Motion Detection as Interaction Technique for Games & Applications on Mobile Devices

Stephan A. Drab
Upper Austria University of Applied Sciences
Mobile Computing
Hauptstrasse 117
Hagenberg, Austria
stephan.drab@fh-hagenberg.at

Nicole M. Artner
Upper Austria University of Applied Sciences
Media Technology and Design
Hauptstrasse 117
Hagenberg, Austria
nicole.artner@fh-hagenberg.at

Keywords
mobile devices, pervasive, interaction techniques, motion detection

ABSTRACT
Mobile devices become smaller and more powerful with each generation distributed. Because of the tiny enclosures the interaction with such devices offers limited input capabilities. In contrast there are hardly any mobile phones purchasable that do not have a built-in camera. We developed a concept of an intuitive interaction technique using optical inertial tracking on mobile phones. The key of this concept is the user moving the mobile device which results in a moving video stream of the camera. The direction of the movement can be calculated with a suitable algorithm. This paper outlines the algorithm *Projection Shift Analysis* developed to run on mobile phones.

1. MOTIVATION
Our approach detects the relative motion of the camera in the two-dimensional pixel space of the image. As camera movement directly results in motion of all scene components the motion of the camera movement can be defined as the inverse movement of the scene. If there are no significant scene components moving for itself conventional motion detection methods can be used to analyse the video stream. There are several algorithms from different fields of computer graphics and image processing used to parameterise the motion of the scene. Although they pursue different approaches, all of them would analyse the scene motion sufficiently. Due to low CPU and memory resources on mobile phones we developed the *Projection Shift Analysis* algorithm for motion analysis. There is a wide range of applications for the motion parameter, for example controlling a game similar to joystick interaction. Another possible application could interpret motion gestures or control the cursor like the stylus input technique on PDAs.

2. RELATED WORK
In [8] Geiger et al. present concepts for mobile games and address interaction issues concerning mobile devices.

The simplest approach to track objects is detecting significant features in an image, e.g. edges. Naturally, edge detection methods like the Robert, Prewitt, Sobel, Laplace or Canny [2] filters are used to achieve this.

Motion Detection in 3D-Computer Graphics, Mixed- and Augmented Reality is often referred to as *Tracking*. Beier et al. presented a markerless tracking system using edge detection in [1]. Comport et al. propose a robust markerless tracking system in [3]. A specialised solution to the problem of markerless tracking was published by Simon et al. in [15]. In [11] Moehring et al. present a marker-based tracking system designed especially for mobile phones. Kato and Billinghurst developed the optical marker based tracking system ARToolKit published in [9].

Foxlin et al. present a wide spectrum of optical inertial tracking systems in [5, 6, 17]. Additionally, a taxonomy of Motion Detection methods has been published in [4]. In [10] Koller presents a method to track the position of cars using an optical system.

Siemens Mobile developed a game called *Mozzies*, that is distributed with the mobile phone SX1 by default. This Symbian based game augments the background video from the camera with moths. The user can point the gun at a moth and shoot it by moving the phone and pressing the appropriate button. In [7] Geiger et al. present an interesting approach of an augmented reality version of a soccer game running on a PDA.

3. INTERACTION TECHNIQUES
This chapter describes and classifies various interaction techniques used on mobile devices nowadays. There are a few main parameters that define the usability of those techniques. The **reaction time** between the user input and the response on output devices such as the display is a very crucial parameter. Any visual response on the output device after about 200 ms is not interpreted as a direct reaction to the user, but as a separate event. The **quantity** of actions a user is able to perform using a specific input technique defines the speed at which he can interact with the device. The **intuitivity** of an input method strongly affects the usability

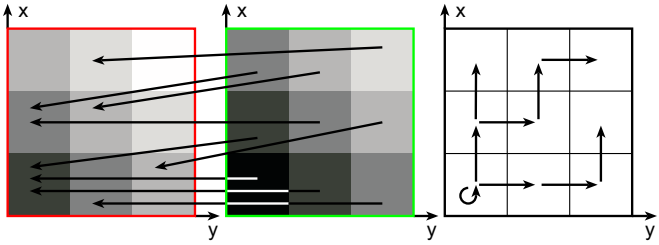

Figure 1: An example of the Block Matching algorithm.

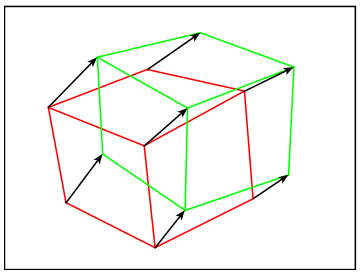

Figure 2: Edge Detection and Tracking sample.

and user acceptance.

The following interaction techniques for mobile phones were evaluated regarding the above parameters. **Keys** are the most common input technique for mobile phones. They offer a very short reaction time but lack intuitivity. The rate of interaction is dependent on the user's experience with this technique. **Voice recognition** is a very intuitive input technique, but lacks fast reaction time and input quantity. **Touch screens** are the most intuitive way of interacting with a screen based mobile device. The reaction time is comparable with key input but the input quantity strongly depends on the GUI design. Unfortunately aside from smartphones there are no mobile phones featuring a touch screen.

Each of these techniques has mentionable advantages. Key interaction offers a good reaction time. Voice recognition is very intuitive to use. Touch Screens combine a fair amount of intuitivity and interaction quantity, but aside from smartphones they are not available on mobile phones. Thus we developed a concept and implementation for an interaction technique particularly for mobile phones that incorporates the advantages of all presented techniques.

4. MOTION DETECTION ALGORITHMS

The term *Motion Detection* describes a set of algorithms that detect the motion in a successive image sequence, e.g. in a video captured by a camera. In this section we present suitable motion detection algorithms:

4.1 Block matching

Block matching is a method from the video compression sector. It divides the images in equally sized blocks to find the best matching block in a reference picture for all blocks of the current image. It yields to describe an image not using color values per pixel, but as block references to the previous image along with other parameters, as Richardson describes in [13].

Even tough the algorithm has not been developed to analyse motion in an image sequence, it can be extended to fit this purpose. The algorithm determines where a block of an image will be positioned in the next image. The resulting block references can be interpreted as motion vectors in units of blocks. An arithmetic middle of all block motion vectors multiplied by the pixel size of a block would lead to a single motion vector describing the approximate relative motion.

In figure 1 a sample of a matching procedure is depicted. The frame on the left shows the reference image. In the middle the current image is depicted which is encoded using block references to the previous image. The arrows in between indicate the block mapping information calculated by the algorithm. In the right frame the block references are vizualised as motion vectors. The arithmetic middle of the motion information from the right frame in figure 1 would result in a vector of $(\frac{8}{9}, \frac{8}{9})$ which almost matches the relative movement of $(1, 1)$ in our example.

4.2 Edge Detection and Tracking

Edge Detection and Tracking is used throughout markerless *3D-Tracking* algorithms[1]. In the first step the edges are extracted from the image. This is possible through either folding the image with an appropriate *Edge Detection* matrix [2, 14] or using the *Hough Transformation* [16]. The detected edges can then be tracked throughout the image sequence. Figure 2 shows an example of detected edges of a cube that have been tracked in two images.

The edge comparison step in the Edge Detection and Tracking algorithm is applied to an unpredictable high number of edges. Thus the processing time used for this operation cannot be forecast and can become a bottleneck in case the images depict a huge number of edges. Although Edge Detection and Tracking is the most promising algorithm, it is not capable of running on devices with low computing power at a reasonable frame rate.

4.3 Analysis of Scene components

Moving scene components in a video stream result in a partial motion of a scene. In [10] Koller describes an algorithm to extract motion information of scene components from an image sequence. The first step in this approach extracts the objects' parameters from the image. Thereafter the proximate image is searched for objects and relative motion in pixel space is calculated. An example is depicted in figure 3. Again, an arithmetic middle of the motion vectors of all scene components could be used to determine the scene and camera motion.

The analysis of the motion of scene components is similar to *Edge Detection and Tracking* and incorporates the same disadvantages like unpredictable computing time and consuming high computing power. Therefore, it cannot be used on mobile devices.

[1]See [3, 15] for examples of 3D Tracking algorithms.

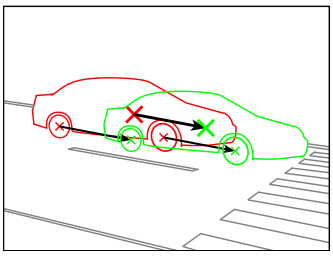

Figure 3: Scene Component Analysis sample.

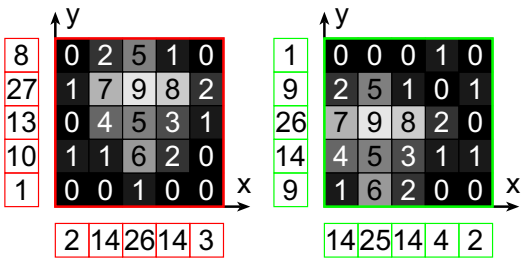

Figure 4: Horizontal and Vertical projection buffers for each of the two successive images.

5. PROJECTION SHIFT ANALYSIS

The previously presented motion detection algorithms qualify for detecting motion information in an image sequence using a high amount of calculating time. However, in mobile applications computing power is a rare resource, therefore we propose a new method called *Projection Shift Analysis*. The algorithm does not require color information, but grayscale images as input. Fortunately, most cameras support captured images in the YUV format[2] which holds a separate luminance (Y) channel containing grayscale information. This algorithm will discard the provided color information (U and V).

5.1 Image Projection

In the first step the image is projected onto its x- and y-axis. That means all luminance values of each row is summed up in the vertical and each column in the horizontal projection buffer as depicted in figure 4. The horizontal and vertical projection buffers $pbh(x)$ and $pbv(y)$ of the image $im(x, y)$ with the corresponding width w and height h are defined as $pbh(x) = \sum_{i=0}^{h-1} im(x, i)$ and $pbv(y) = \sum_{i=0}^{w-1} im(i, y)$ with $x \in [0..w[$ and $y \in [0..h[$.

5.2 Shift Analysis

If the scene within an image sequence is moved vertically or horizontally the corresponding projection buffers will shift equally. For example an image sequence containing a horizontal panning shot from the left to the right will result in a horizontal projection buffer whose values will shift to the left over time. Thus our approach searches for the best matching shift between two projection buffers of two successive images. Figure 5 shows three examples of different shift values.

To estimate the best match we introduce a value called *badness factor* that is similar to a correlation factor and characterises the badness of a shift. A lower value indicates a higher probability for this shift to be correct. The algorithm calculates the badness factor for every possible shift through summing up all squared differences of the values of the compared projection buffers. Because the calculated badness factors result from a different number of compared values they are normalised by dividing it by the number of values compared. Figure 5 shows the calculation of badness factors using the vertical projection buffers of the image displayed in figure 4.

[2]For more information on the YUV format we refer to [12].

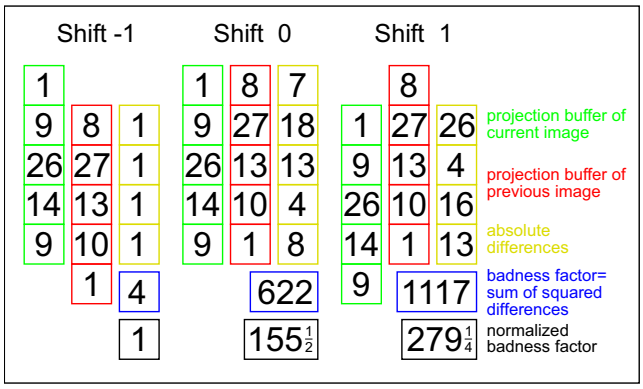

Figure 5: Example of badness factor calculation for different shifts of two projection buffers.

The shift value used to calculate the smallest normalised badness factor is assumed to be the correct relative motion between the two compared images. In our example depicted in figure 4 the best matching shift value of -1 exactly matches the vertical movement of 1 pixel. This procedure can be applied to horizontal and vertical projection buffers equally to estimate the motion in x- and y-axis.

6. RESULTS

Tests determining the robustness of the algorithm *Projection Shift Analysis* elicited the following restrictions:

If there are *significant scene components moving* in the captured video stream the motion detection often "follows" these objects instead of detecting the motion of the background. *Repeating patterns* – for example a chess board – often lead to jumping motion detection results. The color values of the analysed images need to have a *wide dynamic range*. Most cameras of mobile phones use a relatively *long shutter time* due to low quality lenses and CCD chips. If the camera moves too fast this can result in blurred images which cannot be analysed correctly. The quality of the detected motion parameters decreases if a *lower frame rate* is used. The camera used to capture the video stream must support a *sufficient resolution and image quality*. The algorithm does not support detection of *rotations* around the camera's view axis. However, unusable motion detection results are only produced in some cases.

To further examine the robustness two application prototypes were developed to evaluate the functionality and usability of the proposed concept:

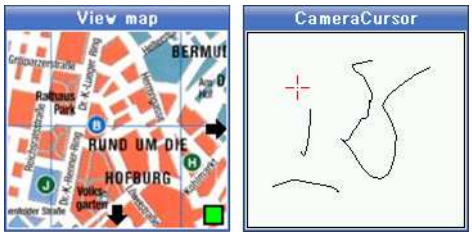

Figure 6: The prototypes MAPnavigator and CameraCursor.

MAPnavigator is an application that allows the user to view different maps. The problem in this context is the low display resolution of mobile phones. Consequently it is not recommended to scale down a huge map because then the user is not able to make out important details in the map. The solution to this problem is to divide the map into several parts. MAPnavigator uses the *Projection Shift Analysis* motion detection algorithm to seamlessly navigate through the map. It is very easy and intuitive to find the requested part of the map using the motion detection technique. The left part of figure 6 shows a screenshot of the application.

The application **CameraCursor** was developed to test if our motion detection algorithm is capable of controlling a cursor. The user is able to control the cross cursor by moving around the mobile phone. By pressing and holding the appropriate button the cursor draws lines while moving on the screen. Additionally the speed of the cursor can be configured at run time. An example of the application is shown in the right part of figure 6.

7. CONCLUSIONS

The *Projection Shift Analysis* algorithm works if none of the given cases in section 6 occur. Compared to the algorithms in section 4 it uses very little CPU and memory resources. Therefore it is ideal to be used on mobile phones. In case the camera is moved quickly the results may be inaccurate by a few pixels. Using the motion parameters as user interaction does not require high exactness for fast movements because the user does not have an indication of inaccuracy as long as the movement displayed on the screen is approximately following his motion. However, there are some cases the algorithm cannot keep track of the motion and fails completely. This mostly happens when the user quickly moves the camera while pointing at dark spots or repeating patterns. Future implementations are planned to filter out motion noise using the *Kalman Filter* which would also reduce jumping motion parameters caused by unfavorable environments.

8. REFERENCES

[1] D. Beier, R. Billert, B. Brderlin, D. Stichling, and B. Kleinjohann. Marker-less vision based tracking for mobile augmented reality. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 258–259. IEEE, Inc., 2003.

[2] J. F. Canny. Finding edges and lines in images. Master's thesis, MIT, 1983.

[3] Andrew I. Comport, Eric Marchand, and Francois Chaumette. A real-time tracker for markerless augmented reality. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 36–45. IEEE, Inc., 2003.

[4] Eric Foxlin. *Handbook of Virtual Environment Technology*, chapter 8, Motion Tracking Requirements and Technologies. 2002.

[5] Eric Foxlin, Yury Altshuler, Leonid Naimark, and Mike Harrington. Flighttracker: A novel optical/inertial tracker for cockpit enhanced vision. In *IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR 2004)*, 2004.

[6] Eric Foxlin and Leonid Naimark. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Joint International Immersive Projection Technologies (IPT)/Eurographics Workshop on Virtual Environments (EGVE) 2003 Workshop*, 2002.

[7] C. Geiger, B. Kleinnjohann, C. Reimann, and D. Stichling. Mobile ar4all. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR)*, 2001.

[8] Christian Geiger, Volker Paelke, and Christian Reimann. Mobile entertainment computing. In *TIDSE*, pages 142–147, 2004.

[9] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *proceedings of IEEE International Workshop on Augmented Reality*, pages 125–133, 1999.

[10] Dieter Koller. *Detektion, Verfolgung und Klassifikation bewegter Objekte in monokularen Bildfolgen am Beispiel von Strassenverkehrsszenen*. Infix, 1992.

[11] M. Moehring, C. Lessig, and O. Bimber. Optical tracking and video see-through ar on consumer cell phones. In *proceedings of Workshop on Virtual and Augmented Reality of the GI-Fachgruppe AR/VR*, pages 193–204, 2004.

[12] Wikipedia, the free encyclopedia. YUV. http://en.wikipedia.org/wiki/YUV.

[13] Iain E. G. Richardson. *Video Codec Design: Developing Image and Video Compression Systems*. Wiley, June 2002.

[14] J. Shen and S. Castan. An optimal linear operator for step edge detection. *Computer Vision, Graphics and Image Processing*, (54), 1992.

[15] Gilles Simon, Andrew W. Fitzgibbon, and Andrew Zisserman. Markerless tracking using planar structures in the scene. In Proc. of ISAR.

[16] Sargur N. Srihari. Analysis of textual images using the hough transformation. Technical report, State University of New York at Buffalo, 1988.

[17] Dean Wormell and Eric Foxlin. Vis-tracker: A wearable vision-inertial self-tracker. In *Joint International Immersive Projection Technologies (IPT)/Eurographics Workshop on Virtual Environments (EGVE) 2003 Workshop*, 2003.