A Virtual Dance Floor Game using Computer Vision

Daniel Brehme, Fabian Graf, Frederik Jochum, Ioannis Mihailidis, Gregory Orchard, Detlev Droege, Dietrich Paulus

Computational Visualistics, University of Koblenz-Landau, Germany, `agas@uni-koblenz.de`

Abstract

Keyboard, mouse, and game controllers are common input devices for most computer games. Alternative devices are rarely used, restricting the player's possible interaction with the virtual world. Computer vision techniques permit interaction with the computer that grants far more freedom than traditional devices, without the need of game-specific hardware and hindering cables. In this paper we describe a dance game where interaction is solely based on image analysis. A tracking system detects the position of the player's feet in real-time. Two cameras are used and the foot positions are estimated in 3d from the two views. The player currently must have a marker attached to each shoe. Our evaluations show that depending on the markers, feet can be tracked accurately in real-time using a two-camera setup and an ordinary laptop computer. In further work we will extend the system to marker-less tracking. The system has been installed in a public museum and has been left in unsupervised mode for several months.

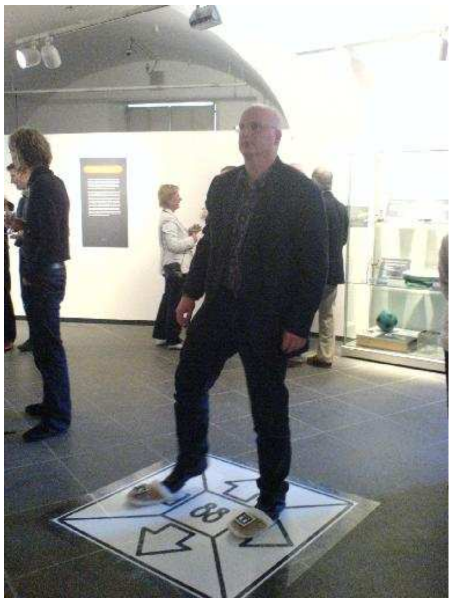


Figure 1: Visitor using our system

1 Introduction

Human-machine interaction for playing computer games usually requires a keyboard, a mouse or a joystick. Alternatively, computer vision can be used to derive the input to the game control module. We will demonstrate how real-time computer vision can be used in an interactive dancing game.

The game system Cam^{2}Dance was developed for an exhibition in Koblenz, Germany,[1] to demonstrate alternative human-machine interaction using computer vision. The system is a prototype that will be developed further by other computer vision students. Figure 1 shows a visitor using the system.

The game system consists of two major components: a vision-based tracking system and the game itself. To ensure minimal costs and sufficient documentation only open source and free software were used. Linux was chosen as operating system and C++ as programming language. Key technologies used are ARToolKit for tracking and SDL with OpenGL for the game. In the following we will briefly examine the specifics of each of these components.

The complete game requires computer vision for interaction, visualization to direct the user (Figure 5,Figure 6, see below), synchronous music with visual commands, and a game control unit. This contribution mainly deals with the computer vision aspect that requires real-time vision, accurate tracking with 30 Hz and restricted 3D estimates from two views. The system will be on display for several months in a museum; therefore a high degree of robustness under varying conditions is crucial. Although the ultimate goal is to track without markers, we chose to test a marker-based system due to the robustness constraints.

The article is organized as follows: In Sect. 2 we outline different tracking techniques with and without markers. Sect. 3 describes our own approach that uses a calibrated stereo setup. In Sect 4 we evaluate different experimental settings. In Sect. 5 we conclude and outline future work.

2 Computer Vision for Computer Games

Unconventional input devices are presently only found with video game consoles, usually simply activating the standard controller buttons with new hardware, e.g. dance pads. Lately vision-based games are on the market, like *Eye Toy* from Sony. Through the freedom of movement completely new approaches to video games are possible.

[1] `http://www.landesmuseumkoblenz.de`

Figure 2: Dance Dance Revolution: 1. Arcade game 2. dance pad (source: from the internet)

Currently, two popular vision-based games are on the market: *eye toy* and *virtual air guitar*.[2] Scientific publications about these systems are not known to the authors. Related scientific work can be found in face, hand, and gesture recognition, e.g. in [4] or in the context of computer use for handicapped people, as in [6].

Interactive music dance games are also popular. They require that the player moves according to arrows appearing on the screen that correspond to the beat of a chosen song. At this point, the player acts according to dancing step patterns shown on a screen. Simultaneously played music coincides with these steps. A dance pad (Figure 2) detects the moves into the four possible directions, similar to the cursor keys on a keyboard. Dance Dance Revolution is the best-known among those games and an open source version is available as StepMania for Windows, Mac, and Linux.

Foot-fall detection as an interface for locomotion in games and virtual environments has been proposed in [1].

Marker-based tracking has been used especially for motion capturing in computer graphics applications, as it allows for stable tracking of many points even with multiple cameras. Markers are also used in medical applications, as they provide accurate measurements, e.g. in [3, 5]. So-called *point fiducials* provide a match from a scene point to image point, whereas *planar fiducials* provide 6 constraints for pose and orientation [7, p. 32]. A well-known open source software system that uses such markers is called ARToolKit; similar ideas are available in the systems ARStudio and ARTag [2, p. 8].

3 Cam^{2}Dance

We created a computer game that has four major components: computer vision, music, visualization, and central game logic module. The sequence of possible actions and process communication is shown in Figure 3. The setup of the

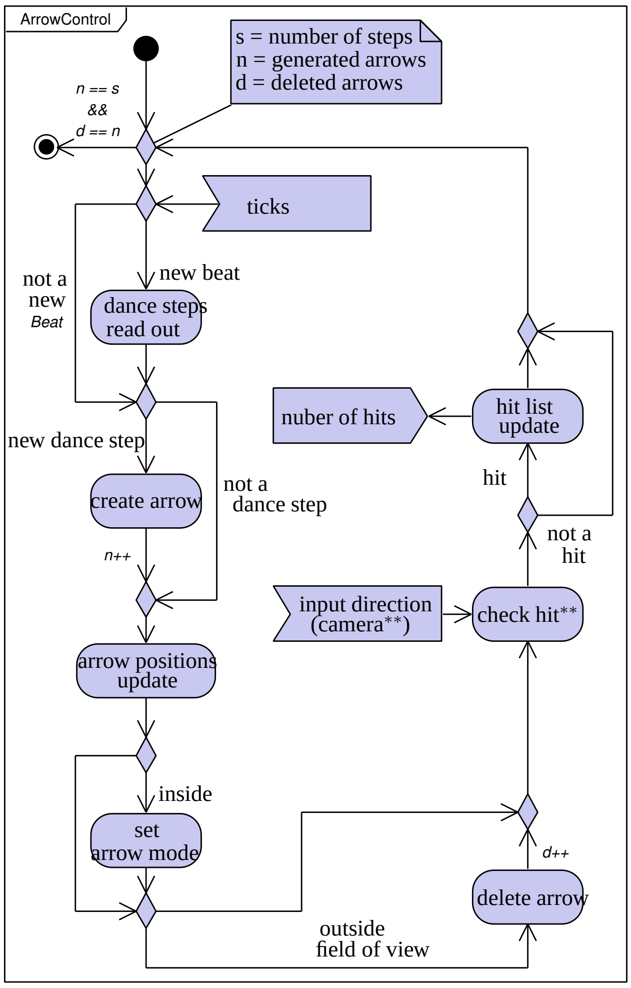


Figure 3: Game logic. Modules described in this article are marked by **

compontents is shown in Figure 4. The user dances on a pad; the commands are shown on a screen using rear projection; user actions are tracked by a stereo camera system; the central control unit decides for the next action and summarizes game status and user scores. In the following we focus on the vision module.

3.1 Overview

Although the game was designed to function optimally with the tracking, it nevertheless can also operate without, only using a standard keyboard as an input device. This is also true for the tracking system that could theoretically control any other game.

[2]`http://airguitar.tml.hut.fi`

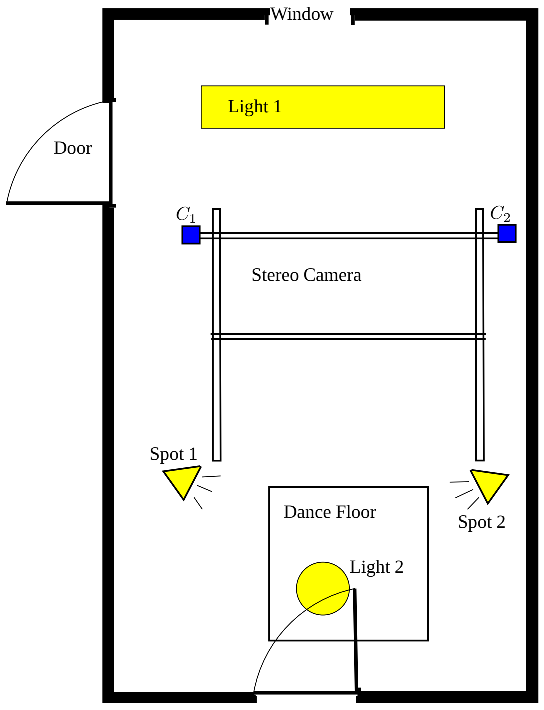


Figure 4: Setup of the system

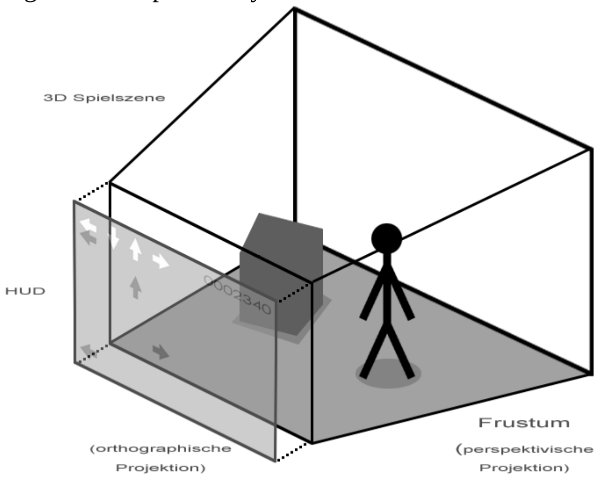


Figure 5: Graphics setup

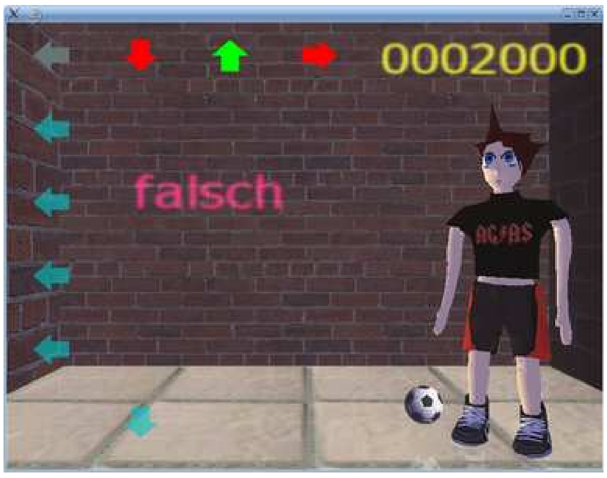


Figure 6: Visual commands and feedback

The computer game is a simplified version of current dance games, since it must be easily understood and operated by visitors of the exhibition of any age. The objective of the game is identical, just all selectable options were removed, e.g. music choice. Instead it runs through a continuous loop with random song selection.

The central elements of the game are computer graphics, music and game logic. A simple framework was developed, using the SDL multimedia library as basis, to allow quick changes to the system. Specific XML files allow a configuration of the game, including textures, screen resolution, and music files with corresponding step files (i.e. dance moves). Sound and synchronous step commands are created from a separate program that analyzes music to detect beats and speed. We decided to use 3D graphics including a dancing character that shows the correct moves and function as a dance teacher. OpenGL was used as graphic language and most models were created with Milkshape, an inexpensive modelling tool; Figure 7 shows screenshots of our game. New scenarios are now designed with Maja.

The background and the dancer are displayed in 3D, whereas the arrows (step commands) and the feedback lie in a 2D overlay (HUD head up display). The graphical setup is shown in Figure 5.

The player dances on a mat with printed directions for his orientation (not necessary for the tracking). The expected dance moves are displayed onto a screen using rear projection. All user actions are tracked by the stereo camera system and translated into commands. The central control unit decides for the next action and summarizes game status and user scores.

The arrows for visual feedback are shown in Figure 6. Several tests have been made how to indicate errors. The option shown in Figure 6 was to display a text (*falsch* — „wrong”). This was lated discarded; instead, the arrows on the top shown in Figure 7 are highlighted when a correct step has been recognized.

Our game logic is the core of the system, controlling all processes. Its main task is to verify the players input via the rules of the game. The sequence of possible actions and process communication with respect to the computer vision module is shown in Figure 8.

3.2 Technical Structure

Two images are captured simultaneously; feet are located using markers whose position is estimated in 3D; the foot positions are then classified as “front”, “back”, “left” or “right”. We use a stereo camera system for two reasons: on the one hand there are situations, where only one foot is visible in one of the cameras, whereas both feet show up in the other (Figure 9). On the other

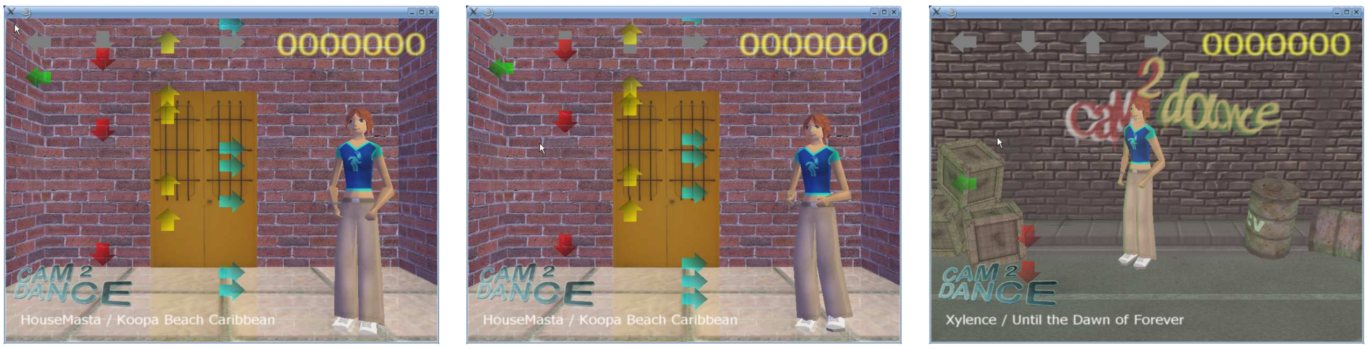

0000000
CAM 2 DANCE
HouseMasta / Koopa Beach Caribbean
0000000
CAM 2 DANCE
HouseMasta / Koopa Beach Caribbean
0000000
cam 2 dance
CAM 2 DANCE
Xylence / Until the Dawn of Forever

Figure 7: Screenshots

hand, we need to tell whether the foot is on the pad or is in the air, e.g. we need its 3D position, which can only be determined from stereo images.

We use planar fiducials, as we need the orientation of the feet as well as the distance of the feet to the floor.

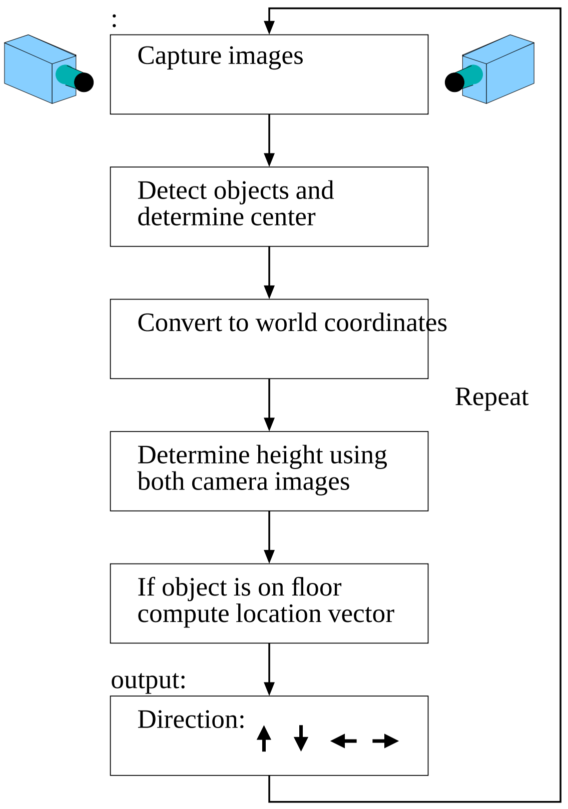

:
Capture images
Detect objects and determine center
Convert to world coordinates
Repeat
Determine height using both camera images
If object is on floor compute location vector
output:
Direction: ↑ ↓ ← →

Figure 8: Processing Steps

The goal of this work is to apply and further develop a robust real-time stereo tracking system for continuous use over several months.

The complete tracking system runs on a laptop with 512 MB RAM and 1.5 GHz Intel CPU. We use two color cameras connected by an IEEE1394-interface and capture images at 30 Hz with a resolution of 640×480 pixels. 8.5 mm lenses provide an appropriate field of view. The cameras are mounted on a rack at a height of 200 cm. The distance to the pad is approximately 200 cm. We track two markers found in the image region of the pad. The markers have a size of 6 cm $\times$ 6 cm; they are attached to shoes which the user has to wear (see Figure 10 and Figure 9). The markers were designed for the ARToolkit[3] The pad size is 90 cm $\times$ 90 cm.

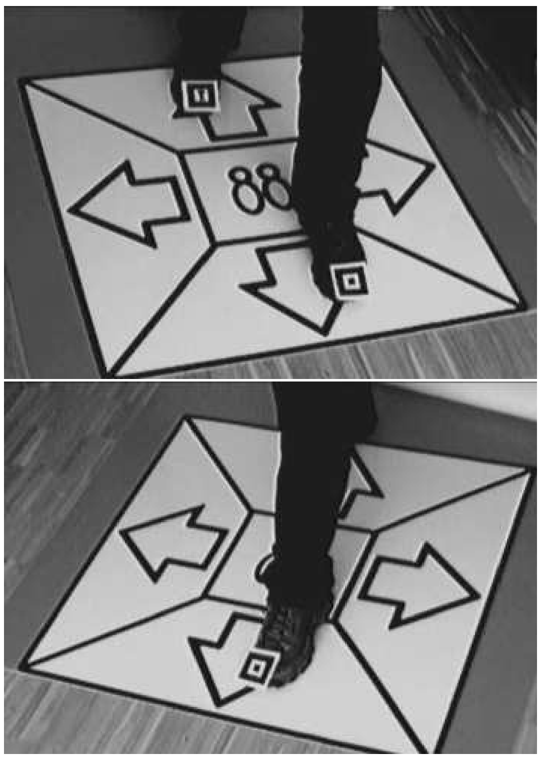


Figure 9: Occluded view on feet from stereo camera.

Several pre-processing steps are required for accurate tracking, in particular calibration of the system, detection of pad in camera image, settings for thresholds etc. As we need to check

[3] Hirokazu Katoa and Mark Billinghurstb and Ivan Poupyrev, ARToolKit, Handbook to version 2.33, 2000, http://www.hitl.washington.edu

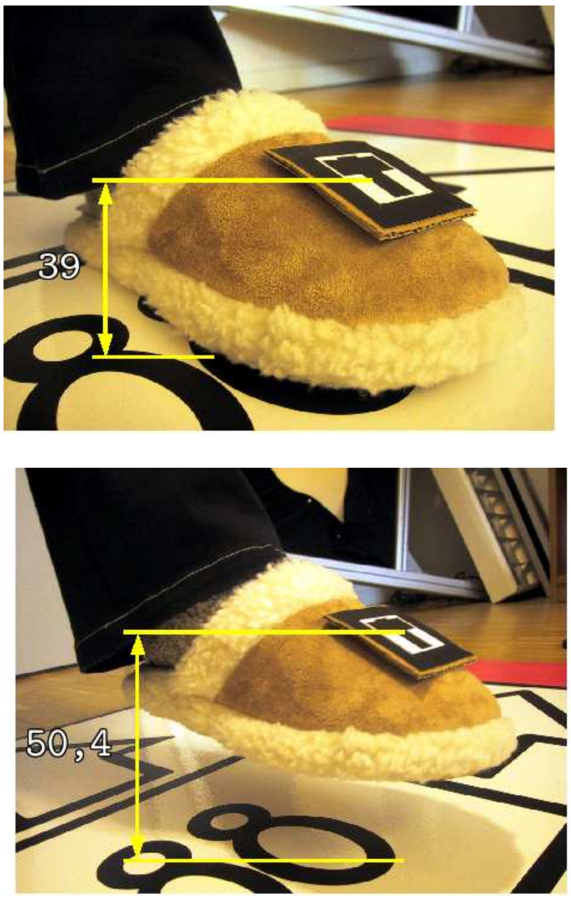


Figure 10: Marker height for foot and lifted foot

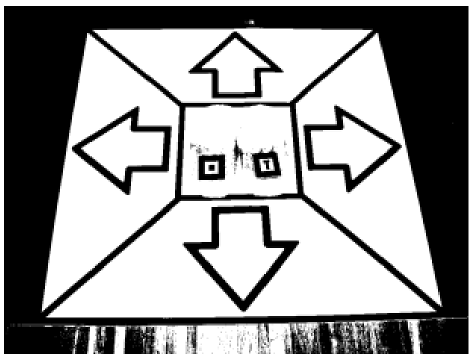


Figure 11: Binarization

whether the foot is on the dance pad or whether it is lifted, we need to set a range for the distance between the marker and the ground (see Figure 10). This distinction is the major reason why the system requires stereo cameras. Although we can estimate depth even form a single video stream using tracking, we need two images to determine the height of the foot over ground accurately.

Initially we compute the homography matrix $\boldsymbol{H}$ from the four

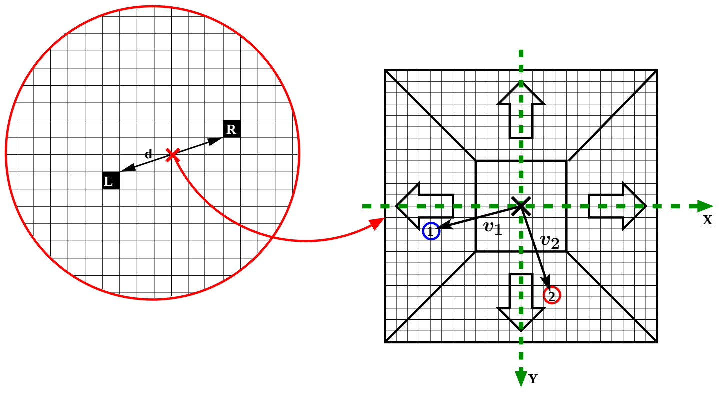


Figure 13: Vectors in 2D coordinate system

corners of the dance pad, which are located accurately in the binarized image (e.g. Figure 11). As ARToolKit requires calibration data, we do an initial camera calibration. As radial distortion is small in our setup, we work without further correction steps.

3.3 Tracking

Tracking using `ARToolKit` requires binarized images that are created by simple thresholding from color input images. This threshold is adjusted initially, when the shoes are placed on the pad and no person is in the field of view (Figure 11). To increase tracking accuracy, we not only determine the threshold for each image separately but for each marker as well, in total four different thresholds. By this proceeding we achieve a 2.5 times higher detection rate. A simple algorithm is used to find the optimal threshold, as shown in Figure 12. To determine the threshold, the markers are placed in the center of the pad.

When the foot rests on the pad, the marker's 3D position is above the pad. The height varies for different users and is determined at game startup (cmp. Figure 10).

A simple algorithm to compute the foot height is outlined in Figure 14. We assume that both feet are inside the area of the pad. Additionaly a 0.5 cm wide white boarder is used in order to increase the accuracy of the used tracking algorithms. Therefore the detection rate of the ARToolKit is 1.13 times higher than without it and the ARToolKitPlus markers could only be detected with this border attached. Our experiments also showed that the tracking algorithms of ARToolKit are more robust against changes in background colors, i.e. the color of the shoes.

The tracking module returns the 3D positions of the markers which are now projected to 2D coordinates of the pad as shown in Figure 13. The points are converted from one image coordinate system to the other using the homography $\boldsymbol{H}$

computed initially.

The output of the module – the 2D vectors for the feet – are passed back to the game control module (cmp. Figure 3).

3.4 Extensions

When the cameras are moved slightly, other man-computer interaction is easily possible. The modular system will permit to play other games which will track hands or faces. Users also asked for games where more than one person will be tracked.

4 Experiments and Results

The position of the dance pad with respect to the cameras is assumed to be fixed. The corners of the pad are marked manually in the stereo images during an initial calibration step.

Both methods of ARToolKit and ARToolKitPlus were evaluated with respect to correctness, tolerance, speed and accuracy. Several marker patterns were used, the ARToolKit markers are shown in Figure 15, as the id-based BCH-coded ARToolKitPlus markers the numbers 0 to 9 were used.

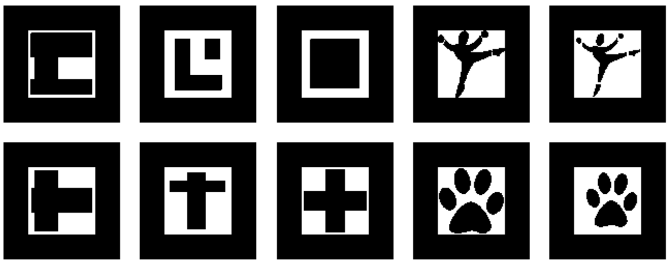


Figure 15: Marker patterns tested for ARToolKit

Mean values for computation times from 20.000 measured tests are shown in Table 1. Computation times were higher for ARToolKitPlus (Table 2).

The comparison of different markers is summarized in Figure 16.

To summarize: Overall the ARToolKit is better suited for our purposes. It is not only faster (Figure 2) than the ARToolKitPlus, it is also more accurate (Figure 17) and more robust against changing background colors. Using the ARToolKit clear rectangular shapes with a sufficient number of rectangular angles seem to perform best as marker pattern.

Users have been asked for their opinion on the human-computer interaction and their process of understanding the logic of the game. Those aged less than 25 usually asked for no further explanation of the program. They considered it part

	two thresholds	one threshold
Mean	2,777 ms	2,277 ms
Max	5,957 ms	3,449 ms
Min	1,842 ms	1,516 ms

Table 1: Computation times for ARToolKit

	two thresholds	one threshold
Mean	17,361 ms	16,48 ms
Max	37,091 ms	27,437 ms
Min	16,102 ms	15,486 ms

Table 2: Computing times for tracking using ARToolKitPlus

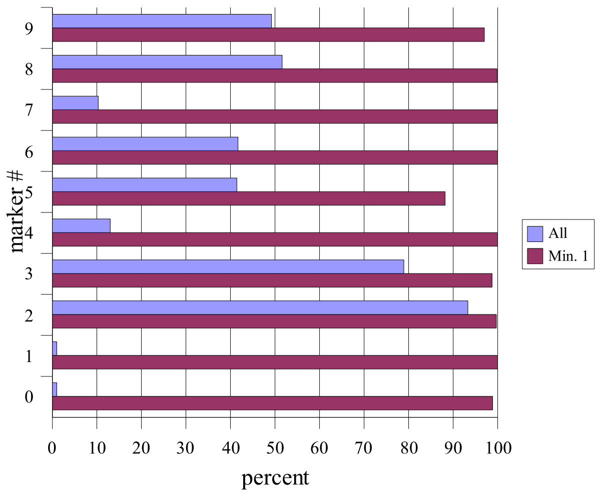
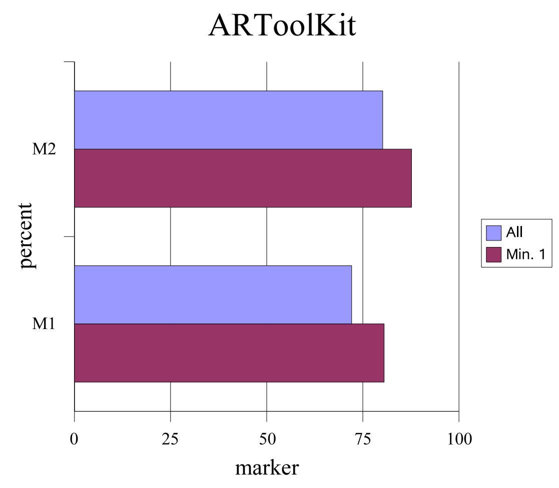


Figure 16: **Top:** Comparison of ARTK-markers (Figure 15), **Bottom:** Comparison of BCH-coded ARKPlus-Marker #0-9. The red bars are counted for markers found in at least one image. The blue bars display cases where markers are found in each image correctly.

of the game to find out, what to do and how to raise the game score. Users above 50 also tried the game but they often asked for an introduction, first. This document was not integrated into the game and the text is not displayed on the game monitor. Instead, a guideline was posted next to the game display on the wall. The feedback of the arrows was understood by the users. Quickly they realized that the „instructor" (the figure dancing on the right as seen in Figure 7) is much harder to follow than the anticipating the moves from the arrows that are shown on the left.

5 Conclusion

The system as a prototype is far from being perfect. The next steps in development will be a revision of the game graphics



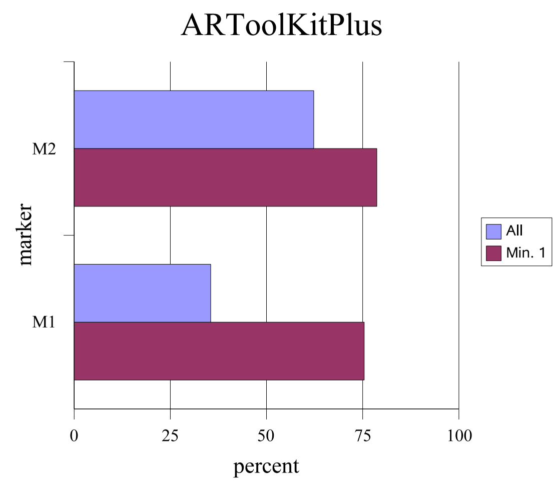


Figure 17: Average detection rate of the used tracking algorithms

using shaders (i.e. Cg), improved animation of the dance character and a tracking system that must not rely on markers to operate. The system has a lot of potential, especially since one major advantage towards classic dance games is currently not implemented yet, because of trying to keep the game play as simple as possible. With camera tracking each foot is separately identified, allowing the game to give individual moves for each foot, a feat not possible with a dance pad.

The contribution describes the vision and tracking part of a larger system that was outlined in Figure 3. The other modules are responsible for graphics, music, synchronization of music, visualization and tracking, and for the complete game logics.

In future work we will use marker-less tracking based on simple template matching to track the feet. The user will have to place his foot initially on a defined location on the dance pad so that the computer can initialize the template.

Acknowledgment

We want to thank Dr. Angela Kaiser-Lahme for the opportunity to take part in the exhibition and for the idea to develop a game soley therefore, building on our scientific background.

References

[1] Bowman, Doug A., Ernst Kruijff, Joseph J. LaViola and Ivan Poupyrev: *3D User Interfaces: Theory and Practice*. Addison-Wesley, New Jersey, 2004.

[2] Fiala, Mark: *ARTag Revision 1, A Fiducial Marker System Using Digital Techniques*. Technical Report 47419, National Research Council Canada, 11 2004.

[3] Fiedler, E., G. Platsch, A. Schwarz, K. Schmiedehausen, B. Tomandl, W. Huk, T. Rupprecht, N. Rahn and T. Kuwert: *Time consumption and quality of an automated fusiontool for SPECT and MRI images of the brain*. Nuklearmedizin, 42(5):215–219, 2003. Automatisierte Fusion von SPECT- und MRT-Bildern des Gehirns: Zeitbedarf und Ergebnisqualität. Journal of the German, Austrian and Swiss Societies of Nuclear Medicine. Schattauer GmbH.

[4] Freeman, W. T., K. Tanaka, J. Ohta and K. Kyuma: *Computer vision for computer games*. In *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, pages 100–105, Washington, DC, USA, 1996. IEEE Computer Society.

[5] GIERGA, DAVID P., JOHANNA BREWER, GREGORY C. SHARP, Margrit Betke, CHRISTOPHER G. WILLETT and GEORGE T. Y. CHEN: *The Correlation Between Internal and External Markers for Abdominal Tumors*. In *Int. J. Radiation Oncology Biol. Phys*, volume 61, pages 1551–1558. Elsevier Science Ltd., 2005.

[6] Hämäläinen, Perttu and Johanna Höysniemi: *A Computer Vision and Hearing Based User Interface for a Computer Game for Children*. Lecture Notes in Computer Science, 2615:299–318, 1 2003.

[7] Lepetit, V. and P. Fua: *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey*. Foundations and Trends in Computer Graphics and Vision, 1(1):1–89, 10 2005.

```
**Determine threshold for markers:**
Current threshold t = 80 (example of a vaule)
Final threshold endT = 200 (example value)
Initialize values for each marker $i$ per camera by 0:
counter count$_i$, maximum count max$_i$, start start$_i$ and end end$_i$ of longest sequence, auxiliary variable tempStart$_i$
FOR t until t == endT
    Search marker
    IF Marker found
    THEN
        IF count$_i$ == 0
        THEN Remember threshold for marker $i$ tempStart$_i$ = t
        count$_i$++
    ELSE
        IF count$_i$ > max$_i$
        THEN
            Store new value for longest sequence
            start$_i$ = tempStart$_i$
            end$_i$ = t-1
            max$_i$ = count$_i$
            count$_i$ = 0
    $t = t + 1$
Result: Threshold for each marker$_i$ = ( start$_i$ + end$_i$ ) / 2
```

Figure 12: Initialize thresholds for markers

```
**Determine foot height threshold:**
Input: time span for algorithm $time$ in ms
Start $start =$ current system time in ms
Stop $stop = 0$
Counter for markers found in center $inCenter = 0$
Counter for loop $k = 0$
Tolerance $p$
Set $\Sigma_i = 0$; Sum of all height values per marker $i$ ( $0 <= i <= 1$)
    Track marker
    IF Both markers on pad
    THEN
        Compute heights $h_1$ and $h_2$
        $\Sigma_1 = \Sigma_1 + h_1$
        $\Sigma_2 = \Sigma_2 + h_2$
        $inCenter + +$
    ELSE $inCenter - -$
    $k + +$
    $stop =$ current system time in ms
UNTIL $stop >= (start + time)$
IF $inCenter > 0$
THEN
    IF $|\Sigma_1/k - \Sigma_2/k| < 6$
    THEN
        Compute $p$
        New threshold for foot height: $((\Sigma_1/k) + (\Sigma_2/k))/2 + p$
    ELSE Return error *too large differences*
ELSE Return error *not found in center*
```

Figure 14: Determine foot height threshold