexing: a review of the state-of-the-art. Multimedia Tools and Applications, 2004 (In Press).

[12] T. Tse *et al*. Dynamic key frame presentation techniques for augmenting video browsing. In Advanced Visual Interface, pages 185–194, 1998.

[13] Vodaphone, Japan. http://www.vodafone.jp/english/.

[14] W. Yan and M. S Kankanhalli. Detection and removal of lighting & shaking artifacts in home videos. In ACM international conference on Multimedia, pages 107–116, Dec 2002.

[15] W. Li, Overview of fine granularity scalability in MPEG-4 video standard. In IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, March 2001.

[16] H. Zhang, et. al., ”Video Parsing, Retrieval and Browsing: An Integrated and Content-based Solution,” Proc. of ACM Multimedia, 1995.