CaMus^2 – Optical Flow and Collaboration in Camera Phone Music Performance

Michael Rohs
Deutsche Telekom Laboratories
TU Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
michael.rohs@telekom.de

Georg Essl
Deutsche Telekom Laboratories
TU Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
georg.essl@telekom.de

ABSTRACT

CaMus^2 allows collaborative performance with mobile camera phones. The original CaMus project was extended to support multiple phones performing in the same space and generating MIDI signals to control sound generation and manipulation software or hardware. Through an optical flow technology the system can be used without a reference marker grid. When using a marker grid, the use of dynamic digital zoom extends the range of performance. Semantic information display helps guide the performer visually.

Keywords

Camera phone, mobile phone, music performance, mobile sound generation, sensing-based interaction, collaboration

1. INTRODUCTION

In this paper we describe how the basic CaMus system, which was first presented at last year's NIME conference [8], was extended to allow simultaneous and collaborative performance of multiple phones and reference-free performance through optical flow detection. We call the collaborative version *CaMus*^{2}. Multiple camera phones connect to a Bluetooth enabled personal computer that then maps camera interaction parameters to MIDI format controls, which can be easily mapped to a wide variety of MIDI-enabled software and hardware. In addition CaMus^2 allows the mobile phones to be informed about the semantics of the mapping of interaction parameters to sound generation and modification parameters and can now share, modify, and display this information. On the technical side we have implemented an optical flow based paradigm to untether the performance from marker grids and enabled the use of dynamic digital zoom to increase the range of the height parameter and extended the grid surface area. The user can now choose whether to use the marker grid or optical flow mechanism. The first allows a fixed reference and precise positioning. The latter permits the performer to move freely and use a wider range of expressive gestures. Interaction takes place single-handed, as is in line with current mobile phone interaction styles.

Mobile technology for music performance has been conceptualized and implemented in various ways [1, 4, 5, 9, 10, 11]. See [3] for a recent review of this community. Currently CaMus is the only system that uses the cameras of mobile phones as sensing device for this purpose.

2. OPTICAL MOVEMENT DETECTION

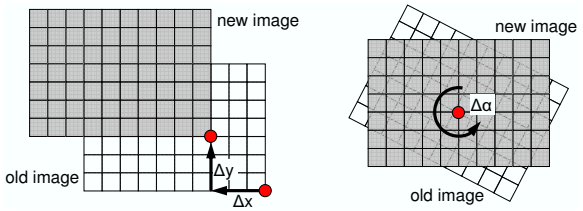

Figure 1: Finding differences in displacements in two successive block images to compute relative movement. Translational (Δx, Δy) and rotational ($\Delta\alpha$) displacements are checked.

The optical movement detection algorithm turns the camera phone into an optical mouse. The algorithm is a refinement of the method described in [6]. It detects relative linear (Δx, Δy) movement of the phone in the display plane and relative rotational ($\Delta\alpha$) movement of the phone around the optical axis, representing three degrees of freedom (see Figure 1). The algorithm operates on the video stream of the camera at a frame rate of 15 fps. Higher frame rates would be desirable and a few devices already provide 30 fps. For movement detection a resolution of 176×144 pixels is sufficient. The algorithm subdivides each frame into a block image, computes cross-correlations between successive pairs of block images for a range of different shift and rotation offsets, and looks for the maximum correlation.

In more detail, the algorithm works as follows: Each frame from the camera is divided into blocks of 8×8 pixels. This down-sampling is required to make the method computationally feasible. With a video resolution of 176×144 pixels the block image has 22×18 blocks. From each block four pixel samples are taken at pixel positions $(x, y), x, y \in \{1, 5\}$. In effect, the whole frame is uniformly sampled with a spac-

NIME07, New York, NY, USA

ing of 4 pixels. In total, $4 \times 22 \times 18 = 1584$ samples are taken in each frame. Since the video stream of the devices we use has a planar YUV 4:2:0 format with 8 bits per pixel, there is no need for gray scaling. The Y components represent the luminance (gray scale) pixel data on which the sampling operates. The average gray value for each block is computed and recentered to 0, i.e. the resulting average gray values are in the range $\{-128, \ldots, 127\}$.

Since two temporally adjacent block images are only $\Delta t = 67$ ms apart (at 15 fps) we assume that there is considerable overlap between them (if movement is not too fast). To compute the spatial displacement between two block images we use cross-correlation as a matching function. It determines which displacement to shift one block image against the other results in the best match. The cross-correlation function (between block images b_1 and b_2) is defined as:

$$r_t(dx, dy) = \frac{\sum_{y=0}^{h-1} \sum_{x=0}^{w-1} b_1(x, y) b_2(x + dx, y + dy)}{(w - |dx|)(h - |dy|)}$$

The denominator normalizes the cross-correlation to the size of the overlapping area. r_t is evaluated in the range $dx, dy \in \{-4, \ldots, 4\}$, i.e. at 81 points. This requires 25276 integer multiplications per frame. The most likely relative linear movement $(\Delta x, \Delta y)$ is the point at which r_t has a maximum:

$$(\Delta x, \Delta y) = \underset{dx, dy \in \{-4, \ldots, 4\}}{\operatorname{argmax}} r_t(dx, dy)$$

If the displacement was scaled by the magnification of the camera view, the real velocity could be computed. However, since the scaling factor depends on the unknown camera parameters and the variable distance of the camera to the background, no scaling is performed. This means that the computed relative movement depends on the distance of the camera to the background. The obtained movement parameters are still useful for interaction.

Relative rotation $\Delta\alpha$ is computed in a similar fashion by rotating the block images against each other. The current block image is rotated by α values between $-24°$ and $24°$, with a step width of $6°$. The rotational coordinate mappings are precomputed and stored in tables for performance reasons. The rotated block image is cross-correlated with the previous block image and the angle $\Delta\alpha$ with maximum correlation is chosen as the most likely amount of rotation.

$$r_r(\alpha) = \frac{\sum_{y=0}^{h-1} \sum_{x=0}^{w-1} b_1(x, y) b_2(\text{rotate}(\alpha, x, y))}{\text{number of overlapping blocks}}$$

In order to suppress spurious movements and shakes, movement is only reported if the signal-to-noise ratio – the maximum relative to the mean correlation – is above a threshold.

The algorithm works quite reliably and detects the relative motion even if the sampled backgrounds only have a limited number of features, like a wall or a floor. Because only a few pixels are sampled, the algorithm performs quickly and leaves enough time for rendering the workspace. On a Nokia 6630, it runs at the full frame rate of 15 $(\Delta x, \Delta y, \Delta\alpha)$ triples per second. When rendering the workspace (with 5 targets in the active layer) and sending updates via Bluetooth the average display update rate is still 14.6 updates per second.

Since movement detection is relative, drift is unavoidable. Particularly if the user makes fast movements the overlap between successive images is not sufficient and relative movement cannot be computed. If the user moves the phone in one direction and then reverses movement back to the starting position, the final workspace position is not identical to the original one. The right selection key of the phone is used as a clutch, which fixes the workspace on the screen and allows the user to reposition his arm. This mechanism is similar to lifting the mouse from the table.

3. GRID TRACKING WITH EXTENDED VERTICAL RANGE

As an alternative option to optical flow, camera phones are tracked over a grid of visual markers. The grid provides a fixed frame of reference for the virtual workspace within which the user interacts. Grid tracking does not suffer from the drift problem of optical flow and can precisely sense very subtle movements. However, the grid has to be present in the camera view, which limits user mobility. The absolute position and orientation of the device within the physical space above the grid is tracked with low latency and high precision. The graphics is rendered perspectively, as if looking through the device onto the background surface.

The approach discussed here is an extension to the one described in [8]. The markers have been extended to a capacity of 16 bits: 2×7 bits for index positions and 2 parity bits. The maximum grid size is thus 128×128 markers or 1024×1024 ccu. Suitable printing sizes are 1.5 mm to 2.0 mm per black-and-white cell, which yields a maximum grid area of 1.54 m to 2.05 m.

In the original implementation, the vertical tracking range (the distance of the camera lens to the grid) was limited to between 2 and 10 cm. This proved insufficient for effective interactions along the z-dimension. In the current extension we use the digital zoom feature that is present in many camera phones to substantially extend the vertical tracking range. Digital zoom increases the apparent focal length at which an image was taken by cropping an area at the image center with the same aspect ratio as the original image. The cropped area is rescaled to the original dimensions by interpolation. No optical resolution is gained in this process, but digital zoom is done by the camera before any compression and does not have to be done by the main processor of the device, it essentially gives high-quality rescaling for free.

The Symbian camera API allows to set the digital zoom level between 0 and some device-dependent maximum value. In an experiment we kept the distance to an object in the camera view constant, continuously changed the digital zoom level, and measured the size at which the object appeared in the camera view ($size_{zoomed}$). We found that $\text{size}_{\text{zoomed}} = \text{size}_{\text{unzoomed}} \times e^{k \times \text{level}}$, or equivalently $\text{distance}_{\text{zoomed}} = \text{distance}_{\text{unzoomed}} \times e^{-k \times \text{level}}$. For Nokia the 6630 (6× digital zoom) $k = 0.0347$ ($R^2 = 0.9983$), for the Nokia N70 (20× digital zoom) $k = 0.0386$ ($R^2 = 0.9992$), and for the Nokia N80 (20× digital zoom) $k = 0.0345$ ($R^2 = 0.9974$).

During grid tracking, digital zoom is continuously adjusted, such that markers appear at a size that is best suited for detection. If no markers are detected in a camera frame, different zoom levels are tried. The algorithm is complicated by the fact that changes to the zoom level via the camera API do not result in immediate changes in the next camera frame. Instead, the new digital zoom setting becomes valid only 2 to 5 frames after the adjustment is made. Therefore, the algorithm chooses the setting that is most likely to yield smooth distance changes.

With this method the vertical recognition range for a grid with a cell size of 1.5 mm is increased from 10 cm to 30-50 cm, depending on the device. Markers are recognized in view finder mode with a frame size of 176×144 pixels at a rate of 10-15 frames per second, depending on the complexity of the rendered scene. At larger distances the perspective mapping of the grid gets more unstable, which could be mitigated by filtering and smoothing the perspective mapping. Overall, visual grid tracking provides a 3-D physical interaction space of 150×150×30 cm, in which multiple devices can be tracked with low latency and high precision.

4. COLLABORATIVE PERFORMANCE

CaMus[2] allows multiple camera phones to connect to a Bluetooth enabled personal computer that then maps camera interaction parameters to MIDI format controls, which can be easily mapped to a wide variety of MIDI-enabled software and hardware. In addition CaMus[2] allows the mobile phones to be informed about the semantics of the mapping of interaction parameters to sound generation and modification parameters and can now share, modify, and display this information.

Bluetooth (www.bluetooth.org/spec) allows multiple simultaneous connections to an RFCOMM server. In order to use this to make CaMus collaborative, we extended the receiver software on the personal computer from the original architecture [8] to allow multiple serial Bluetooth connections to be used. The computer opens and registers the serial devices as needed upon startup in the local SDP database. The mobile camera phones then get to select which of the found serial channels are to be used for communication of this particular phone with the computer. Each phone retains the channel as identifier for itself within the network to communicate settings with the personal computer and other devices. All connections currently go through the PC (see Figure 2) but direct communication between mobile camera phones through the same serial Bluetooth protocol is planned.

The interaction parameters received from the various camera phones via the serial Bluetooth channels are then in turn converted into MIDI channels and can be used by any MIDI-enabled software and hardware. Parameters can be mapped to the same or separate sounds or effects and hence performers can jointly manipulate one sound or contribute different sounds or effects through the same architecture.

The communication protocol currently supports an array of commands that typically are sent from the camera phone to the host PC, which may or may not send a response, depending on the opcode (see Table 1). In principle the protocol is however non-hierarchical and is set up to support direct phone-to-phone communication.

5. VISUALIZATION OF SEMANTICS

In order to improve the visualization on the camera phone display we introduce semantic information and a communication protocol to share this information between participants in the Bluetooth network. The semantic information carries the mapping of interaction parameters from the target detection system to sounding sources and sound manipulation parameters.

The visual targets of the original CaMus system now will display a textual description of the function of the target.

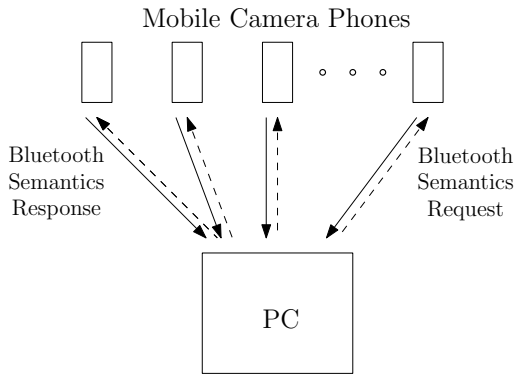

Figure 2: Bluetooth wireless network between camera phones running the optical tracking and workspace visualization software and the PC running MIDI based sound synthesis software.

For example a target may represent a low-pass filter (see Figure 3). In this case the relevant semantic information can be presented inside the target square by the visualization.

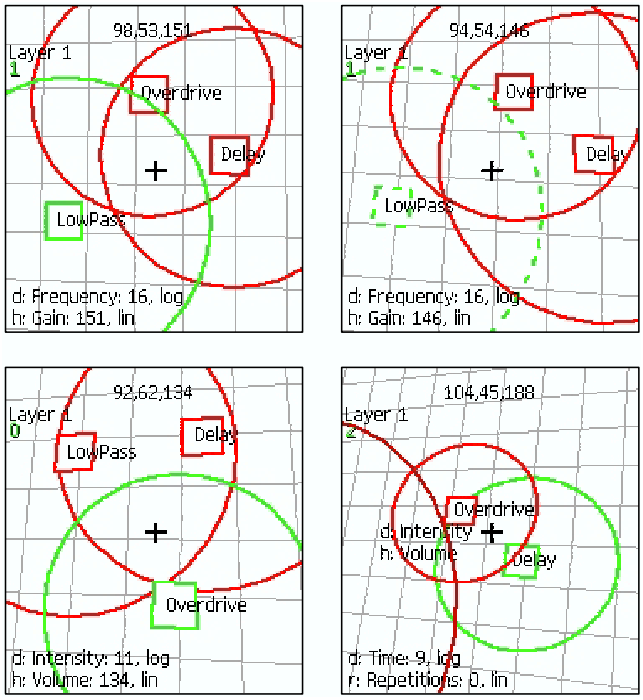

Figure 3: Three different filters with their semantics are located at different places in the workspace.

In addition for each target, semantic information of the control parameters related to the target can be exchanged. These control parameters [8] are x and y distance from the target, Euclidean distance d from the target, height h over the target sheet, rotation angle α, and tilt angles θ_x and θ_y relative to the x- and y-axis, respectively. Each of these can be given a semantic name. For example distance d can be named "Volume." Furthermore, the raw input data can be mapped to a new value range. For example height may be between 40 and 160 but the control parameters used by the

Opcode	Description
CONNECT	Inform a host about a connect with initial information about the phone
SEND_MOVEMENT	Send movement information (x-y position, height, rotation, tilt)
SEND_PARAMETERS	Send auxiliary parameters (parameters relative to targets, x-y distance, height, rotation, tilt)
DELETE_TARGET	A target has been deleted
REQUEST_SEMANTICS	Request semantic information from other participants in the network
SEND_SEMANTICS	Send semantic information about targets (name, range and mapping of parameters)
NO_SEMANTICS	No semantic information is available for this participant

Table 1: CaMus communication protocol command opcodes.

MIDI software are 0-127. Through the exchange the mobile phone is informed what the mapping ranges are, and if the mapping is linear or logarithmic.

This information can be requested for any participant from the network. Currently the personal computer serves to host and provide this information, but the protocol in use is designed to allow exchange and modification of this data by all participants in the network.

6. CONCLUSIONS

In this paper we presented the extension of the CaMus system for performance of music with mobile camera phones for multiple users and untethered from the requirement of a fixed marker grid through an optical flow detection mechanism. Multiple camera phones communicate with a personal computer through a Bluetooth network sending performance parameters derived from a visual tracking system. This information is then mapped to MIDI to connect to arbitrary MIDI-enabled sound software or hardware.

In order to enhance the contextual information for the performer, we have added semantic information that can be shared by all participants in the network and displayed on the mobile camera phone's visualization. Different aspects of the performance can be placed on multiple separate layers. This way performers can separate sounding functions from effect functions and can choose to share a common performance space or separate it as needed for a given performance context.

Technologically optical flow additionally frees the movement space, but at the cost of a fixed reference and the potential for some drift. We have also extended the range of the height allowable by the system through the use of dynamic digital zoom to improve the possible performance. This is in part result of an interface study which showed that the confinement of the vertical movement was a limiting factor [7].

As immediate future work we plan to implement the sound synthesis and manipulation engine completely on the mobile device itself and hence remove the need for a personal computer to serve as the sound source. This has already been prepared by the portation of the sound synthesis toolkit STK to Symbian [2]. As the mobile device itself defines the mapping semantics of interaction parameters to sounding results, we plan to implement editing of this information on the phone itself. At the same time, the communication will be extended to allow direct exchanges between multiple mobile devices in the absence of a personal computer. Through this network semantics information will be sharable among all users of the CaMus^2 system. This allows each phone to visualize the performance context of all participants interactively and while this context is edited on the fly.

7. ACKNOWLEDGMENTS

We are grateful for the input of numerous NIME'06 participants on the original CaMus project, specifically for the suggestion to implement an optical flow version of the system.

8. REFERENCES

[1] W. Carter and L. S. Liu. Location33: A mobile musical. In *NIME '05: Proceedings of the 2005 Conference on New Interfaces for Musical Expression*, pages 176–179, May 2005.

[2] G. Essl and M. Rohs. Mobile STK for Symbian OS. In *Proc. International Computer Music Conference*, pages 278–281, New Orleans, Nov. 2006.

[3] L. Gaye, L. E. Holmquist, F. Behrendt, and A. Tanaka. Mobile music technology: Report on an emerging community. In *NIME '06: Proceedings of the 2006 conference on New Interfaces for Musical Expression*, pages 22–25, June 2006.

[4] M. Kaltenbrunner. Interactive Music for Mobile Digital Music Players. In *Inspirational Idea for the Intl. Computer Music Conference (ICMC)*, Sept. 2005.

[5] G. Levin. Dialtones - a telesymphony. `www.flong.com/telesymphony`, Sept. 2, 2001. Retrieved on April 1, 2007.

[6] M. Rohs. Real-world interaction with camera phones. In *Second International Symposium on Ubiquitous Computing Systems (UCS 2004)*, pages 74–89. LNCS 3598, Springer, July 2005.

[7] M. Rohs and G. Essl. Which one is better? – information navigation techniques for spatially aware handheld displays. In *ICMI '06: Proceedings of the 8th International Conference on Multimodal Interfaces*, pages 100–107, Nov. 2006.

[8] M. Rohs, G. Essl, and M. Roth. CaMus: Live music performance using camera phones and visual grid tracking. In *Proceedings of the 6th International Conference on New Instruments for Musical Expression (NIME)*, pages 31–36, June 2006.

[9] G. Schiemer and M. Havryliv. Pocket Gamelan: Tuneable trajectories for flying sources in Mandala 3 and Mandala 4. In *NIME '06: Proceedings of the 2006 conference on New Interfaces for Musical Expression*, pages 37–42, June 2006.

[10] A. Tanaka. Mobile music making. In *NIME '04: Proceedings of the 2004 conference on New Interfaces for Musical Expression*, pages 154–156, June 2004.

[11] A. Tanaka and P. Gemeinboeck. A framework for spatial interaction in locative media. In *NIME '06: Proceedings of the 2006 conference on New Interfaces for Musical Expression*, pages 26–30, June 2006.