AUDIO-ENHANCED PANORAMIC IMAGE CAPTURING AND RENDERING ON MOBILE DEVICES

Gerald Clemens[1], *Francesc Sanahuja*[2], *Christophe Beaugeant*[1]

[1]Siemens AG, COM, Mobile Phones, Haidenauplatz 1, 81675 Munich, Germany
[2]Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany
Email: firstname.lastname@siemens.com

ABSTRACT

This paper presents the key techniques of a panoramic image application suite fully implemented on a camera- enhanced mobile phone. The way of capturing panoramic images has been optimized with regards to usability though taking into account the generally lower system performance available on a mobile device.

Furthermore, we introduced a new method to capture audio being associated with the captured scene. This enables to auto-scroll the panoramic image synchronized to the audio playback, resulting in a video-like user experience.

1. INTRODUCTION

Panoramic image capturing is gaining popularity especially among digital camera users. Capturing larger images than would be possible with a single snapshot increases visual impression.

Conventional image stitching as known from digital camera manufacturers is done by the user snapping several photos and subsequently the software stitching the photos on user's request as a post-recording step on a different device, e.g. a PC. However, as mobile devices evolve they are expected to implement all functional steps on one single device. With regards to panoramic image capturing this means a solution shall cover both steps from photo taking to stitching the images.

From the usability point of view such an application shall enable the user to capture panoramic images within a short period of time with a maximum of support by the application, still providing an intuitive user interface.

Moreover, using the digital camera of a mobile phone to take the panorama these devices inherently offer the possibility to record audio samples during capturing the panoramic image, making it a real multimedia application. Thus, when later on viewing the panoramic image in the playback mode, it can be autoscrolled synchonously to the audio.

According to the previous remarks, we built an application suite that covers two important parts: the record view to capture a panoramic image and the playback view to render a previously recorded panoramic image.

The process of creating a panoramic image (in the record view) is covered in section 2. We present a solution to real-time capturing of a panoramic view which we refer to as Real-Time Stitching (RTS), then a solution for adding audio being associated with the image, and we give a global overview of our record view mode.

Section 3 deals with the rendering of a panoramic image recorded according to the process presented in section 2.

Finally, section 4 focuses on the compliance issue; when dealing with mobile telephony and networking, sharing such images - e.g. via MMS or email, respectively - may be of high interest for connectivity, giving to the application its full multimedia dimension.

2. RECORDING AN IMAGE

2.1. Real-time stitching

Usual technics to compute a panorama view from images taken by a digital camera are based on the principle of taking several pictures and to stitch these pictures 'off line', i.e. in a PC environment. To stitch the images, a correlation between the pictures is computed to adjust the whole final panoramic image. As an example, for two adjacent pictures of size $L \times H$ pixels the displacement between the images can be found by maximizing the correlation between the luminance values of two windows (W_k and W_{k+1}) of size $n \times m$ pixels, where W_k is the centered cut-out region of the first image, and W_{k+1} is a cut-out region of the next image which is displaced horizontally by u and vertically by v pixels relatively to its center:

$$S_k(u^*, v^*) = \min_{1 \le u \le H-m, 1 \le v \le L-n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left| W_k(i,j) - W_{k+1,(u,v)}(i,j) \right| \tag{1}$$

with $W_k(i,j)$ returning the luminance value of pixel (i,j) of window W_k; now, the two images can be stitched according to the found displacement (u^*, v^*). The derivation of equation 1 can be found in [1]. We use luminance based pattern matching because it is suitable for mobile phones due to its low performance needs.

When digital camera and the image stitching device like a PC are separate devices, it is advisable to take as few photos as possible from the scenery a panoramic image shall be composed of in order to save memory space, even though from a distortion point of view more photos would improve the stitching result qualitywise.To overcome this problem, RTS can be applied, given the capturing device has enough performance.

The fundamental of the RTS algorithm is that instant images are joined to the panoramic image being created as soon as they are available from the camera, which can be described then as a video camera delivering frames. The stitching is done automatically by means of a pattern matching algorithm based on correlation similar to Eq [1] applied to the so far created panoramic image and each new frame. Furthermore, the image grabbing is not released manually but the application grabs images from the camera sensor at a certain frame rate. In the sense that the stitching of a frame is done before the next frame is available this kind of processing can be referred to as a real-time process.

Using this method, the (growing) panoramic image and one image frame from the camera must be held in memory. Thus, additional RAM memory for the panoramic image - or at least the part of it necessary for correlation - must be provided when being compared to a conventional photographical snapshot application. On the other hand, only the complete resulting panoramic image is stored to the flash file system making it possible to discard the single frames immediately after they have been processed.

A recording session using the proposed RTS algorithm basically runs as depicted in the flow chart of fig.1:
As soon as the user starts the panoramic image recording, the camera sensor grabs images continuously and delivers them to our RTS algorithm. The first frame is the starting point for the panoramic image generation and is therefore just copied to the panoramic image memory buffer. For each following frame the RTS algorithm finds the displacement referring to the point where an image frame has been stitched to the panoramic image the last time. If the displacement exceeds a predefined value the new frame is stitched to the so far panoramic image.

A significant advantage of using RTS in a panoramic image recording application is that the knowledge of the displacement of each image frame with respect to the previous

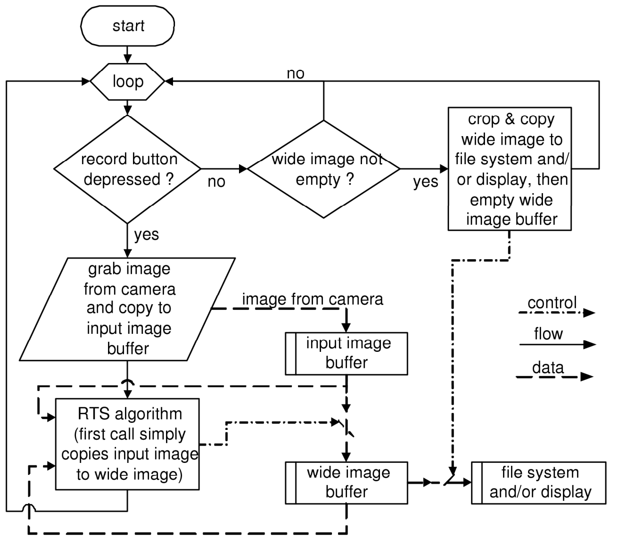
start
loop
no
record button depressed ?
no
wide image not empty ?
yes
crop & copy wide image to file system and/ or display, then empty wide image buffer
yes
grab image from camera and copy to input image buffer
image from camera
input image buffer
control
flow
data
RTS algorithm (first call simply copies input image to wide image)
wide image buffer
file system and/or display

Fig. 1. Panoramic image recording session using the Real-Time Stitching (RTS) algorithm

ones found by the algorithm enables to improve usability dramatically above conventional panoramic image recording applications by giving immediate feedback to the user already at recording time.

This could be by indicating the progress of the recording in terms of angular degress referring to e.g. 360 (see progress bar in fig. 2). This progress *p* (percentage) can easily be calculated from the distance *d* (pixels) of the latest attached frame to the first frame copied to the panoramic image and the camera sensor's characteristics aperture *a* (angular degrees) and pixels per degree *ppd*:

$$p = \frac{a + d/ppd}{360} \times 100 \tag{2}$$

Another indication of high interest to the user is if he is sweeping the camera in such a way to maintain the maximum height in terms of a horizontal panoramic image:
If the panoramic image is later on cropped to be rectangular and contain recorded image information on every pixel, the maximum height of such a panoramic image can never exceed the height of the images grabbed from the camera sensor. In order to create such an image of maximum height, the user should possibly not deviate from the vertical position of the first image grabbed during the recording. To support him in this intention, the vertical deviation from the optimum vertical position can be indicated by means of an artificial horizon (see fig.2), where the middle is optimum; the most upward and most downward position on the scale are programmatically chosen such as to maintain a predefined mimimum height of the image. The application shall position the indicator on a scale from -100 to +100 during

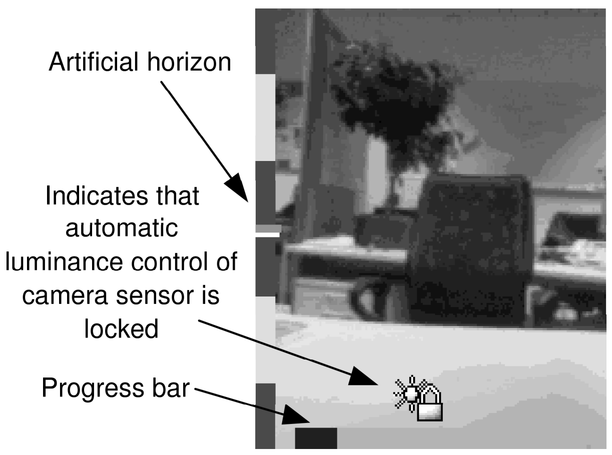
Artificial horizon
Indicates that automatic luminance control of camera sensor is locked
Progress bar

Fig. 2. Screenshot of our application in **recording** state; it records a panorama by sweeping the camera to the right.

the recording as calculated from the following equation:

$$ind = \frac{(pos - disp - top + dimI/2 - dimO) * 200}{bot - top + dimI - dimO * 2} - 100 \tag{3}$$

, where pos denotes the vertical position of the latest stitched image frame's center within the so far created panoramic image, $disp$ the vertical displacement between the grabbed image and the panoramic image, top denotes the upper and bot the lower horizontal line within which the panoramic image contains valid image information, $dimI$ is the height of the input images, $dimO$ is the predefined minimum height of the panoramic image not to be undershot; all parameters are in units of pixels, and the origin of every coordinate is the upper left corner. As long as $-100 \leq ind \leq 100$ the panoramic image has at least the height of $dimO$, where $ind = 0$ means the optimum vertical position the user should try to keep; if ind reaches a value outside the previously defined range, then the RTS is cancelled and a decision on the storage of the panoramic image is asked for. Certain thresholds in between may be introduced to output further messages to the user.

As can be seen from the formula, the more the user already has deviated from the optimal position - and hence the height $bot - top$ of the generated image decreased - , the more sensitive the indicator will react, and the scale itself will automatically be adapted (including the optimum position).

2.2. Additional synchronous audio

As an additional feature various digital cameras offer the possibility to enhance the image by giving the user the opportunity to record an associated audio stream; the storage of this audio stream is done either separately or this may be done in the same file. Using such a an embedded audio format we propose to save the audio to the user comment segment [2] of the JFIF (JPEG File Interchange Format, [3]) file wherein the JPEG image is saved.

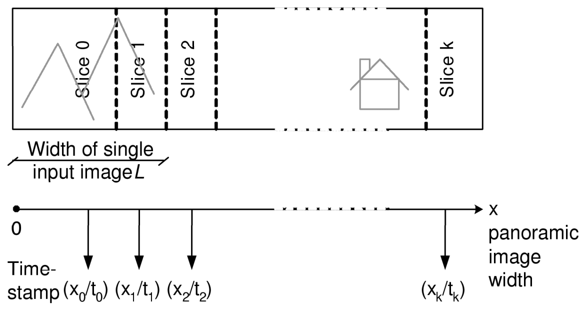

Fig. 3. The first timestamp stored is $(x_0/t_0) = (\frac{L}{2}/0)$, referring to the first image (slice 0) added to the panorama.

Recording e.g. a horizontal panorama using the proposed RTS method defines a time interval, namely from the start of the capturing to the last image frame grabbed; for every slice of an image frame added to the so far created panoramic image the horizontal displacement with respect to the first frame captured is known as well as the temporal distance to the start of the recording.

When the frame rate of the image capturing process of our RTS has been high enough, it is possible for a playback application to autoscroll the image by means of the collected spacing/temporal information at the same speed the user swept the camera over the scenery in such a way that the visible cut-out region of the image matches exactly the camera preview image during the recording, on the basis of the temporal resolution of the human visual sense.

Accordingly, when starting audio recording at the same instant of time when image grabbing is started, a context is established between every horizontal position (when it is about horizontal panoramic images) within the recorded image and the audio.

The method to record the spacing/temporal information works as follows:
As said, RTS actually stitches new portions of the successively grabbed images to the so far panoramic image in realtime. This way the panoramic image created becomes wider the farther the user sweeps the camera over the scenery.
For every new portion the position relatively to the first grabbed image is stored. The same is done with the current time relatively to the instance of time when the user started the recording. This results in an array of - hereinafter called - timestamps (consisting of temporal/spacing information (see fig.2.2)) which are stored together with the image and a reference to the image itself, e.g. as comment segment within an jpeg-file. During the recording of the image, sound is recorded as well and also stored together

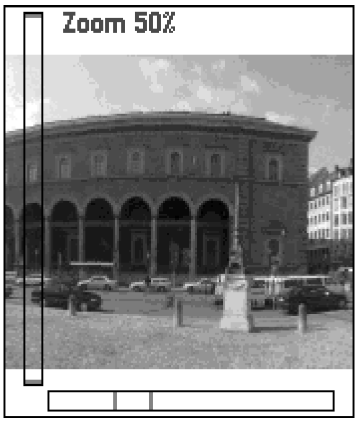

Fig. 4. Screenshot of our application in **playback** mode

with the image (e.g. again as comment segment within an jpeg-file, or as separate, associated audio-file). Thus, after the recording the following has been stored being associated with each other: the panoramic image itself, the timestamps and the audio stream.

3. RENDERING

Rendering such a timestamps- and audio-enhanced panoramic image gives a video-like user-experience when letting the application autoscroll the image. For our playback view which implements audio-synchronous auto-scrolling the rendering works as follows:

1. Zooming to the cut-out region of the panoramic image having the first stored position as centre

2. Start playback of audio stream and concurrently start to automatical scroll the panoramic image according to the timestamps which have been stored during the recording process

This way, during playback the cut-out region of the panoramic image is scrolled at the same speed as the person recording the scenery swept the camera over the scenery, while audio matches to the cut-out region currently shown. A screenshot of our playback view is depicted in fig.4 (there, the brighter lines inside the black rectangles indicate the width or height, respectively, and also the position of the cut-out region currently shown with respect to the whole image - just like scroll bars of a window in a PC application).

Instead of playing the clip from the start to the end (including pause-functionality) another kind of playback is to manually scroll to an image-position of interest, then press a button to play the audio correlated to this image area.

4. COMPLIANCE AND MESSAGING

For mobile network operators having all steps of the panoramic image capturing focused on one single mobile phone means that those should be able to immediately send the gathered images via data services, such as MMS and email. In order to share panoramic image files, they should be readable and displayable by any mobile device (compatibility and universality of our application).

For our suite, we decided that the image itself has a higher priority than the associated audio. Thus, the minimum requirement to our panoramic image file has been put on the compliance to simple JPEG viewers, which are commonly implemented on current mobile phones. This requirement is fulfilled by the usage of the JFIF image file format, where the audio is hidden in the so-called comment segment, such that conventional JPEG viewers will display only the image itself, while our playback mode will render both, the image synchronously scrolled to the audio being played back. As a result, the panoramic image file is backwards-compatible to other viewers but additionally provides all the captured information to players being capable to read and render the audio and timestamp information.

5. SUMMARY AND CONCLUSIONS

The presented methods enable the complete panoramic image generation process from image grabbing to stitching to be ported to a mobile phone, while providing an easy-to-use user interface. The applied RTS was further exploited by recording associated information, namely audio and spacing. When employing the playback method proposed, this gives a video-like user experience.

The compliance issue could be improved with SMIL format [4]. Indeed this standardized format would permit to add audio and timestamp information in a similar way as the one based on JFIF. Furthermore, an algorithm to compensate for light variation during recording could be added to our current system, making it robust in varying light conditions.

6. REFERENCES

[1] Chia-Yen Chen, "Image Stitching - Comparisons and New Techniques," in *CITR-TR-30*, 1998, p. 10.

[2] "ITU-T Recommendation T.81," 1993, p. 32, ITU-T.

[3] Eric Hamilton, "JPEG File Interchange Format," 1992, C-Cube Microsystems.

[4] "SMIL 2.0 Recommendation," 2005, W3C.