Low-Cost Raycast-based Coordinate System Registration for Consumer Depth Cameras

Dennis Wiebusch* Martin Fischbach Florian Niebling Marc Erich Latoschik

University of Würzburg

Abstract

We present four raycast-based techniques that determine the transformation between a depth camera's coordinate system and the coordinate system defined by a rectangular surface. In addition, the surface's dimensions are measured. In contrast to other approaches, these techniques limit additional hardware requirements to commonly available, low-cost artifacts and focus on simple non-laborious procedures. A preliminary study examining our Kinect v2-based proof of concept revealed promising first results. The utilized software is available as an open-source project.

Index Terms: H.5.1 [Information Systems]: INFORMATION INTERFACES AND PRESENTATION—Multimedia Information Systems I.4.8 [Computing Methodologies]: IMAGE PROCESSING AND COMPUTER VISION —Scene Analysis;

1 Introduction

Today, many virtual, augmented, and mixed reality applications rely on low-cost markerless motion tracking, commonly achieved using depth cameras. Obtained positions and orientations typically have to be transformed from the camera's to the application's coordinate system. For this purpose, a coordinate system transformation has to be determined. Required offsets can be measured manually, if the hardware setup is simple, e.g., if the camera can be placed right above a single screen with no tilt. However, in other setups a manual measurement may be very time consuming, require special equipment, or result in very poor accuracy; for instance in CAVE-like multi-screen setups, when using Display Walls, or if a certain viewing angle is required [1, 4]. To our knowledge, no easy-to-use approach for registering coordinate systems of screens and depth cameras has been proposed before.

In this paper we present four raycast-based techniques, which determine the transformation between a depth cameras's coordinate system and the coordinate system defined by a rectangular surface. In addition, the surface's dimensions are measured. This does not only allow to realize common techniques, like head tracking or pointing in single- or multi-screen setups, but also to register the depth camera's coordinate system with any rectangular surface in the physical environment, like a table or a wall. The presented techniques focus on (1) limiting additional hardware requirements to commonly available, low-cost artifacts and on (2) simple procedures that achieve reasonable accuracy with respect to the accuracy of the camera.

2 Related Work

There are multiple techniques that focus on the detection of objects and the subsequent registration of multiple depth cameras based on point clouds [2, 3]. However, these approaches do not allow to register with objects that are not visible to the cameras, e.g., screens below the sensor. Other techniques solve this task, but only for optical marker-based tracking systems [5]. Finally, approaches that do achieve the desired registration rely on known real-world positions of screen corners and objects that can be seen by each camera [4].

3 Raycast-based Registration

All presented techniques build upon casting and recording rays from several positions that each intersect one corner of a rectangular surface (as in [5]). The intersection points of all associated rays determine the corners' positions in the depth camera's coordinate system. The vectors $\vec{r}$ and $\vec{u}$ along the screen's edges, its normal $\vec{n} = r \times u$, as well as its center $\vec{c}$ are calculated from these positions. The surface's dimension is calculated from $\vec{r}$ and $\vec{u}$ and the sought transformation matrix $M = \begin{pmatrix} \vec{r}/|\vec{r}| & \vec{u}/|\vec{u}| & \vec{n}/\vec{n} & \vec{c} \end{pmatrix}$ is composed.

This approach allows to register with surfaces that are outside the camera's field of view, e.g., a screen located below a Kinect v2 sensor, as long as two points on the required ray are visible to the camera. Due to unavoidable imprecisions in the measurement process, originating from hardware, software, or the human operator, two rays cast at the same point in space may not intersect. In that case the intersection point is approximated by the center of the shortest line segment between the two rays.

However, recording more than two rays per corner may increase the achieved accuracy, since it counteracts low precision. In this case, the sought transformation matrix is overdetermined and has to be approximated with respect to a certain error metric. One simple solution is to average the approximated intersections of all rays pairs for each corner. In addition, the application of estimation methods like the RANdom SAmple Consensus algorithm facilitates the removal of outliers.

3.1 Recording Rays

The principal approach for recording rays is to determine two key points in the camera's coordinate system that define the ray. Three of the presented approaches require access to the camera's depth image and a thereto registered color image. The fourth one requires access to the joints of a skeleton extracted from the depth image.

For a proof of concept we used Microsoft's Kinect v2 and the Flexible Action and Articulated Skeleton Toolkit to register the coordinate system of a monitor ($1.018\,\text{m} \times 0.572\,\text{m}$). The supplied APIs allow to access all required data: positions of specific skeleton joints are accessed using respective identifiers via VRPN. In order to select specific points in Kinect's depth image to determine key points on the ray, a dedicated software tool has been developed. It displays an RGB-D image that is calculated using functions provided by the Kinect SDK and supports to freeze frames, select pixels, and store collections of thus defined rays. A screenshot of the implemented tool[1] and our setup are shown in figures 1e and 1f.

Skeleton-based Pointing (SP) The first technique utilizes joints from the skeleton reconstructed by the Kinect SDK. A user points at a corner and a ray is cast along the line specified by her head joint and right hand joint (see figure 1a).

*e-mail: dennis.wiebusch@uni-wuerzburg.de

[1]The developed software is available at `https://github.com/simulator-x/raycast-based-registration.git`

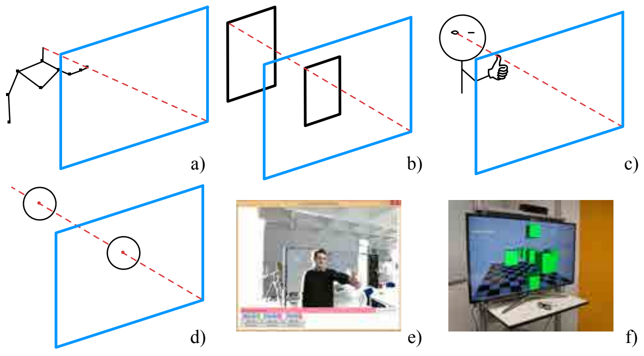

Figure 1: Illustration of the four presented ray casting techniques: a) skeleton-based pointing, b) edge bearing, c) thumb bearing, as well as d) thread and sphere. In addition, a screenshot of the dedicated tool for ray determination e) and the hardware setup used for the preliminary user study f) is shown.

Edge Bearing (EB) The second technique is based on the alignment of the top right corners of two boards, e.g., wooden plates or books, with one of the screen's corners (see figure 1b). This can be done by sense of proportion or—to speed up the process—by attaching a laser pointer to the edge of the rear board. The corresponding ray is recorded by selecting the respective corners of the boards as key points using the dedicated tool.

Thumb Bearing (TB) The third technique requires a user to position her thumb in a way that a ray cast from one of her eyes through the thumb intersects a corner of the screen (see figure 1c). The corresponding ray is recorded by selecting the user's pupil as well as the center of the thumb tip as key points using the dedicated tool.

Thread and Sphere (TS) The fourth technique requires two spheres that are attached to a thread. Two users stretch this thread and position one end directly at a corner of the screen (see figure 1d). The corresponding ray is recorded by selecting the two exit points of the thread at the spheres surfaces' as key points using the dedicated tool. In order to simplify the process, the exit points can be colored to be located in the RGB-D image more easily, or the centers of the spheres can be detected using an automatic fitting algorithm.

4 Evaluation

Since it is virtually impossible to obtain the exact registration of the Kinect sensor's with the screen's coordinate system, we compared the transformations that were determined with the presented techniques to a manually measured one. The latter was obtained by aligning the Kinect sensor with the screen, leaving only a positional offset in one direction to be measured and compensated by the transformation. Results of the comparison are shown in table 1.

Technique SP performs by far the worst, since the skeletal reconstruction only provides a rough estimation of the position of the user's eye and fingertip. The EB technique generates surprisingly bad results (visible in the screen size measurements). This is probably due to the fact that TOF cameras, like the used Kinect v2 sensor, generate erroneous measurements at edges of objects. The TS an TB techniques yield comparable results.

We conducted an additional preliminary user study to compare the utility of the four techniques. For this purpose, a scene consisting of eight cubes was created: one in the center, surrounded by one in every direction, and an eighth one in the back (see figure 1f).

Participants were asked to inspect the scene (rendered in stereoscopic 3D) by moving around and to observe the objects for their shape stability. After each task, the users had to rate the stability on a scale from 0 to 100. Ten participants (3 females and 7 males) aged between 18 and 50 ($M = 25.65$, $SD = 5.36$), 8 of which had experience with head-tracked setups, took part in our pre-study. The results are shown in figure 2. No significant differences could be observed between the presented techniques and the manually determined registration. This result is assumed to be owed to the high latency induced by the Kinect sensor, masking the effects to be measured. However, findings suggest that all techniques except for the skeleton-based one perform as good as the manual measurement regarding perceived errors.

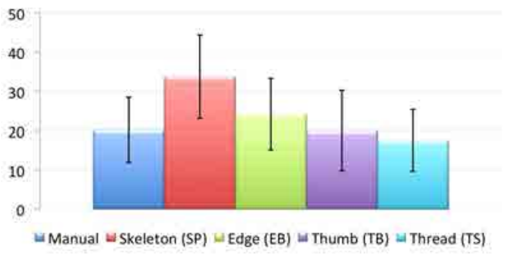

Figure 2: Results of the pre-study. Findings suggest that all techniques besides the skeleton-based one yield no perceivable difference to the estimated optimal (i.e. manually measured) registration.

	center [mm]			**dimensions** [mm]		**angles** [°]	
T	**x**	**y**	**z**	**w**	**h**	**ori.**	∡
SP	15	139.5	367.8	323.5	124.6	9.88	4.24
TB	13.9	0.3	33	1.1	9.3	2.32	0.47
EB	18.6	8	22.7	42.2	30.4	5.76	1.57
TS	3.3	22.8	19.1	15.3	1.3	3.88	0.04

Table 1: First comparison of registration errors for the presented techniques (T). The deviations from the manually measured center and orientation (ori.) describe the errors regarding the determined matrices. In addition, the error concerning screen dimensions and perpendicularity of $\vec{r}$ and $\vec{u}$ (∡) is listed.

5 Conclusion

We presented four simple low-cost raycast-based techniques to register the coordinate system of a rectangular surface with a consumer depth camera's coordinate system. A preliminary evaluation of a Kinect-based proof of concept shows promising results for the *Thread and Sphere* and the *Thumb Bearing* techniques. Of these two, the TS technique is potentially most flexible, since it does not require the user to face the Kinect. Next steps include detailed accuracy measurements, software automation of the processes, as well as a study designed to reduce the influence of the Kinect's low latency on the user experienced shape stability by running the determined registrations with a professional high-speed tracking system.

References

[1] Y. Chen, Z. Liu, P. A. Chou, and Z. Zhang. ViiBoard: Vision-enhanced Immersive Interaction with Touch Board. `http://research.microsoft.com/en-us/projects/mic_viiboard/`, 2015.

[2] A. Khosravani, M. Lingenfelder, K. Wenzel, and D. Fritsch. Co-registration of kinect point clouds based on image and object space observations. In *Proceedings of LC3D workshop*, 2012.

[3] R. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE, May 2011.

[4] A. D. Wilson and H. Benko. Combining Multiple Depth Cameras and Projectors for Interactions On, Above, and Between Surfaces. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 273–282. ACM, 2010.

[5] B. Wöldecke, D. Marinos, and C. Geiger. Flexible Registration of Multiple Displays. In *Proceedings of The International Symposium on Pervasive Displays*, PerDis '14, pages 86–91. ACM, 2014.