Camera-Based Virtual Environment Interaction on Mobile Devices

Tolga Çapın[1], Antonio Haro[2], Vidya Setlur[3], Stephen Wilkinson[4]

[1] Bilkent University, tcapin@cs.bilkent.edu.tr
[2] D4D Technologies, mail@antonioharo.com
[3] Nokia, vidya.setlur@nokia.com
[4] Texas Instruments, stephen.wilkinson@ti.com

Abstract. Mobile virtual environments, with real-time 3D and 2D graphics, are now possible on smart phone and other camera-enabled devices. Using computer vision, the camera sensor can be treated as an input modality in applications by analyzing the incoming live video. We present our tracking algorithm and several mobile virtual environment and gaming prototypes including: a 3D first person shooter, a 2D puzzle game and a simple action game. Camera-based interaction provides a user experience that is not possible through traditional means, and maximizes the use of the limited display size.

Keywords: camera-based interaction, virtual environment, mobile device, computer vision.

1 Introduction

Recent advances in mobile device hardware have made it possible to create mobile virtual and mixed reality environments. It is now possible to align real images spatially with synthetic content in real time [6]. However, to be suitable for real-world applications, key interaction limitations remain, including: a small physical display size, precise registration of virtual content with the real environment, and intuitive user input techniques.

Mobile devices currently support interaction through a joystick or a set of standard keys. Although these modalities are sufficient for simple modes of interaction, more intuitive interaction techniques are needed for mixed reality and virtual environment systems. For a greater degree of immersion, better tracking is essential.

To address these limitations, we present our interaction technique for camera-equipped mobile devices, using the camera sensor as an input device [5]. Our solution is based on analyzing the series of live input images from the camera and estimating the motion of the device. We apply our approach to some of the most interactive application scenarios: navigation and gesture-based interaction in highly interactive 3D virtual environments.

While previous work has created 3D and augmented reality game genres that map left/right/up/down camera motions to those exact motions in a game, our work also allows for gesture recognition using the camera's egomotion. For example, a character can be made to jump by shaking the mobile device; our approach

recognizes the shaking gesture and maps it to a jump action. In addition, camera motion can be mapped to 3D viewing. This is particularly useful on mobile devices where a limited number of buttons and limited display size make the creation of an immersive gaming experience difficult. By mapping physical motion to 3D viewing transformations in a first-person game, the user has a more immersive experience with no additional hardware or sensors required.

Figure 1: A 3D virtual environment. The user can move the device up, down, left and right to look around the space. Mapping physical motion to viewing direction in the space creates the illusion of a small window into an environment that surrounds the user.

2. Related Work

Our work is similar to that of Moehring et al. [6], where a 3D model was overlaid on live video captured by a smart phone's on-board camera viewing a set of markers used to capture orientation. Our work does not overlay graphics on video, rather video is analyzed without user-introduced tracking features to determine 2D ego-motion instead of full 3D camera pose. The scroll [8] and peephole [9] displays works are also related as their goals were to estimate 2D device motion towards the creation of more intuitive user interaction. While additional sensors were required in these works, such as mechanical and ultrasonic sensors, we use only the camera as our direction sensor. Doing so allows regular camera-equipped smart phones to have an additional interaction modality without modifying the phone hardware.

Camera-based user interaction has been used in games in several commercial works. The most successful mixed reality platform for gaming presently is Sony's Eyetoy [1], which consists of a 60 fps camera attached to a Sony Playstation 2 game console. Incoming video is analyzed to estimate a player's motion and to segment them from their environment. Our work has similar goals, but is focused on the mobile domain where hardware constraints are more significant. In the mobile domain, a number of mixed reality games are currently available. These range from camera-movement tracking for moving on-screen crosshairs to simple body part region tracking. Mobile games in this category include: Ghostblaster by Futurice Oy, Bitside GmbH's "HyperSpace Invasion" and "Marble Revolution". These are all similar in concept in that the camera can be moved horizontally and vertically to

move an on-screen cursor. Camera motions are directly mapped to in-game actions but with no gesture recognition or mapping to 3D interaction as in our work.

Related camera-based tracking work includes the Mozzies game available for the Siemens SX1 mobile phone. While camera motion is indeed estimated in these games to translate sprites accordingly, it should be noted that the detected motion does not need to be exact as the sprites are rendered on top of the video but not attached to any feature. As such, only approximate motion tracking is used. Since our tracked motion needs to match the user's physical motion more precisely, a higher degree of accuracy is required which from our testing is not present in current commercial camera motion tracking-based games. It should also be noted that in these works, only mobile camera motion is used as input; in our work we use the magnitude of the motion and shake detection as well.

Rohs et al. [7] perform tracking based on dividing incoming camera frames into blocks and determine how the blocks move given a set of discrete possible translations and rotations. Our algorithm is instead based on tracking individual corner-like features observed in the entire camera frames. This allows our tracker to recognize sudden camera movements of arbitrary size, as long as some of the features from the previous frame are still visible, at the trade-off of not detecting rotations. Kalman filter based camera motion estimation was demonstrated by Hannuksela et al. [4]. The Kalman tracker has higher motion estimation accuracy, as expected, since Kalman filtering greatly improves the quality of intra-frame matching. However, the computational requirements are significantly greater since several matrix multiplications and inversions are needed per frame. On devices with limited computational resources, our algorithm provides sufficient motion and velocity accuracy for user interaction, freeing computational power for intensive tasks such as collision detection and animation computations, at the trade-off of more limited accuracy, since the temporal filtering in our algorithm cannot match a Kalman filter.

3. Camera-Based Tracking

Our approach is to use the mobile phone's on-board camera as the source of input. The user's physical movement of the device is captured in incoming video, which is analyzed to determine scroll direction and magnitude. The detected direction is sent to the phone's UI event handler exactly as a corresponding mouse event.

3.1. Tracking Algorithm

Our tracking system was implemented on the Symbian OS. We use a feature-based tracking algorithm that determines how corner-like features have moved between the previous frame and the current frame. Finding corner-like features is difficult because traditional corner detectors are too computationally complex for real-time performance in gaming. Edges are easier to detect, however they are not temporally coherent. Instead, edge information in the x and y direction is combined to find corner-like features. The sum of the absolute values of the x and y derivatives, otherwise known as the Sobel operator, provides a practical estimate of corners. All

corners cannot be detected using the Sobel operator; however, it provides a useful first-step culling of pixels for additional processing.

Our algorithm is as follows. Frame n is filtered using the Sobel x and y filters. The 50 strongest features per frame are found and the motion of each is estimated using template matching. 15x15 search windows are used to find the features in the next frame since this size is sufficient to capture visually distinct regions and significant intra-frame motion. Once the relative movement direction is determined per feature, the sum of features that have moved in the up, down, left and right directions is totaled to remove temporal inconsistencies. Every 4 frames, the direction with the maximum count is used as the overall camera motion estimate. Since the tracker typically performs at 12 frames per second, the direction estimate is updated several times per second. This is the only temporal filtering needed since the features are robust. The direction estimation fails if at least one feature from the previous frame is not visible in the next frame, which we have not observed in our experiments in various indoor and outdoor environments. In practice, the tracked motion is smooth and amenable for viewpoint selection in virtual environments, especially since the tracker is not computationally expensive.

3.2. Shake Detection

We use motion history images (MHI) [3] to recognize camera shake gestures since MHIs were originally created for action and gesture recognition. MHIs capture motion histories of image sequences in a single grayscale image which can be processed to perform simple, computationally inexpensive gesture recognition.

MHIs are computed by performing background subtraction between the current and previous frames of a video sequence. At locations where the pixel values change, the MHI is updated by decrementing by a pre-defined constant amount. In this manner, the MHI compactly captures the variations of pixel differences at locations in the image across time. By averaging the intensity values of the MHI, the average camera motion magnitude can be estimated. The average camera motion can then be used to detect large, sudden movements or shaking of the device quite reliably and computationally inexpensively.

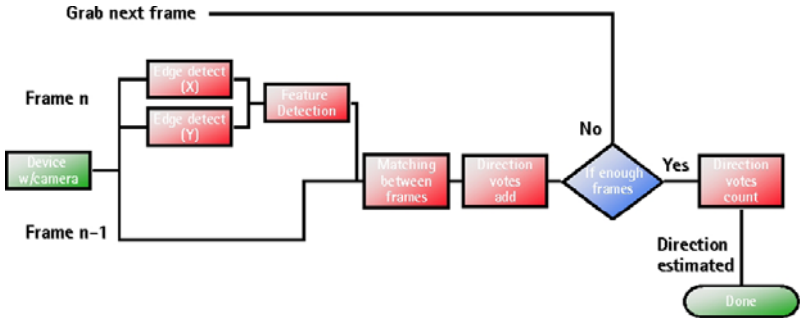

Figure 2. Our tracker uses the current and previous frame captured by the camera for tracking. Corner like features are detected in the new frame which are matched with the features found in the prior frame so that the prior frame does not have to be reprocessed for features. Direction estimates are accumulated for a number of frames before a movement direction estimate is made.

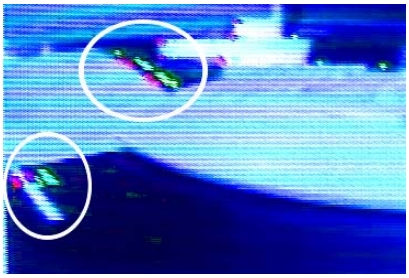

Figure 3: Feature detection and tracking to estimate 2D camera egomotion. Red pixels are features detected in the current frame, green pixels are features detected in the previous frame (leftward motion shown). The motion of individual features is aggregated to determine global egomotion.

4. Mapping Motion to 3D Interaction

Creating an immersive 3D experience is difficult on mobile devices due to the limited display size. The most immersive experiences are typically created using a combination of large displays reducing peripheral vision as much as possible and/or virtual environment navigation tied to the user's physical motion. In our prototype (Figure 1), we map the user's physical motion to the viewpoint to create the illusion of a window into a 3D world. We have used the tracking and shake recognition algorithms presented to create several mobile 3D game prototypes. In each case, using the camera as the input modality created a user experience not achievable with traditional keypad based input.

Our renderer loads standard Quake III (tm) or Quake III Arena (tm) maps. The rendering is done using the OpenGL ES implementation available on Series 60 based mobile devices. Pre-computed vertex lighting and fixed point calculations are used to improve performance due to the lack of a floating-point unit on our test hardware, a standard Nokia 6630 smart phone. The renderer is able to render realistically lit virtual environments with several thousand polygons per scene at 5-10 frames per second, depending on the environment that is used.

Interaction with the virtual environment is performed with a combination of physical motion and keypad presses. The navigation within the environment is controlled by the directional keys on the keypad. For viewpoint selection, the user looks around in the scene by physically moving the device in the directions that they would like to look. We map the tracked camera motion directions to a trackball as in traditional mouse-based 3D interaction. The game actions are mapped to keypad presses. In this paper, we do not address 3D object interactions such as picking.

5. Mapping Motion to 2D Interaction

Darken et al. [2] have studied 2D and 3D mixed-dimension interaction in virtual environments. They have concluded that a hybrid interface, where 3D and 2D interaction techniques were matched to each individual task, such as object selection, text reading, would be the most preferable solution. We consider that camera-based input can be used for 2D interactions in the virtual environment, as well as those that are 3D. To illustrate real-life applications, we created 2D puzzle and action game prototypes to investigate these ideas using the camera motion and shake detection algorithms presented.

We modified the open source Series 60 port of the “Frozen Bubble” puzzle game, switching the game control from using the keypad to using the camera (Figure 4). In our version, the user moves their mobile device left and right to aim and performs sudden shakes to launch their bubble. This has the effect of significantly changing game play as careful arm motions are now required to aim, instead of a number of button presses. Switching the aiming and launching to be camera-motion based has the result of creating a more physically-accurate aiming experience, which increases the excitement of the game.

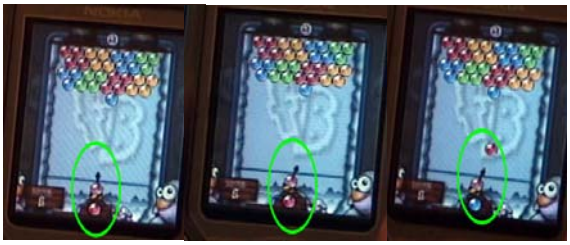

Figure 4: A 2D puzzle game. Players move the device left and right to aim, and shake the device to launch a bubble. Aiming is more difficult using physical motion than with a keypad, creating a more exciting experience.

We created a camera-based action game prototype as well. Using sprites and artwork from Konami's "Track and Field" game for the Nintendo Entertainment System, a new game (Figure 5) was created. A runner must jump over a never-ending number of approaching hurdles. To jump, the player must time the shaking of their device correctly so that the character does not crash into hurdles. Relying on the camera exclusively for input results in a game that is very simple to learn and understand but difficult to master. The timing of a button press is easier than that of a shake since most button press physical motions are very similar, yet most shaking motions are quite different. Consequently, to play well, a player must learn not only the timing needed but also how to perform consistent shaking motions for each hurdle so that the next shake is timed correctly. We believe this additional level of user involvement results in a game that provides richer sensory feedback.

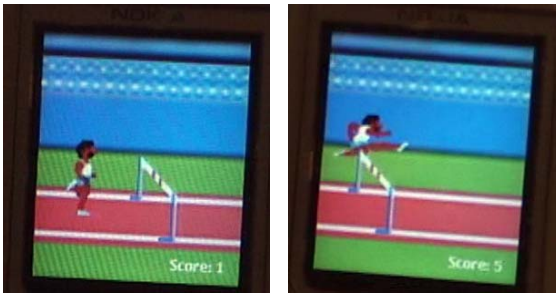

Figure 5: A 2D action game. The player's character must jump over the hurdles. A jump command is issued by shaking the device at the correct time to avoid tripping on the hurdle.

Results

We implemented the tracking algorithm and applications in C++ using the Series 60 Second Edition Feature Pack 2 SDK. Our test platform was a Nokia 6630 mobile

phone, which features an ARM 9 220mhz processor, 10 megabytes of RAM, 176x208 screen resolution, and a 1.3 megapixel camera capable of capturing frames at 15 fps.

In order to support intuitive and efficient user interaction, it is important to understand what kind of information is provided by the tracking algorithm, and what the limitations are given the output of the tracking algorithm. The most basic but potentially most important input that can be acquired from the tracking algorithm is the two dimensional movement of the mobile device on a plane parallel to the camera in 3D. With this type of data, the camera can be used as an input device to capture the device's movement in up/down, left/right directions, as well as its speed in each direction. Mobile camera-based input has restrictions, primarily due to limitations of mobile device hardware. Forward and backward motion cannot be detected with the current algorithm, so six degree of freedom movement is not supported yet. Forward/backward motion is possible to detect if the algorithm were extended, however this would increase computational demands and reduce the frame rate, impoverishing the user interaction.

Physical movement speed is another challenge for camera-based interaction. The algorithm must perform all of its video analysis in the time between camera frames being captured to support real time interaction.

Thus, there are implicit limits on the computational complexity of the tracking. In addition, there is a fundamental assumption in our algorithm that each frame contains some portion of the prior frame. This assumption is motivated by the observation that users will typically not move their phones erratically when focused on a task. We have also verified our tracking solution in informal experiments and found that it works well in practice. Users usually operate mobile phones with one hand. Mobile phones can also be used anywhere in an office, school, public, home, etc. Considering these environments, there are certain interactions which are not appropriate:

Precise tasks: Precise motion is very difficult holding a mobile device with one hand. Interaction should not require operations like 'move the device 2.5cm up', or 'move the device 34 degrees from the horizontal line.' As a result, camera-based interaction will probably be most useful when navigating large amounts of data, or zoom level dependent data.

Large motion: This restriction is more serious in some environments, such as in crowded public locations. In such situations, it may be advantageous to provide a 'clutch' to turn the tracking on/off. This would emulate the act of lifting a mouse once the edge of a desk is reached in traditional desktop interaction. In our informal testing we did not provide a clutch, however in commercial implementations this is a consideration to keep in mind.

Extended and/or frequent interaction: Using single handed operation, interactions that require extended time and/or frequent movement may fatigue users.

Our approach works best with coarse selections at different speeds and scales of data. It is critical that visual feedback follows physical motion and that the feedback differs according to motion speed, in order to provide an intuitive user experience. The most typical use case is moving the device to scroll UI content such as a list or a document.

Conclusions

We introduced a new approach to improve the user experience for interacting with virtual environments on mobile devices. A computer vision-based tracking algorithm was presented to detect both physical motion direction and gestures to permit one-handed physical movement based interaction. A camera was chosen since cameras are now widely available on mobile devices and are very powerful sensors that can be used without introducing new sensors.

We demonstrated our approach in 2D and 3D interaction. In the future, we would like to collect user feedback to determine how to improve user interaction further using mobile cameras. While we applied the camera-based interaction to only viewpoint selections and simple gestures, we would like to investigate its application to navigation, object interactions, and avatar control.

While our tracking algorithm is computationally efficient and works well in practice, there are some situations that cannot be handled. Severe lighting differences will cause the template matching to stop working properly. Motion in front of the camera is ambiguous and can affect tracking results as it is impossible to tell whether the camera is moving or not without either significantly more expensive computations or other sensors. Shadows may confuse the tracking system, but there are known computer vision techniques for robust tracking in the presence of shadows that will be incorporated into the tracking algorithm once additional processing speed is available.

References

1. Campbell D., Computer Vision in Games, Game Developers Conference 2005.
2. Darken R.P., Durost R., Mixed-dimension interaction in virtual environments. VRST2005: pp. 38-45.
3. Davis, J.W. and Bobick, A. The Representation and Recognition of Action Using Temporal Templates, In IEEE International Conference on Computer Vision and Pattern Recognition, August 1997, pp. 928-934.
4. Hannuksela, J., Sangi, P., and Heikkila, J. A Vision-Based Approach for Controlling User Interfaces of Mobile Devices. To appear in IEEE Workshop on Vision for Human-Computer Interaction (V4HCI), 2005.
5. Haro A., Mori K., Capin T., Wilkinson S., Mobile Camera-Based User Interaction. *Proceedings of ICCV-HCI* 2005: 79-89.
6. Moehring, M., Lessig, C., and Bimber, O. Video See-Through AR on Consumer Cell Phones. In *Proceedings of ISMAR 2004*.
7. Rohs, M. Real-world Interaction with Camera-phones, In *2nd International Symposium on Ubiquitous Computing Systems (UCS 2004)*, Tokyo, Japan, November 2004.
8. Siio, I. Scroll Display: Pointing Device for Palm-top Computers, *Asia Pacific Computer Human Interaction 1998* (APCHI 98), Japan, July 15-17, 1998, pp. 243-248.
9. Yee, K-P. Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers, In *Proceedings of CHI'03 Human Factors in Computing Systems*, pp. 1-8.