The International Journal of Virtual Reality

Towards Achieving Robust Video Self-avatars under Flexible Environment Conditions

Loren Puchalla Fiore [1], Victoria Interrante [1]

1 Department of Computer Science, University of Minnesota

***Abstract*—The user's sense of presence within a virtual environment is very important as it affects their propensity to experience the virtual world as if it were real. A common method of immersion is to use a head-mounted display (HMD) which gives the user a stereoscopic view of the virtual world encompassing their entire field of vision. However, the disadvantage to using an HMD is that the user's view of the real world is completely blocked including the view of his or her own body, thereby removing any sense of embodiment in the virtual world. Without a body, the user is left feeling that they are merely observing a virtual world, rather than experiencing it. We propose using a video-based see-thru HMD (VSTHMD) to capture video of the view of the real-world and then segment the user's body from that video and composite it into the virtual environment. We have developed a VSTHMD using commercial-off-the-shelf components, and have developed a preliminary algorithm to segment and composite the user's arms and hands. This algorithm works by building probabilistic models of the appearance of the room within which the VSTHMD is used, and the user's body. These are then used to classify input video pixels in real-time into foreground and background layers. The approach has promise, but additional measures need to be taken to more robustly handle situations in which the background contains skin-colored objects such as wooden doors. We propose several methods to eliminate such false positives, and discuss the initial results of using the 3D data from a Kinect to identify false positives.**

***Index Terms*—off-the-shelf virtual reality, self-avatar, video-see-thru head mounted display**

I. INTRODUCTION

Virtual environments have become very realistic in recent years, especially those experienced through head mounted displays (HMDs). However, a major disadvantage of HMDs is that any view of the real world is completely occluded, including the user's view of his or her own body. This can be quite disorienting when navigating a virtual environment. One solution is to track the user's body and create a virtual avatar within the virtual environment. This allows the user to see a generic body in place of his or her own and generally reduces disorientation, and increases presence as shown by Slater and Usoh in [1]. This works well for some situations, but it is still not the user's own body, and thus is not as realistic as it could be. Furthermore, in some situations this can hurt the sense of presence more than if no body was rendered, for instance if the body is retargeted incorrectly resulting in appendages that are out of place or proportioned incorrectly. When interacting in immersive virtual environments several groups have shown that even a partial avatar embodiment can enable enhanced task performance within the environment [2], [3], [4]. It has also been shown, by Ries *et al.* in [5], that the avatar fidelity and quality can have an effect on task performance. We would like to determine if a self-avatar, one that is a near-perfect replica of the user, increases task performance even further.

The purpose of this research is to use a video see-thru HMD (VSTHMD), which has cameras mounted to the outer-front of the visor, to capture video of the real-world as it would be seen by the user wearing the display. Then, using computer vision techniques, we will segment the user's body (hands, feet, arms, and legs) from the captured video and composite them into the rendering of the virtual environment. This will allow the user to see his or her own body within the virtual environment and will, in theory, increase the sense of presence the user feels within such an environment [1]. Also, since we will have separate video for each eye, if we can locate the user's hands within the video from each camera we can use stereo vision to determine their approximate location in three dimensions. By using the segmentation of the user's hands to determine stereo we essentially side-step the general form of the stereo correspondence problem. This localization would afford interaction within the virtual environment without the need for expensive hand tracking systems.

There are a few problems, however, keeping us from jumping right in and studying interaction with such a VSTHMD. Currently, commercially available video-based see-thru HMDs are prohibitively expensive, costing tens of thousands of dollars. Simple HMDs however are relatively cheap because of their applicability as a portable television, see for example the Sony HMZ-T1 which costs \$800. Because of this cost differential, we first investigated building our own VSTHMD using the HMD our lab already owned. Once we have a VSTHMD, we must develop an algorithm that is capable of robustly identifying pixels belonging to the user's body in the video streams coming from the cameras in the head-mounted display. This is a generalized form of the background/foreground segmentation problem from computer vision. Our case is interesting because both the camera, attached to the user's head, and the foreground consisting of

Manuscript received on DD MM YY.
fiore@cs.umn.edu

the user's body are moving arbitrarily in 3D, which makes the problem inherently intractable unless additional information can be gathered. Thankfully, there are several assumptions we can make which simplify the problem. These design considerations, the assumptions we can make, and the hardware development of our VSTHMD will be discussed in the Assumptions & Hardware section.

Next, we would like to take some time to discuss the related prior work, before moving on to an overview of our assumptions, hardware, and design considerations. After that, we will look at preliminary results we have obtained using multiple histogram modeling of foreground and background color distributions. Finally, we will discuss problem areas of our current design, what we are working on to correct these problems, and our future goals once these have been eliminated.

II. Related Work

Several researchers have previously addressed the problem of how to provide people with a robust, video-based self-avatar in an HMD-based immersive virtual environment. Petit *et al.* in [6] propose a solution that uses multiple cameras to capture video images of the user from an external point of view along with a green-screen to simplify the process of background subtraction. Bruder *et al.* in [7] use a head-mounted camera and background subtraction via color-segmentation in order to give the user a self-avatar of their arms and legs. Our solution is inspired by the latter approach, and seeks to extend it by considering more robust methods for modeling the differences in background and foreground color. Our end goal is a system that can work in any room, regardless of the color, shape, or layout of that room, by using an offline room calibration step.

In general, foreground segmentation can be thought of as finding foreground and background images such that when combined using a binary mask they form the image captured from the camera. There has been an enormous amount of work in the area within the image processing and computer vision communities. The work can be roughly divided into those that use a single camera, or multiple cameras, and then further divided based on the assumptions on foreground movement relative to the background. The largest area of research is from a single camera, and when the background is stationary and the foreground is moving. These assumptions are particularly well-suited for the task of surveillance and traffic monitoring, which were studied by Stauffer and Grimson in [8] when they developed their Gaussian mixture model approach to foreground segmentation. Many authors have research enhancements to this model, such as Zivkovic [9] or Lee [10] who both propose improved Gaussian mixture models which can compute many of the parameters automatically from the video instead of from offline user input. While these methods are very popular, and work quite well when their assumptions are met, they have shown in testing to not work well for our needs because it is the case that quite often the background in our video is moving and the foreground is stationary.

Depth is also another widely used cue in foreground segmentation. In [11], Harville *et al.* extend the standard RGB colorspace of the Gaussian mixture model with a fourth channel of depth information which they use to enhance the results of the segmentation. Another interesting approach, using graph-cuts, is found in [12] where Kolmogorov *et al.* are able to segment video into foreground and background layers for use in video conferencing. Both of these systems are seen as potential future work to adapt to our system.

When both the background and the foreground are moving, as is often the case when the camera is not stationary, there is little we can do to recover the foreground and background layers. Both Hayman *et al.* in [13] and Sheikh *et al.* in [14] attempt to tackle this problem by using optical flow to identify the type and amount of camera movement, and then compensate in order to identify motion that is not caused by the camera motion. These results are interesting, but still assume that the foreground is moving which is not something that we can always be assured of in our setting.

III. Assumptions & Hardware

As discussed, when the camera and the user (*i.e.* the foreground) are both moving arbitrarily there is very little in general that we can be sure about. Thankfully, however, there are some assumptions that we can make. The first assumption is that this VR system will always be used in a room that we know about ahead of time. This allows us to use an offline training phase where we can learn various image and geometric statistics about the room. The second assumption that we make is the user's head, and by extension the HMD and cameras attached to it, are tracked with 6 degrees of freedom within this room. Since this technology is to be used along with an immersive virtual environment which requires this tracking, it is a reasonable assumption. The third assumption that we can acquire some information about the user in advance of them using the system. Currently we do this by taking a few seconds of video of the user looking at their hands and feet while wearing the VSTHMD. This video is then used in a short calibration step prior to the system use. In the examples shown below the virtual environment we use is a replica of our real-world lab rendered in a non-photorealistic black-and-white style.

In our lab we already have an SX60 HMD manufactured by NVIS. This HMD can display 1280x1024 resolution stereoscopic images to the user using two OLED displays, and has a FOV of 60 degrees diagonally. We therefore wanted to find cameras which could match these display properties. What we settled on was the Logitech C615 USB webcam, which can capture video at 1920x1080, at 24fps, and has a FOV of 74 degrees diagonally. Two of these webcams are mounted onto the SX60 HMD using velcro, which has been attached to the webcams with hot melt glue. The glue is applied to the back of the webcams, and then sanded down in order to give a smooth surface with which to apply the velcro and also act as an alignment guide to aim the webcams. The final result is shown in Fig. 1(a) with detail of the attachment shown in Fig. 1(b). The images are captured from the cameras using the OpenCV library over USB 2.0. Once in our program, they are cropped and resized and rendered onto an OpenGL quad to display on the HMD screens. The exact amount of

cropping is determined via overlaying the video with a rendering of our lab room model and then adjusting by hand until the images match.

The webcams themselves have a fairly short depth-of-field, and as a result if the user looks rapidly around the room the image will become blurred while the camera autofocus attempts to adjust to the scene. Using a camera with a different lens, and a camera with a faster shutter and auto-focus response would alleviate these problems, but the additional weight and size would make mounting the cameras aligned with the eyes and wearing them more difficult. A system of mirrors, as demonstrated by State *et al.* in [15], would be one way to incorporate larger cameras without negatively affecting the weight distribution and alignment of the cameras.

(a) COTS VSTHMD (b) COTS VSTHMD attachment detail.

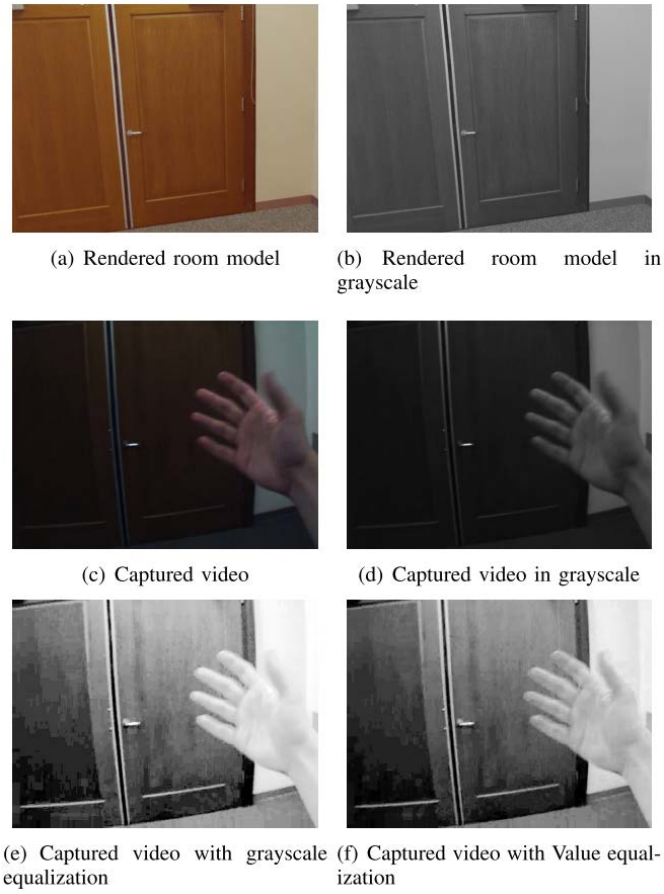
Fig. 1. The prototype Commercial-off-the-shelf Video See-Thru HMD built using two USB webcams attached to our existing NVIS SX60 HMD.

The velcro was used to attach the webcams in the hope that they could be moved to adjust for different interpupillary distances between users. However, using velcro has made it difficult to align the cameras with the lab room model for longer than a few hours of use, since they have a tendency to move slightly during motion of the user's head. The development of a different attachment mechanism is therefore a highly desired future project.

IV. ALGORITHM

4.1 Image Differencing

The first method we tried was one of simple frame differencing between a rendered scene of the 3D model of our room and the live video. The motivation for this is that any difference between the rendered room model and the video will show up as foreground. As a result, anything not included precisely in the room model, such as chairs and desks, will show as foreground. To work around this we could in the future use a Microsoft Kinect, or similar device, to obtain a room model that includes the furniture instead of the hand-built model we are currently using. The difference was taken as the absolute value of the rendered image subtracted from the video image in the grayscale color space. A threshold was then applied and any pixels with an error larger than this threshold were marked as foreground pixels. Histogram equalization was used to normalize the range of values between the two images. The histogram equalization was tested both in the grayscale space, and in the Value channel (of HSV) before conversion to grayscale. By performing the equalization on Value alone, the hope was that the Hue would remain the same which cannot be guaranteed if the equalization is done in grayscale. We show in Fig. 2 the result of both types of equalization to a sample image. Median filtering was done after the difference was computed in an attempt to reduce the number of spuriously labeled foreground pixels. A result of this method is shown in Fig. 3. This figure shows the image from the rendered 3D model of the room along with the input video frame, and the virtual environment that we wish to composite the user into. The results show the gray mask before thresholding, and the result after thresholding the gray mask and composition. Notice that the wrist is missing from the final composition because a threshold low enough to include the wrist would also include large sections of the door since the door appears brighter in the gray mask than the wrist area.



(a) Rendered room model (b) Rendered room model in grayscale

(c) Captured video (d) Captured video in grayscale

(e) Captured video with grayscale equalization (f) Captured video with Value equalization

Fig. 2. The results of histogram equalization for the purposes of contrast and brightness correction. The histogram equalization is shown here performed in grayscale (e) and on the Value channel of an HSV image (f). In both cases, the equalized video more closely matches the rendered image of the room model, which allows us to easily compare the two images.

The results of this were promising, as the hand and several objects in the room, such as chairs, which were not present in the 3D model, were roughly segmented correctly. However, areas where the lighting in the 3D model differed significantly from the real room proved troublesome as well as colors that would map to similar grayscale values. Another problem with this method, and with any method relying on our 3D model of

the room, is the alignment between our rendering of the model and the live video. Currently we are just relying on the hardware design of the VSTHMD to align the video, and are not performing any rectification or additional alignment in

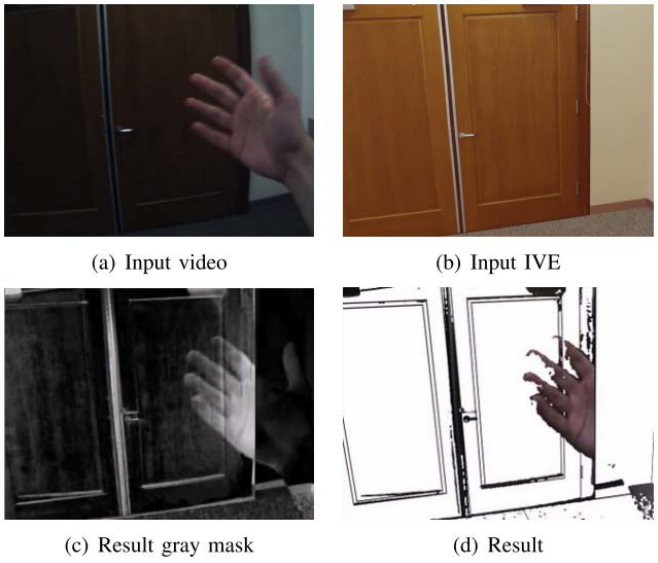


Fig. 3. The results of simple frame differencing when histogram equalization is performed on the Value channel of the input video. The hand and wrist are lost because the intensity difference between the hand and the wall is too low relative to the other differences in the image due to lighting inconsistencies, as well as the fact that the comparison is done in grayscale instead of full RGB colorspace.

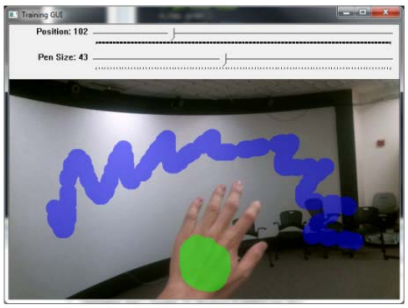


Fig. 4. Foreground and background color calibration for a single background histogram and single foreground histogram model. The background is labeled as blue pixels, the foreground as green pixels.

software. There is also the issue of temporal alignment to deal with as well. Since the tracker updates at close to 1000fps, and the camera only updates at 20-30fps, the position where we render the room is not synchronized with when we capture the frame from the camera. Because of the difference in update rates the quality of the segmentation will suffer when the viewing direction is changing rapidly. Managing this temporal alignment by delaying or queuing the tracker data would help alleviate this problem to some extent, but we currently make no such adjustments. We found that frame differencing is only a partially successful solution to the problem which is why we needed to turn to the more involved approach inspired by the work of Bruder *et al.* [7] discussed in the next section.

4.2 Color Classification

4.2.1 Single Background Histogram

Using our COTS VSTHMD we next investigated segmentation based on skin color. We first implemented the algorithm of [16] which creates RGB histograms of the foreground (skin) and background (everything else) in an offline training phase. The foreground and back histograms are normalized to give estimates of the probability distributions $P(x|skin)$ and $P(x|\neg skin)$. These distributions, along the with probability of skin or not skin in the training data, are then used at run-time to compute the probability that a given pixel x corresponds to skin using,

$$P(skin) = \frac{P(x|skin)P(skin)}{P(x|skin)P(skin) + P(x|\neg skin)P(\neg skin)}.$$

Pixels with a probability above a threshold θ are marked as being skin. We also looked at Hue/Saturation (HS) histograms, as well as Hue-only (H) histograms for classification. The algorithm remained the same in these cases, only the color and histogram space were changed.

The first step is to obtain training pixels that can be used to create the histograms needed for classification. To this end we developed a graphical user interface, shown in Fig. 4, which allows a user to scan through a video file and highlight skin and background pixels by drawing on them with the mouse. These pixel labelings are then used to create the foreground and background histograms.

Results from this color based segmentation for two different frames of video are shown in Fig. 5. These figures show the raw masks for a threshold of $\theta = 0.4$. Also shown is the result of applying a morphological hole closing operation to the mask, in order to fill in any gaps that may be present in the mask in an attempt to create a nicer looking mask. In our trials, RGB histograms performed the best with HS histograms a close second. The reduction to only Hue appeared to be too much, as background pixels were very often misclassified as skin. The HS histograms use less memory than the RGB histograms, yet appear to be very similar in classification accuracy. This might be an important fact to consider if we attempt GPU optimization in the future, and want to store and access these histograms on the GPU itself.

4.2.2 Multiple Background Histograms

The case shown on the right of Fig. 5 shows when the user looks at the doors to our lab. Since the doors are wooden, they match closely the color of the user's hand and as a result the segmentation contains many false negatives. What we are currently investigating is the use of multiple background histograms each stored according to the direction the user is facing. The run-time histogram is then interpolated from the stored background histograms using the user's current facing direction to determine the interpolation weights. In order to learn these background histograms, we use a captured video of the entire room. We also capture and store the tracker data simultaneously during this process. In order to make sure each area of the room is equally weighted, we first build a spherical

environment map from the captured video and tracker data as shown in Fig. 6. The environment map is then sectioned into multiple regions, one for each of the background histograms. The current sectioning we are using is based on the vertices of a regular dodecahedron and is shown in Fig. 7, however we are also investigating other sectioning that places more regions along the 0 degree elevation since that is where the majority of use will occur. As can be seen in Fig. 8, this method produces superior results to the single histogram case when run on the same input.

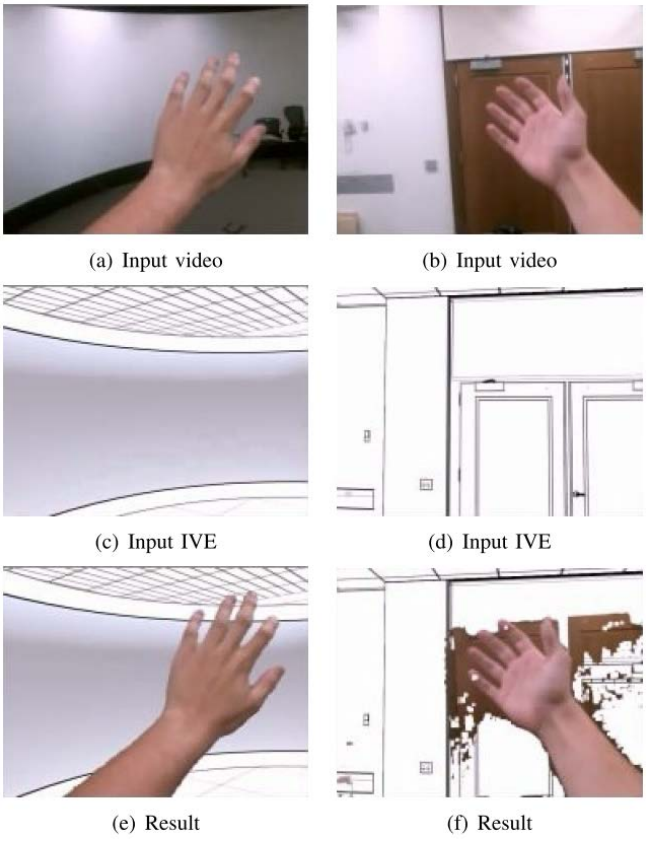


Fig. 5. Results of segmentation using the single background histogram and single foreground histogram model. Morphological closure was performed on the binary masks in order to reduce holes in the segmented regions. The result (b, d, f) shows numerous false positives due to the wooden door in the background.

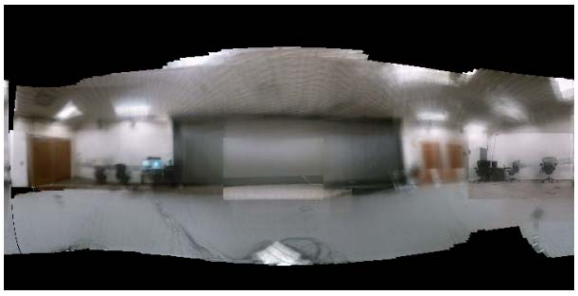


Fig. 6. Spherical environment map of our lab created from a tracked video capture. No video frames were captures for the regions at ±90 degrees elevation in this particular video which leaves black regions at the top and bottom of the map.

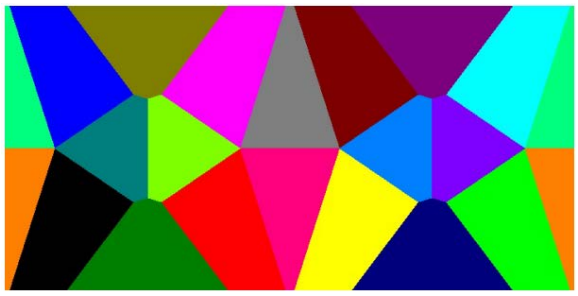


Fig. 7. The prototype Commercial-off-the-shelf Video See-Thru HMD built using two USB webcams attached to our existing NVIS SX60 HMD.

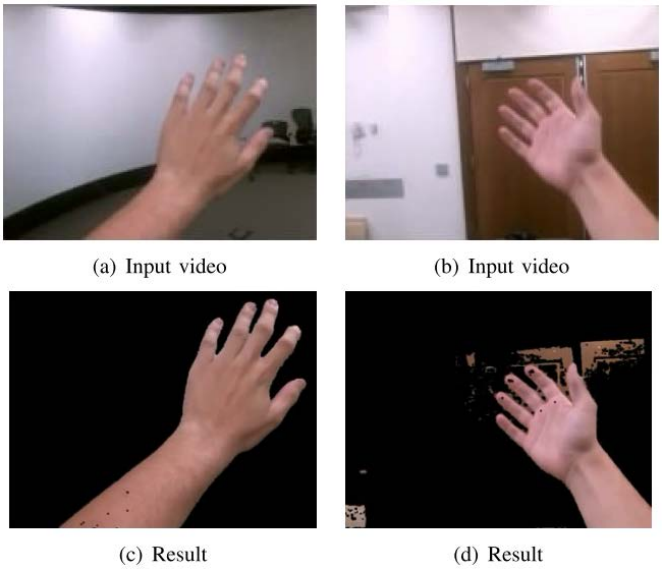


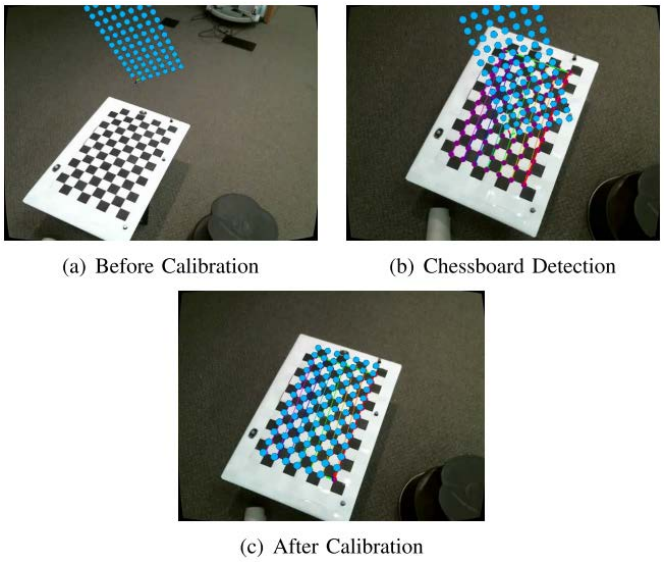
Fig. 8. Results of using a multiple background histogram model on the same input video as Fig. 5. No IVE was rendered in this test, to better show the segmented regions.

V. KINECT SENSOR INTEGRATION

Up until this point our investigations have all been using the images from the HMD cameras and segmenting them using image features alone. This works well in some cases, but it is likely that some false positives will always remain when using video features alone. Also, some clothing will have too many colors to produce a usable foreground model. In these situations we propose to use one or more Microsoft Kinects, mounted around the room or potentially on the HMD similar to Suma *et al.* in [17], to get a rough estimate of the user's body location in 3D. This 3D location can then be used to facilitate the segmentation process. This next section discusses our work on performing implementing such a system using a single Kinect external to the user. This skeleton and depth information from the Kinect is processed in order to determine which sections of the image possibly contain the user's body as a way to filter spuriously segmented pixels.

5.1 Setup & Calibration

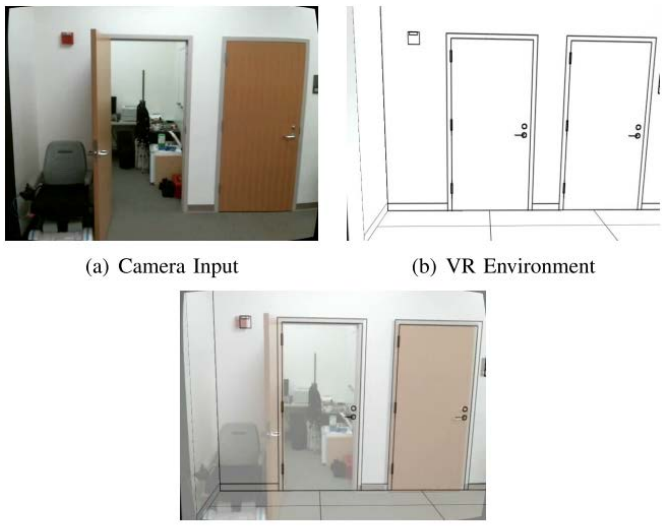
For this series of experiments we have used the Vicon tracking system in our lab. It consists of 12 Vicon MX cameras that provide six degree-of-freedom tracking of the objects needed for the experiment. We track the HMD, the Kinect, and a large chessboard pattern which can be seen in Fig. 9. We are using Microsoft Kinect SDK v1.5 for these experiments which gives us skeletal and depth data in meters with respect to the Kinect's coordinate frame. The tracking via the Vicon allows us to transform this Kinect data into our virtual environment's world space.



(a) Before Calibration (b) Chessboard Detection

(c) After Calibration

Fig. 9. These images show the calibration process of the HMD cameras extrinsic parameters. (a) shows what the user sees before the calibration, not that the grid of spheres does not align with the chessboard. (b) shows the overlay generated by OpenCV when it detects the chessboard pattern, the circles show the reprojected corner points as a sanity check. (c) shows the alignment after calibration which is triggered by pressing a key on the computer's keyboard once OpenCV detects the chessboard. The spheres are in the world coordinate space and move with the chessboard based on the Vicon tracking data. When they are aligned the user knows that the calibration was successful.



(a) Camera Input (b) VR Environment

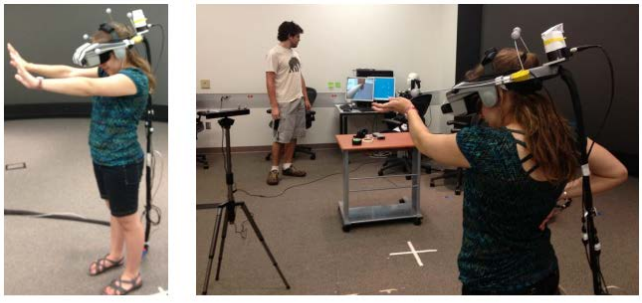
(c) Overlay

Fig. 10. Here is shown (a) the image captured by one of the HMD cameras, (b) the rendering of the virtual environment which in this case is a non-photo realistic model of our room, and (c) an alpha-blended overlay of the two images to show how well they align after the camera intrinsic and extrinsic parameters have been calibrated.

The critical step to making the Kinect, Vicon, HMD, and cameras perform well together is to calibrate everything into a global world frame. The Vicon tracker space and the room were calibrated together upon installation, so we need not worry about this. The transformation from the Kinect to the world was manually chosen by selecting a point near the Kinect's depth camera origin. For the cameras mounted on the HMD we need to calibrate the camera intrinsic parameters (focal length, principal point, aspect ratio) and extrinsic parameters (rotation and translation in world frame). The intrinsic parameters were calibrated using multiple images of a chessboard pattern that was processed using the OpenCV library. The chessboard corners were reprojected into the image and as a sanity check to ensure the intrinsic parameters found were valid. This was done for each camera separately. As a result of the calibration, OpenCV also gives a transformation ${}^{E_i}_{C}T$ from the chessboard C to the camera, or eye, coordinate frame E_i. Using the Vicon we also have transformations from the chessboard to the world, ${}^{W}_{C}T$, and from the HMD to the world, ${}^{W}_{H}T$. Together we can then find the location of the camera origin with respect to the HMD,

$$ {}^{H}_{E_i}T = inv({}^{W}_{H}T) \cdot {}^{W}_{C}T \cdot inv({}^{E_i}_{C}T) $$

where $inv(\cdot)$ is the operation which reverses the direction of the transformation (*i.e.* $inv({}^{H}_{W}T) = {}^{W}_{H}T$). This allows us to compute the camera positions very precisely at every frame. Using these intrinsic and extrinsic parameters we can then compute OpenGL projection and worldview matrices which will render the virtual environment to match what is seen from the camera, similar to how it is done in augmented reality. The calibration process of the extrinsic parameters can be seen in Fig. 9, and results of the alignment between the camera and the virtual environment can be seen in Fig. 10.

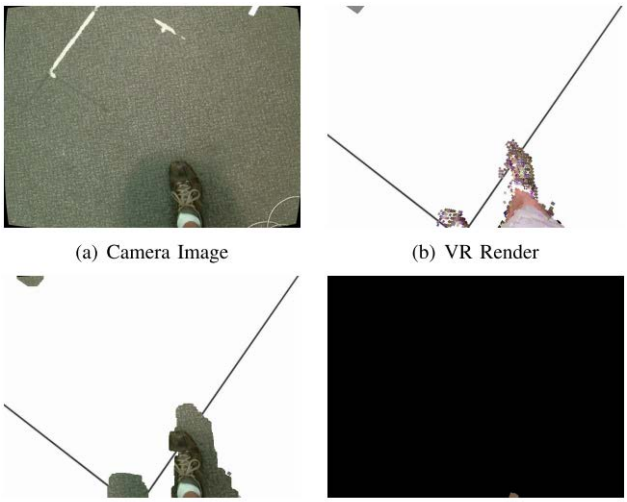


(a) HMD with camera detail (b) User facing Kinect

Fig. 11. The user must stand within the Kinect's field of view in order for the segmentation to be effective. Currently we have a single Kinect, as shown in this figure, but eventually we hope to use this system with multiple Kinects which collectively cover the volume of the room.

(a) Kinect Data (b) Expanded Mask

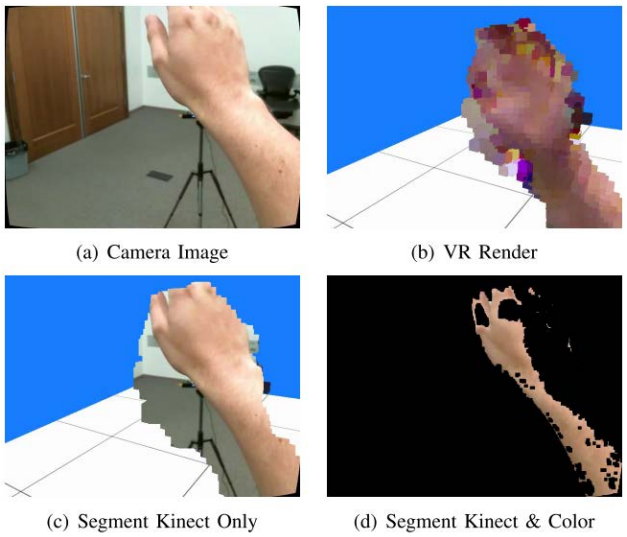
Fig. 12. To generate the segmentation mask from the Kinect data we take the pixels rendered by the data and expand the region by performing a dilation operation. The result of such an operation is shown here on a frame looking at the user's feet.



(a) Camera Image (b) VR Render

(c) Segment Kinect Only (d) Segment Kinect & Color

Fig. 13. Here we show the results when the depth image is used to segment the user. Notice that the color segmentation does not show the feet, because they are not skin colored, but the kinect can find the feet.



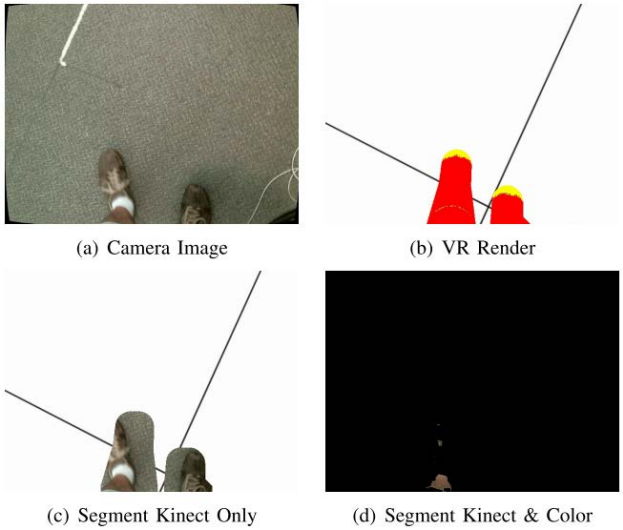
(a) Camera Image (b) VR Render

(c) Segment Kinect Only (d) Segment Kinect & Color

Fig. 14. Here we show results when the depth image is used to segment the user. Because of the additional kinect data, none of the door pixels are selected as foreground. The arm is oversatured because the user is standing beneath the ceiling lights, so parts of the arm are misclassified.

5.2 Usage & Results

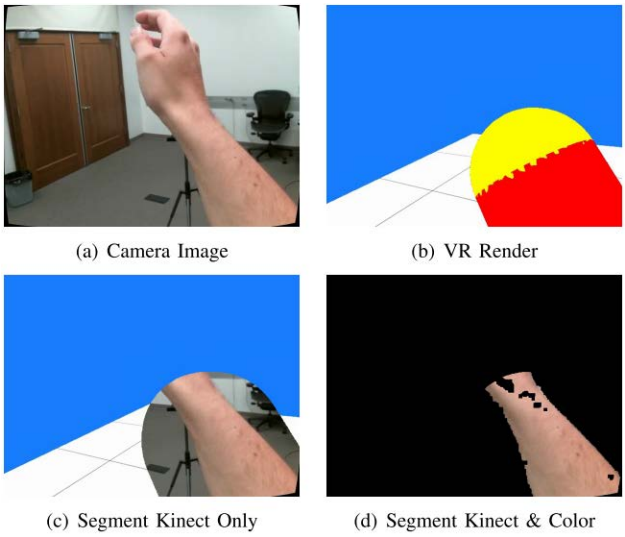
Our eventual goal is to have multiple Kinect sensors which can cover the entirety of our room, but for now we have a single Kinect which covers approximately a $6m^2$ area. As such the user needs to stand in front of and facing the Kinect as shown in Fig. 11. Once we have the system properly calibrated we can then transform the Kinect data into the world space and render it in our virtual environment. We can render the skeleton as a series of cylinders, as shown in Figs. 15 and 16, or render the raw color pixels at their appropriate depth as shown in Figs. 14 and 13. Once we have these rendered we can also segment the portion of the rendered image corresponding to the Kinect data and replace it with the images captured from the cameras. We expand the area rendered from the Kinect data in order to create the segmentation mask to try and capture the entirety of the user's body. The result of this is shown in Fig. 12. We have also shown in the figures the results when the camera image data is segmented further using the color histogram methods discussed earlier in this paper. This focuses the color segmentation only onto the areas that the Kinect has identified as containing the user's body, thereby eliminating many of the spurious pixels caused by the doors. This color histogram segmentation shown in the figures was computed in a post-process offline step currently because we have multiple codebases which have not yet been merged.



(a) Camera Image (b) VR Render

(c) Segment Kinect Only (d) Segment Kinect & Color

Fig. 15. Here we show the results when the skeleton data is used to segment the user. Notice that the color segmentation does not show the feet, because they are not skin colored, but the kinect can find the feet.



(a) Camera Image (b) VR Render

(c) Segment Kinect Only (d) Segment Kinect & Color

Fig. 16. Here we show results when the skeleton data is used to segment the user. Because of the additional kinect data, none of the door pixels are selected as foreground.

5.3 Discussion

The rendering of the hands and feet, since they are very close the cameras, is extremely sensitive to any errors in the calibration of the system. As a result of this, as it can be seen in the result figures, the entire hand or foot in the camera image is never perfectly aligned with the Kinect rendering. This is most likely because of our current manual calibration of the Kinect's coordinate frame origin with respect to the Vicon. An area of future research is to develop a method similar to the one used for the HMD cameras to calibrate the Kinect precisely with respect to the Vicon coordinate frame. Another interesting area of future research is the selective enabling of the color segmentation. For instance when the user is looking at his or her feet, the Kinect data alone is almost enough to correctly segment the legs and feet without the need of color segmentation. Also, because of the numerous colors in the clothing, the color segmentation of the feet and legs would become very complex. This could be skipped by relying on the Kinect for the legs and feet.

VI. Future Directions

Even with the work done so far, there is much left to do. Here we discuss what we are currently working on, and where we would like this research to progress after that.

6.1 False Positives

With the use of multiple background histograms, the false positive detection rate dropped considerably, however there were still some misclassified regions. One possible way to fix this that we are currently looking at is to use a Gaussian mixture model to model the background probabilities instead of a histogram. This would remove any aliasing artifacts caused by the fixed number and fixed width of the histogram bins currently used. The next thing we are looking at is using Gabor filters (similar to [18]) in order to model some texture information along with RGB intensities in our foreground and background models.

6.2 Clothing Detection

Our current approach works well when the user is wearing t-shirts and shorts, but does not work when clothing is visible in the image. We would like to compensate for this by including clothing in the foreground model. This would be done by having the user stand against a white wall, and take video from the VSTHMD of the entire user's body instead of just their arms and legs. This is something we want to investigate after the addition of texture features, since the addition of clothing colors could create more overlap between foreground and background models without texture.

6.3 User Studies & Applications

Once we have a system that can robustly give the user a self-avatar we would like to investigate the effects of this self-avatar on the realism and sense of presence the user experiences within the virtual environment as compared to tracker-based 3d model self-avatars. The first study would be to test users' distance perception without a self-avatar, with a 3d self-avatar, and with a video self-avatar. It has been shown in [2] that the quality of the 3D self-avatar may affect the user's distance perception accuracy. The next logical step after this would be to study user's ability to interact with the virtual world using their video self-avatar.

VII. Conclusion

In conclusion, we have shown several attempts at detecting the user's body from the video obtained while wearing a video see-thru head mounted display. We have seen that simple frame differencing and Gaussian mixture model approaches using a 3D room model can somewhat work, under restricted conditions, but have noisy output. Depth information should be theoretically possible to obtain since we have two cameras, however our room with its uniform colors and textures has proven difficult for block matching stereo correspondence algorithms. This results in nearly unusable depth information for most frames. We have also discussed the creation of a commercial-off-the-shelf (COTS) VSTHMD, using HD resolution webcams. Using the VSTHMD we developed, we looked at using skin color to segment the user's hands and arms. We have seen that using histograms of known skin and background pixels (trained in an offline setup phase) can produce very good results when no wooden objects are also in the frame. Using multiple location-aware background histograms reduces the number of false positives, but they still remain in some areas. The elimination of these false positives due to wooden objects is an area of future research that we hope to solve using additional texture features, and more accurate Gaussian mixture models of the probability distributions. We have also investigated the use of the Microsoft Kinect as a way to obtain an estimate of the portions

of the camera images containing the user, thereby eliminating the spurious pixels of similarly skin-colored objects in the room. When the spurious pixels are near the user's body in the image the Kinect data does not help us, but when they occur of opposite sides it can ignore non-user regions.

REFERENCES

[1] M. Slater and M. Usoh, "Body Centered Interaction in Immersive Virtual Environments," *Artificial Life and Virtual Reality*, N. Magnenat Thalmann and D. Thalmann, Eds. John Wiley and Sons, 1994, pp. 125-148

[2] L. Phillips, B. Ries, M. Kaeding, and V. Interrante, "Avatar self-embodiment enhances distance perception accuracy in non-photorealistic immersive virtual environments," *Proceedings of the 2010 IEEE Virtual Reality Conference (VR)*, pp. 115-118, Mar. 2010.

[3] B. Lok, S. Naik, M. Whitton, and F. Brooks, "Effects of handling real objects and self-avatar fidelity on cognitive task performance and sense of presence in virtual environments," *Presence: Teleoperators & Virtual Environments*, vol. 12, no. 6, pp. 615-628, 2003.

[4] B. J. Mohler, S. H. Creem-Regehr, W. B. Thompson, and H. H. Bülthoff, "The Effect of Viewing a Self-Avatar on Distance Judgements in an HMD-Based Virtual Environment," *Presence: Teleoperators and Virtual Environments*, vol. 19, no. 3, pp. 230-242, Jun. 2010

[5] B. Ries, V. Interrante, M. Kaeding, and L. Phillips, "Analyzing the effect of a virtual avatar's geometric and motion fidelity on ego-centric spatial perception in immersive virtual environments," *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology (VRST '09)*, vol. 1, no. 212, pp.59-66, 2009.

[6] B. Petit, J. D. Lesage, C. Menier, J. Allard, J. S. Franco, B. Raffin, E. Boyer, and F. Faure, "Multicamera Real-Time 3D Modeling for Telepresence and Remote Collaboration," *International Journal of Digital Multimedia Broadcasting*, vol. 2012, pp. 1-12, 2010.

[7] G. Bruder, F. Steinicke, K. Rothaus, and K. Hinrichs, "Enhancing presence in head-mounted display environments by visual body feed-back using head-mounted cameras," *2009 International Conference on CyberWorlds*, pp. 43-50, 2009.

[8] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *Proceedings of the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 1999)*. Fort Collins, Colorado, USA: IEEE Computer Society, 1999, pp. 246-252.

[9] Z. Zivkovic, "Improved adaptive Gaussian mixture model for back-ground subtraction," *Proceedings of the 17th International Conference on Pattern Recognition* (ICPR 2004), no. 2, 2004, pp. 28-31.

[10] D. S. Lee, "Effective Gaussian mixture learning for video background subtraction." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 27, no. 5, pp. 827-832, May 2005.

[11] M. Harville, G. Gordon, and J. Woodfill, "Foreground segmentation using adaptive mixture models in color and depth," *Proceedings of the IEEE Workshop on Detection and Recognition of Events in Video*. Vancouver, BC, Canada: IEEE Computer Society, 2001, pp. 3-11.

[12] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-layer segmentation of binocular stereo video," *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, vol. 2, 2005, pp. 407-414.

[13] E. Hayman and J. O. Eklundh, "Statistical background subtraction for a mobile observer," *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV 2003)*. IEEE Computer Society, 2003.

[14] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in Proceedings of the 12th IEEE International Conference on Computer Vision (ICCV 2009). Kyoto, Japan: IEEE Computer Society, Spe. 2009, pp. 1219-1225.

[15] A. State, K. P. Keller, and H. Fuchs, "Simulation-based design and rapid prototyping of a parallax-free, orthoscopic video see-through head-mounted display," *Proceedings of the 2005 International Symposium on Mixed and Augmented Reality (ISMAR)*, 2005, pp. 28-31.

[16] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, vol. 46, no. 1, pp. 81-96, 2002.

[17] E. Suma, D. M. Krum, and M. Bolas, "Sharing space in mixed and virtual reality environments using a low-cost depth sensor," *IEEE International Symposium on Virtual Reality Innovations*, Singapore, Mar. 2011.

[18] Z. Jiang, M. Yai, and W. Jiang, "Skin detection using color, texture and space information," *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, 2007, pp. 366-370.