Range Camera for Simple behind Display Interaction

Anton Treskunov, Seung Wook Kim, Stefan Marti
Samsung Information Systems America (Samsung Electronics US R&D Center)
300 Orchard Pkwy, San Jose, CA 95134, USA
{treskunov, seungwook.k}@gmail.com; stefanm@media.mit.edu

Abstract

This paper presents a method to estimate hand positions behind a display, while the person's body is in front of the display. This allows for direct and bare hand interaction with virtual objects in an AR setting. As a sensor, we use a time-of-flight range camera, which provides depth information per pixel, but is noisy. While hand tracking has previously been attempted by localizing body parts and fitting a body model into the data, we propose a simpler algorithm. We first remove outliers, and then model the hand as an oriented box whose position and orientation is determined by a PCA algorithm. The calculated hand model is then used in a physics engine based interaction with virtual content.

1. Introduction

Progress in the field of computer vision together with increases in raw processing power create an opportunity for new interaction methods with computers, making the human body the actual controller (as opposed to using traditional controllers like keyboard, mouse, joystick etc.). Such a system needs to reliably and in real-time identify, locate, and track relevant parts of the user's body. This task is made easier by a proliferation of range cameras that measure per-pixel distance (as opposed to light intensity and color). There is a growing body of research in body tracking using depth sensors, and commercial applications are starting to appear, such as the Microsoft Kinect.

In general, applications that use body tracking in the user interface need to solve issues such as body self occlusion, how to isolate relevant body parts and discriminate between multiply people, as well as how to overcome hardware limitations that may result in noise and low resolution.

However, in many cases the vision task could be simplified, particularly if the interaction happens in close range. We have developed a human computer interaction method where the user is located in front of a 3D display (implemented, i.e., as stereoscopic rendering on a transparent display), and interacts with a spatial environment by reaching with his hand behind the display. That way, the hand is spatially co-located with real and virtual content and, at the same time, does not obscure the display surface.

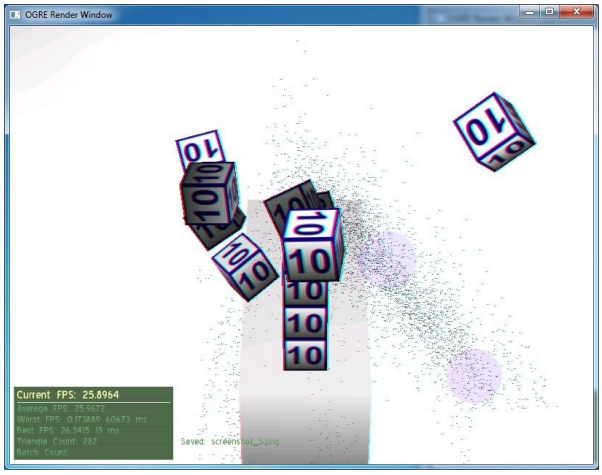

Figure 1. A screenshot of the interaction. A wall of dices is toppling over after colliding with a person's hand, represented by two pink spheres; dots are filtered points from the range camera before outlier removal.

We have found several ways to streamline the implementation. First, instead of precise localization and identification of body parts and guessing user intentions, we do a rough approximation of the interacting body parts as a low polygon model and check it for collision with CG content. A GPU accelerated physics engine handles collision detection. Second, in a situation where the user is located in front of the display, and his hand does manipulations behind the display, it is reasonable to assume that the camera sees only the manipulating hand of the user in front of a mostly static background. That assumption eliminates the need to identify and track body parts. The remaining pieces are hand localization, and approximating the hand as a volumetric body for the purpose of occlusion detection. This is a non-trivial task as depth sensor data is inherently noisy.

Figure 2. User is interacting with virtual content by reaching behind display.

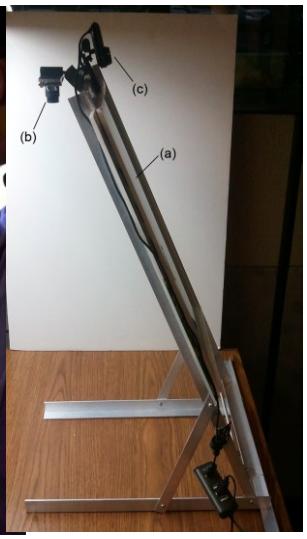

Figure 3. System side view: (a) transparent LCD panel; (b) depth camera; (c) Webcam.

In the rest of this paper, we first will describe existing work in related domains, and give an overview of the hardware setup that we used. In the Algorithm section, we will describe steps to mitigate sensor noise by filtering outliers and to calculate a simple hand representation as an oriented bounding box. We will conclude with a list of further applications for our system.

2. Related Work

There is a rich research body in the domain of hand tracking: for example, Piekarski and Thomas in their 2002 Tinmith-hand work [12] use fiducial marker detection via ARToolKit to perform 3 DOF tracking of the hands. Buchmann at al. [2] utilized fiducial marker tracking for hand and two fingers in their 2004 FingARtips work. Instead of using flat markers, Wang and Popovic [16] used a multi-colored glove in their 2009 paper. In 2005, Kölsch and Turk [8] described the HandVu hand tracking system. After explicit initialization, HandVu tracks the learned hand in the 2D image plane using a non-stationary monocular color camera. The tracking was demonstrated to be robust against camera movements and arbitrary cluttered backgrounds. In 2007, the Handy AR work by Lee and Höllerer [10] tracked an outstretched hand in space by recognizing fingertips. Tracking is initialized with a calibration step in order to learn the hand geometry. The user's palm is then used to place CG objects on it for AR applications. Lee at al. [9] in their 2008 work used skin tone based hand detection and 3D from stereo for hand tracking. However, they report "most of the users commented on the unstable fingertip tracking which sometimes affected their interaction."

There is also fast growing field of time-of-flight cameras [6] and its applications. As mentioned by Ghobadi at al. [5], while "range images of TOF cameras are independent of the texture and lighting conditions, they are somehow affected by the color of the object." In other words, and in accordance with our experience, they are quite noisy. Ghobadi at al. use a combination of range data with intensity image to separate hands from other body parts. Plagemann at al. [13] use a novel interest point descriptor together with a boosted patch classifier to localize body parts using range data. However, in our case body part segmentation is not necessary.

3. Hardware setup and system overview

Our demo system consists of interactive display hardware, as well as real-time 3D rendering and motion tracking software. The hardware platform includes a transparent LCD panel, a Webcam and a depth camera (a time-of-flight range camera by PMDTech) [14]. The software platform integrates the 3D graphics engine OGRE [11] with our hand tracking algorithm and the commercial face tracker FaceAPI [4]. The transparent LCD panel exhibits a transparency of 10% when it displays entirely white color or when it is completely turned off. With sufficient illumination around the back region of the LCD panel, the user can see his hand through the translucent display (Figure 2).

On the top part of the display, a depth camera is attached facing downward (Figure 3), toward where the user's hand is located during the reaching-behind interaction we propose. Hand detection data, which is collected from the depth camera and processed through our algorithm described in the next section, is visualized as a virtual hand in the real-time 3D environment based on OGRE. When the virtual hand touches virtual objects in the 3D scene, the built-in physics engine simulates collision and gravity effects, providing the user with natural visual cues (Figure 1).

To increase the reality of the interaction, the demo renders the 3D scene stereoscopically, so that virtual objects appear *behind* the transparent display, rather than *on* the display surface. Since our current transparent panel can render only at 60 Hz, we use anaglyphic stereoscopic rendering that require the user to wear red-cyan glasses.

Due to the optical-see-through nature of the display, a CG scene should be registered with the real world behind the display to be convincingly augmenting it. In addition, using only stereoscopic rendering of content on a display surface confuses our visual perception mechanism because the lack of additional depth cues such as motion parallax.

To improve accurate registration and to provide motion parallax depth cues, our demo system tracks the user's face in real-time and adjusts the perspective of 3D scene accordingly. A Web camera (as shown in Figure 2) detects the user's face with 30 fps and provides the scene renderer with 3DOF face position. The renderer then transforms and translates its viewing frustum for perspective correction. This view-dependent rendering method allows that (1) the virtual content appears in place with reference to the physical background, and (2) the user experiences different perspectives of the 3D content as he moves around in front of the display.

4. Hand Tracking Algorithm

We use a tracking-by-detection approach with a frame-to-frame dependency. The remainder of this section describes the hand detection and localization procedures.

Our demo implementation uses a time-of-flight (TOF) depth camera which provides a 200x200 pixel array of grey-level intensity, signal amplitude, distance, and calculated x,y,z coordinates. This data capture procedure is running in a separate thread, with a new frame being available every 45 ms.

Because the camera is facing away from the user and sees only his hand, we do not need to detect body parts (as described by Plagemann et al. [13]), and can approximate foreground objects (i.e., a hand) with an oriented bounding box.

Since the camera provides signal amplitude in addition to the pixel depth, we can reduce pixel noise with a simple amplitude threshold. Those weak pixels correspond to either a distant background, or to patches of bad infrared reflectivity. In a typical scenario, when the hand is 20 cm away from the camera, about 70% of the pixels have a weak signal and are discarded therefore.

In the case of a static display, to avoid a recognition step, we assume that the background remains approximately static, and filter it out in a method similar to z-buffer. To this end, we accumulate the depth data for every pixel over 50 consecutive frames; assign some big depth number when the signal is weak, and collect minimal depth values per pixel. The procedure eliminates another 10% of all pixels.

After a data frame is filtered based on amplitude (discard pixels with a weak signal) and background test, the remaining pixels form a point cloud, and statistical filtering is applied to eliminate isolated outliers.

This technique is similar to Rusu et al. [15] and proceeds as follows. We use the ANN library [1] to form a KDD tree and calculate the k nearest neighbors in 3D space for each point p_i from the cloud. To mitigate small clusters of outliers, we ignore the first (nearest) k_0 neighbors and store the average distance D_i to the rest $k - k_0$ ones. Then points with value D_i too different from the rest are discarded as outliers. I.e., point p_i is outlier if $D_i > D + \alpha\sigma$, where D is the mean of $\{D_i\}$, and σ is a standard deviation. Coefficient α was set to 0.1 empirically; values k and k_0 to 30 and 10 respectively.

The resulting points are clustered based on distance, and only the largest cluster is kept. For that step, we reuse neighbors the lists found in the previous step, so clustering is rapid. These two distance-based filtering steps remove about 30% of all points from the cloud. The contribution of the different filtering steps to the overall detection time is summarized in Table 1.

However, the process of finding neighbors is computationally expensive and takes most of the algorithm time (see below). To keep the algorithm running in real-time, we down-sample the image before extracting the point cloud for statistical filtering and clustering.

Table 1. Percentage of 3D points reported by range camera eliminated by filtering steps.

Weak	69%
Background	10%
Outliers	3%
Total	82%

To summarize, the following steps are involved in filtering the sensor data:

1. Discard weak pixels
2. Discard pixels failing z-test
3. Down-sample
4. Form point cloud
5. Find nearest neighbors for each point
6. Discard outliers based on average distance
7. Cluster points based on mutual distance
8. Keep the largest cluster

Orientation and size of an approximating box are calculated using Principal Component Analysis (PCA) [7], using Eigen library [3]. PCA is performed over the remaining points. Eigenvectors of the covariance matrix define the box orientation. In case the matrix determinant is negative, we reverse the sign of the least significant Eigenvector. The box size is approximated as $6\sqrt{\lambda_{x,y,z}}$, where $\lambda_{x,y,z}$ is the corresponding Eigenvalue, i.e. well-known ±3σ interval. Finally, for collision based interaction methods, we add two spheres on the opposite sides of the calculated box (Figure 1).

Overall, hand detection and localization takes 22 ms on an Intel i7 CPU running at 2.8 GHz, with most of the computation used by step 5 in the filtering sequence, as summarized in Table 2.

Table 2. Running time of the algorithm, in the case of a hand 20 cm from sensor, on a i7 CPU at 2.8GHz

Weak & background filter	0.3ms
Down-sampling	0.3ms
KDD tree creation	1ms
Neighbor search	20ms
Clustering	0.3ms
Total detection time	22ms

5. Conclusion

This paper has presented a method for tracking a person's hand behind the display using a range camera. This setup allows for a simple but natural interaction method using bare hands with virtual objects in Augmented Reality applications.

Instead of building a body parts model, the presented method relies on setup-imposed constraints, and models the hand as an oriented bounding box, which is used for a physics engine driven simulation. A performance evaluation shows that the algorithm runs in real-time on a modern PC, not requiring a powerful GPU module.

Other applications for this method include gaming (direct interaction with game content, game characters, etc.), virtual worlds (e.g., real person shaking hands with avatars), telepresence (combining local and remote video and CG content, and manipulating remote content), CAD/CAM (direct manipulation of CG objects), medical imaging (navigating and manipulating interactive 3D medical image content), and many more.

References

[1] ANN: A Library for Approximate Nearest Neighbor Searching. http://www.cs.umd.edu/~mount/ANN/ accessed on April 7, 2011

[2] Buchmann, V., Violich, S., Billinghurst, M., and Cockburn, A. (2004). FingARtips: gesture based direct manipulation in Augmented Reality. In Proc. Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE '04), Stephen N. Spencer (Ed.). ACM, New York, NY, USA, 212-221.

[3] Eigen C++ template library http://eigen.tuxfamily.org/index.php?title=Main_Page accessed on April 7, 2011

[4] FaceAPI, accessed on April 7, 2011 http://www.seeingmachines.com/product/faceapi/

[5] Ghobadi, S. E. , Loepprich, O. E. , Hartmann, K. , Loffeld, O. (2007). Hand Segmentation using 2D/3D Images. In Proc. Image and Vision Computing New Zealand, pp. 64–69, Hamilton, New Zealand, December 2007.

[6] Gokturk, S., Yalcin, H., and Bamji, C. (2004). A time of flight depth sensor, system description, issues and solutions. In IEEE workshop on Real-Time 3D Sensors, 2004.

[7] Jolliffe, I. T. (1986). Principal Component Analysis. Springer-Verlag. pp. 487. doi:10.1007/b98835. ISBN 978-0-387-95442-4.

[8] Kölsch, M., and Turk, M. (2005). Hand Tracking with Flocks of Features. In Video Proc. CVPR IEEE Conference on Computer Vision and Pattern Recognition.

[9] Lee, M., Green, R., Billinghurst, M. (2008). 3D Natural Hand Interaction for AR Applications. In Proc. 23rd International Conference Image and Vision Computing New Zealand, 26-28 Nov 2008

[10] Lee, T. and Höllerer, T. (2007). Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking. In Proc. IEEE International Symposium on Wearable Computers (ISWC), Boston, MA

[11] Ogre3D Open Source 3D Graphics Engine http://www.ogre3d.org/ accessed on April 7, 2011

[12] Piekarski, W. and Thomas, B. H. (2002). Using ARToolKit for 3D Hand Position Tracking in Mobile Outdoor Environments. In 1st Int'l Augmented Reality Toolkit Workshop, Darmstadt, Germany.

[13] Plagemann, C., Ganapathi, V., Koller, D., Thrun, S. (2010). Real-time identification and localization of body parts from depth images. In Proc. of ICRA'2010, pp.3108-3113

[14] PMDTec CamBoard http://www.pmdtec.com/products-services/reference-design/ accessed on April 7, 2011

[15] Rusu, R. B. , Marton, Z. C. , Blodow, N., Dolha, M., and Beetz, M. (2008). Towards 3D Point Cloud Based Object Maps for Household Environments. In Robotics and Autonomous Systems Journal, Special Issue on Semantic Knowledge, 2008.

[16] Wang, R.Y., and Popovic, J. (2009). Real-Time Hand-Tracking with a Color Glove. In ACM Transaction on Graphics SIGGRAPH 2009, 28(3)