Positional Tracking With Time-of-Flight Depth Sensor and Webcam

Shannon Kao and Marianna Neubauer
Stanford EE 267, Virtual Reality, Course Report, Instructors: Gordon Wetzstein and Robert Konrad

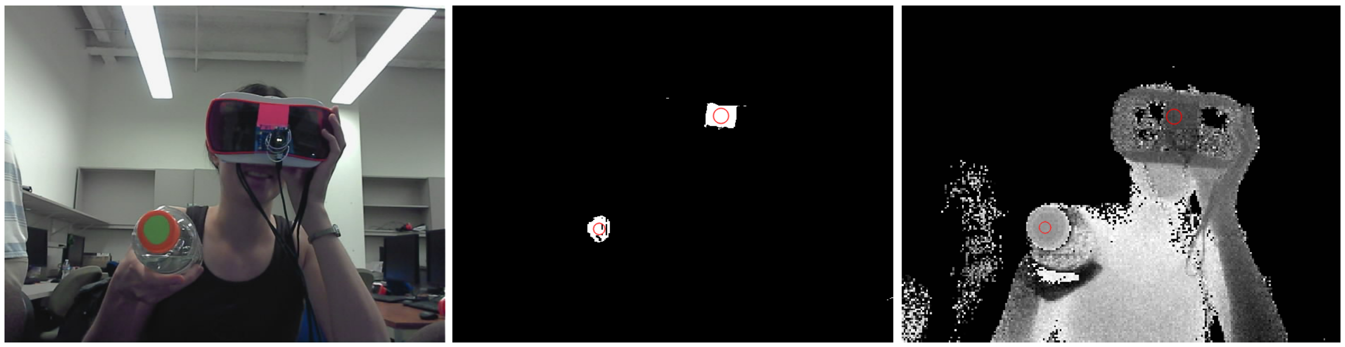

Figure 1: *Mapping markers from color image to depth image. Left: The color image. Middle: binary image that shows the markers and red circles that show the blob detection. Right: blobs mapped to depth image.*

Abstract

In virtual reality systems, the primary form of input is often the user's body positioning and gesture. The first step to processing any of this input is to detect the position of the user in the scene. Integrating work on color-based marker detection and depth imaging, we present a system that uses a combination of an RGB webcam and a time-of-flight depth camera to track markers based on color and shape. We then implemented a small painting demo to showcase these features, tracking the position of the head mounted display as well as a marker in the scene in order to allow users to draw 3D strokes in a virtual environment.

1 Introduction

A virtual reality experience is greatly enhanced when users can directly interact with and modify the world they are in. In order to allow users to manipulate virtual objects, some form of positional tracking is required to record both the users position as well as any real-world objects that may have influence on the virtual environment.

The ultimate goal with positional tracking is a system that detects the user's body and position accurately and efficiently so that the user can interact with the virtual world as seamlessly as they do the real world. However, accuracy and efficiency are often tradeoffs, where a more accurate system requires more computational power, while a fast system sacrifices precision. Given that these algorithms are implemented in virtual reality devices, speed is absolutely necessary, so our approach starts from a highly efficient base algorithm and strives to make it more accurate

Positional tracking can be approached with a variety of different techniques–often, in fact, these techniques work best when augmenting each other. There are well-established positional tracking systems using inertial [Yun et al. 2007], magnetic [Raab et al. 1979], optical, or even wi-fi [Evennou and Marx 2006] based methods, but more recently, fusing multiple streams of data from different types of sensor has proved very effective.

Our system is based largely on optical tracking, using an RGB camera to isolate markers, then a depth sensor to read in the 3D positions of the points of interest. Color-based marker detection is an extremely efficient process, but the stream of depth points output from the system was fairly inaccurate, so we applied various smoothing and filtering techniques to stabilize the final positional data.

2 Related Work

2.1 Optical Tracking

Optical tracking involves the use of cameras to stream data, which is then processed to calculate the position of a user or object in the scene. It can be further broken down into markerless or marker-based techniques.

2.1.1 Markerless

Markerless optical tracking is desirable because it requires little to no setup by the end user of the application. However, it's generally more computationally expensive [Dewaele et al. 2006], and less accurate. Often, markerless tracking requires foreknowledge of object geometry or scene structure [Simon et al. 2000], or numerous cameras with which to calculate exact positions.

2.1.2 Marker-based

Placing markers in known positions on an object (or using a marker with a known pattern [Kato and Billinghurst 1999; Claus and Fitzgibbon 2004]) is a well-established method for positional tracking. Markers may be active (LEDs) or passive (reflective stickers) [Park and l. Yoon 2006]. The system, knowing the structure of the marker at some given rest position, can use the observed orientation to calculate the current position of the tracked object. Our system uses marker-based optical tracking, as we are optimizing for both speed and accuracy. Since the range of our HMDs is fairly limited, and the application is small-scale we are not too concerned with the cost of marker setup.

2.2 Color-based Marker Detection

While many marker-based positional tracking systems rely on specific patterns within a marker or marker placements, there has been a recent push towards marker-based systems that focus on detecting a distinct color [Pylkko et al. 2001], or arrangement of colors [Jifeng Ning 2009; Gabriel et al. 2005].

While color-based systems are fast, they are extremely susceptible to changes in lighting or atmospheric effects [State et al. 1996]. In that vein, there has been significant research in recovering the exact colors of a scene from camera data [Barnard et al. 2002] and on color correction for scenes that may suffer from adverse lighting conditions [Bagherinia and Manduchi 2013].

These systems must adjust to particular lighting conditions because they rely on the specific size and orientation of the marker to produce all positional information in the scene. Our system, however, augments color-based marker detection with information from a depth camera, allowing us to separate the tasks of locating a region of interest and calculating the 3D position of that region.

2.3 Mapping Color and Depth Data

Using data from multiple sensors to increase the accuracy of position and orientation tracking has already proven to be very effective [State et al. 1996]. Recent research has applied the same principles to combining color and depth images in optical tracking [Bleiweiss and Werman 2009]. A high-resolution web-cam is relatively cheap and fairly widespread. Depth cameras, on the other hand, are more expensive, especially at higher resolutions. Bartczak proposes a system that integrates low-resolution depth data with high-resolution color images. Our application utilizes some of these techniques to output a more accurate depth measurement, despite the noise in our input depth stream.

3 Methodology

Our system combines RGB web-cam and time-of-flight depth data to track markers based on color and general shape. We implemented image filtering, based on color, blob detection to remove false positives and noise, and image warping to match the depth image to the correct color coordinates. Finally, we produced a demo application that allows the user to ”paint” virtual lines in 3D space.

3.1 Colored Markers

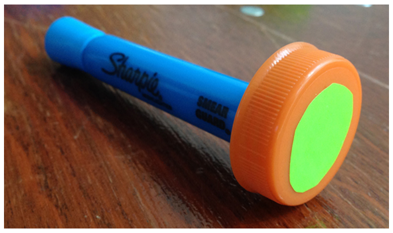

Figure 2: *The drawing tool contains a green color marker for tracking.*

We use Avery’s high visibility stickers to mark the drawing tool and the headset so the RGB webcam can easily track the objects with a color threshold. The high visibility stickers are especially bright and have completely diffuse reflection. The brightness allows the RGB to easily track the color and the diffuse reflection allows the depth camera to easily compute the depth. And specular reflection or transparency throw off the time of flight depth computation. The colors neon green and neon pink were chosen because these colors are rarely observed in clothing or in nature. The headset is covered in blue duct tape to ensure the pink marker does not conflict with the

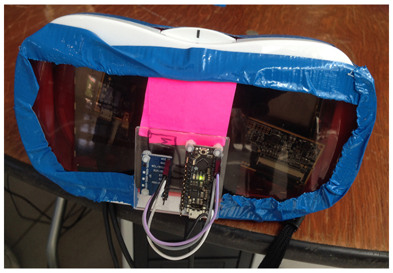

Figure 3: *The headset contains a pink color marker for tracking.*

red color on the casing. The sticker on the drawing tool is placed on a broad flat surface to reduce the effect of shading due to the curvature of the object.

3.2 Input data

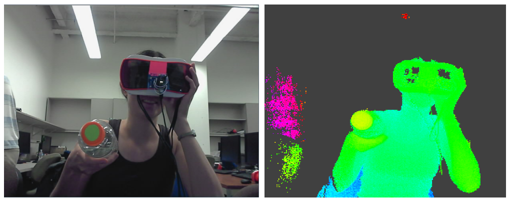

Figure 4: *Color (left) and depth (right) image from SoftKenetic DS 325 DepthSense camera.*

The input to our system is a stream of high-resolution (640x480) RGB webcam data and low-resolution (320x240) depth images from the SoftKinetic DepthSense 325. These input frames are converted from raw data into OpenCV images. The DepthSense has a frame rate of 30-60 fps, and a depth resolution of 1.4cm to 1m. It is designed for fairly close range interactions, which suits our needs.

3.3 Masking

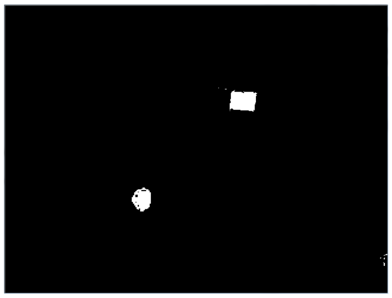

Figure 5: *Binary image showing the color image thresholded to green and pink colors.*

Using OpenCV we convert the images from RGB to HSV space. As our thresholding is color-based, HSV is optimal, as we can specify a very narrow threshold for hue, and a wider range for saturation and value, allowing for lighting differences when using the application. We picked very bright, unusual colors for our markers (neon green and neon pink) to minimize the overlap between the marker color and hues in the background. After thresholding the HSV image, the system returns a binary mask showing areas with the appropriate color. This is done once for the drawing tool and once for the headset.

3.4 Blob detection

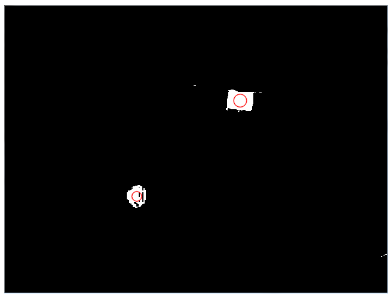

Figure 6: *Keypoints detected in the binary image.*

Though our color masking is designed to isolate the markers as much as possible, the nature of the camera and the environment often causes noise or false positives. In order to remove any pixels that may have been selected as the right color, but are not part of the true marker, we run an OpenCV blob detection algorithm on the binary mask. This locates the largest blob of white pixels within a range, allowing us to discard both tiny specks of color generated by noise, and large background patches (if a user was wearing a shirt with pink on it, for example). Finally, the system also checks the circularity of the blob, so that long stripes (probably noise) are discarded in favor of more circular shapes.

3.5 UV mapping to depth

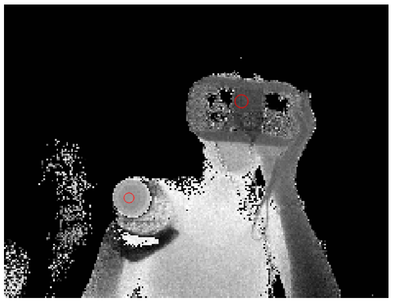

Figure 7: *Keypoints mapped to depth image.*

The streamed depth image differs from the color image in both resolution and field of view. The depth image is significantly smaller than the color image, and the bounds of the image are not perfectly aligned, so a linear scaling between the two results in incorrect depth values. We warp the depth image to fit our color data by first cropping it so that the bounds of the field of view match the color image, then upsampling the image to match the resolution of the color image.

3.6 Depth stabilization

As we are rescaling the depth image, the depth pixel at the center of the marker is fairly unstable. Simply reading in this pixel as the HMD position resulted in a large amount of jitter even when standing completely still. We implemented two methods, a straight average over the depth of every pixel in the detected marker, and a Gaussian blur over the depth pixels to filter out noise. Ultimately, we found the average provided more stable results. Additionally, if the camera lost track of the marker it would cause a highly disorienting jump in the virtual environment. We addressed this issue by doing a weighted average of the previous HDM position with the new depth position, as well as dropping any measured depth values that were clearly off-screen.

3.7 Coordinate transform

Finally, with the 3D position of both the drawing tool and the HMD, in coordinates relative to the DepthSense camera, we implemented a transform to convert the 3D position of the drawing tool into view space. This involves a translation, based on our marker-based measured position, and a rotation, based on IMU measurements. To make the user experience more interesting we also scaled the translation transform so the translation of the headset results in a more dramatic motion of the drawing tool.

3.8 Rendering

Once we have the position of the drawing tool in 3D space relative to the HMD, we visualize the path it traces using OpenGL. In particular we made a Stroke class that stored the 3D positions of the drawing tool as vertices and renders as a GL_LINE_STRIP. In order to run the Depth Sense image capture and OpenGL simultaneously, we run the Depth Sense capture on its own pthread. The IMU data stream also runs on its own thread.

4 Results

In order to visualize our positional tracking, we implemented a TiltBrush-like application that allows a user to draw a line in 3D space. We tracked and recorded the 3D position of the marker, as streamed in from the DepthSense. Using these points as vertices, we rendered a line in 3D.

We used positional data of the HMD to calculate a transformation matrix that would convert the 3D stroke points into viewspace. This matrix was used to transform the stroke, so that it would translate and rotate itself based on the movement and position of the users head.

The setup allows the user to move the drawing tool and HMD anywhere in the frustum of the DepthSense Camera up to about a meter away from the camera. It allows for a 180 degree view (or more accurately about 20,000 square degree for a hemisphere) of the drawing.

Figure 8 shows an example of a drawing from multiple angles. Notice the range of views include from the bottom of the drawing looking up Also notice the viewer can see the drawing from inside the drawing by moving closer to the camera in the z direction. The part of the line that is straight and has a high frequency likely represents an error on the marker tracking during the RGB threshold or blob detection stage when the drawing tool went out of view of the DepthSense camera.

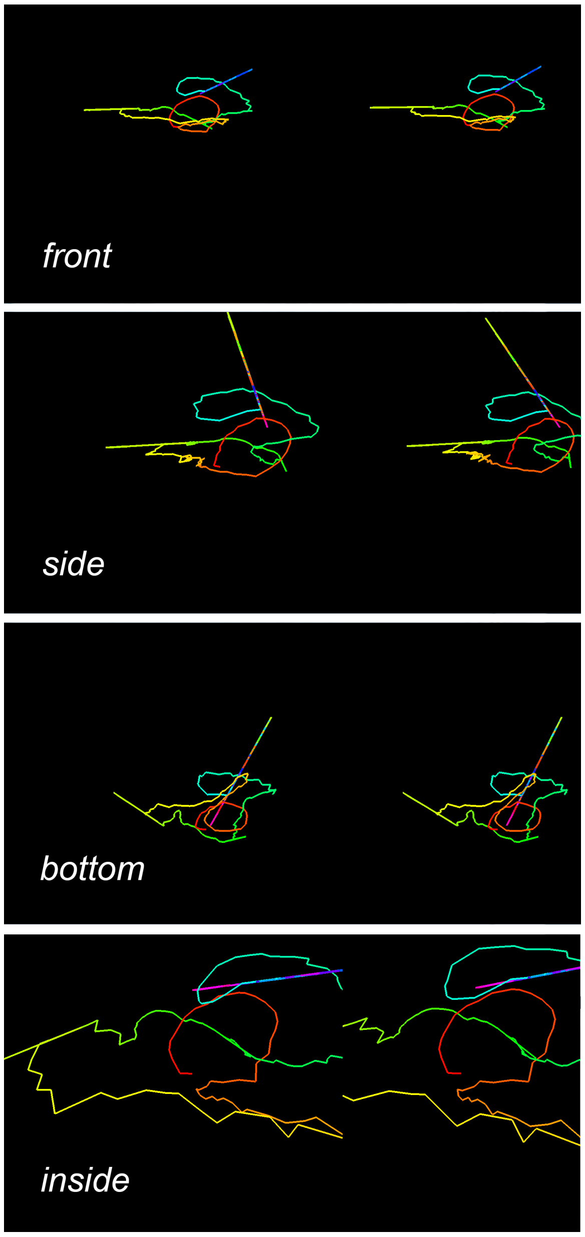

Figure 8: *HMD tracking allows the user to view a stroke traced out by our tracked marker from various angles.*

5 Discussion

5.1 Current Limitations

We are able to track a marker at high precision within the view of the DepthSense camera. However, there are several limitations to our setup that prevent a truly immersive virtual reality experience.

First, both the headset and the drawing tool must be in the view of the DepthSense camera. If the drawing tool or user's hand obscures the headset when drawing then the camera loses track of the headset.

Second, the DepthSense camera has a limited range of depth that it can detect, therefore the user must remain close to the camera, within one meter, in order to draw. Preferably the user should be seated in a chair in front of a table with the camera.

Third, the depth image has a low resolution and is noisy. This results in inaccurate positioning of the drawing tool and the headset. Figure 4 shows the depth image. Gray pixels indicate there is no depth information either because it is beyond the near and far clipping planes or because specular reflections redirect the light. There is noise on the left side of the image. The headset and hair contain a lot of specular reflection which prevents accurate depth positioning.

Fourth, in terms of usability and design, we found that users often moved the marker out of the range of the camera, or blocked the HMD marker with their hand while drawing. When the camera loses track of the marker, it works from the last known value, but this sometimes results in jittering or unstable results.

Lastly, the threshold for the color image needs to be calibrated to the lighting conditions of the room. Since we require a narrow range of color for the threshold to ensure we pick up the correct marker, a change in venue is enough to change the color of the markers such that the threshold no longer detects them.

5.2 Future Work

There are several solutions to these problems that can be attempted in a future project. To expand the space in which we can track the drawing tool, we can place a depth sense camera on the headset itself. This way the user can walk around and the drawing tool should still remain in view of the camera. To track the position of the headset, we could perform a Visual SLAM algorithm using the camera on the headset by tracking still feature in the scene, or use double integration of the accelerometer in the IMU.

A depth camera with higher resolution and less noise can reduce the amount of jitter in tracking the headset and the marker. Another way to reduce jitter in implement a more robust smoothing filter on our tracked positions, but that will introduce lag. However, in the trade-off between a lag in the display of the drawing and the smoothness and accuracy of the drawing, a user may by willing to sacrifice responsiveness for smoothness.

An additional feature that would be useful for future iterations is some kind of indication for when a marker is no longer being tracked. This approaches the jittering problem from an alternate angle, encouraging users to correct their own behavior when the camera cannot solve the problem.

Lastly, we could manipulate the depth image itself such as by blurring the entire image to reduce noise. Bartzcak and Koch [Bartczak and Koch 2009] propose a method to create a dense depth map from a low resolution time of flight depth map and high resolution color image. A dense depth map could increase the precision of our tracking.

6 Conclusion

Positional tracking is an integral part of an interactive virtual reality experience. There are many solutions to track the position of the user and objects the user manipulates simultaneously. This project attempted to use a single view color web-cam and time-of-flight depth camera. The color camera tracked colored markers in 2D image space and those pixel positions mapped to 3D positions in the depth image. This project successfully tracks the position of the headset and marker simultaneously within the constraints of the frustum of the color and depth cameras. Thus a user can draw a line in the virtual environment and view that line from multiple angles.

7 Contact Information

This project was implemented by Marianna Neubauer (mhneub@stanford.edu) and Shannon Kao (kaos@stanford.edu).

Acknowledgements

To Robert, for answering hundreds of questions during office hours, to Steve, for helping us install all the pieces of software, and to Gordon, for creating and teaching an incredibly fun class!

References

BAGHERINIA, H., AND MANDUCHI, R. 2013. Robust real-time detection of multi-color markers on a cell phone. *J. Real-Time Image Process. 8*, 2 (June), 207–223.

BARNARD, K., CARDEI, V., AND FUNT, B. 2002. A comparison of computational color constancy algorithms. i: Methodology and experiments with synthesized data. *IEEE Transactions on Image Processing 11*, 9 (Sep), 972–984.

BARTCZAK, B., AND KOCH, R. 2009. *Advances in Visual Computing: 5th International Symposium, ISVC 2009, Las Vegas, NV, USA, November 30-December 2, 2009. Proceedings, Part II*. Springer Berlin Heidelberg, Berlin, Heidelberg, ch. Dense Depth Maps from Low Resolution Time-of-Flight Depth and High Resolution Color Views, 228–239.

BLEIWEISS, A., AND WERMAN, M. Fusing time-of-flight depth and color for real-time segmentation and tracking. *School of Computer Science The Hebrew University of Jerusalem*.

BLEIWEISS, A., AND WERMAN, M. 2009. Fusing time-of-flight depth and color for real-time segmentation and tracking. In *Proceedings of the DAGM 2009 Workshop on Dynamic 3D Imaging*, Springer-Verlag, Berlin, Heidelberg, Dyn3D '09, 58–69.

CLAUS, D., AND FITZGIBBON, A. W. 2004. *Computer Vision - ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV*. Springer Berlin Heidelberg, Berlin, Heidelberg, ch. Reliable Fiducial Detection in Natural Scenes, 469–480.

Depthsense sdk download. `http://www.softkinetic.com/language/fr-BE/Support/Download/EntryId/417`. Accessed: 2016-05-26.

DEWAELE, G., DEVERNAY, F., HORAUD, R., AND FORBES, F. 2006. The alignment between 3-d data and articulated shapes with bending surfaces. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part III*, Springer-Verlag, Berlin, Heidelberg, ECCV'06, 578–591.

EVENNOU, F., AND MARX, F. 2006. Advanced integration of wifi and inertial navigation systems for indoor mobile positioning. *EURASIP J. Appl. Signal Process. 2006* (Jan.), 164–164.

GABRIEL, P., HAYET, J. B., PIATER, J., AND VERLY, J. 2005. Object tracking using color interest points. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, 159–164.

High visibility labels. `http://www.avery.com/avery/en_US/Products/Labels/Addressing-Labels/High-Visibility-Labels_05978.htm`. Accessed: 2016-06-01.

JIFENG NING, LEI ZHANG, D. Z. C. W. 2009. Robust object tracking using joint color-texture histogram. *International Journal of Pattern Recognition and Artificial Intelligence 23*, 7, 1245–1263.

KATO, H., AND BILLINGHURST, M. 1999. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on*, 85–94.

PARK, J., AND L. YOON, Y. 2006. Led-glove based interactions in multi-modal displays for teleconferencing. In *16th International Conference on Artificial Reality and Telexistence–Workshops (ICAT'06)*, 395–399.

PYLKKO, H., RIEKKI, J., AND RONING, J. 2001. Real-time color-based tracking via a marker interface. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, 1214–1219 vol.2.

RAAB, F. H., BLOOD, E. B., STEINER, T. O., AND JONES, H. R. 1979. Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic Systems AES-15*, 5 (Sept), 709–718.

SIMON, G., FITZGIBBON, A. W., AND ZISSERMAN, A. 2000. Markerless tracking using planar structures in the scene. In *Augmented Reality, 2000. (ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, 120–128.

STATE, A., HIROTA, G., CHEN, D. T., GARRETT, W. F., AND LIVINGSTON, M. A. 1996. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '96, 429–438.

TAKALA, V., AND PIETIKAINEN, M. 2007. Multi-object tracking using color, texture and motion. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–7.

YUN, X., BACHMANN, E. R., MOORE, H., AND CALUSDIAN, J. 2007. Self-contained position tracking of human movement using small inertial/magnetic sensor modules. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2526–2533.