

Computer Vision for Augmenting Reality

Matthew Beckler and Andrew Frenz

May 8, 2008

Abstract

A system for augmenting video streams with other images or video has been developed. Utilizing inexpensive digital camcorders, color-based occlusion detection, and projective image transformations, single images or whole video streams can be overlaid over a selected region of a larger background video stream. Matlab is used extensively to read, process, and output video files in a variety of formats. Split-color image analysis is used to allow for the segmentation of background from obstructions, and the overlay frame is drawn appropriately. The projective transform is used to transform the overlay stream to fit over the marked region.

1 Introduction

An interesting area of Computer Vision deals with augmenting reality. Augmented reality is a field of research that studies how the real world can be combined with computer generated data. Usually the viewer must wear goggles or have some sort of other screen or projection through which to view the combination of the real and virtual world. There are many different areas of study within augmented reality and computer vision. For this project, we chose to work on developing a system to augment reality by embedding video and images into an environment. The idea is that certain objects in the real world would be recognized by the computer vision system and replaced or modified with video streams. For instance, four markers could be set up on a wall and the vision system would recognize these markers and replace the wall contained inside the four markers with a window that has video of whatever outdoor scene you wanted. Someone living in the inner city could create virtual windows showing a sunny beach paradise outside.

Besides virtual windows, there are many other potential applications. For instance, billboards in the future may be blank white. A computer vision system in the car would recognize a billboard and would replace the white billboard as seen through the windshield with a custom ad tailored to whomever was in the car. When a man interested in technology is driving, the billboards would show ads for computers, palm pilots, and other products of this category. Then when his teenage daughter takes the car out that night, the exact same billboards would instead show her makeup and clothing ads along with whatever else

teenage girls might be interested in. These are just a couple of the applications that could be implemented with a computer augmented reality system that recognizes certain things in reality such as markers on a wall or a billboard outside, and replaces it with images and video to enhance the human's experience.

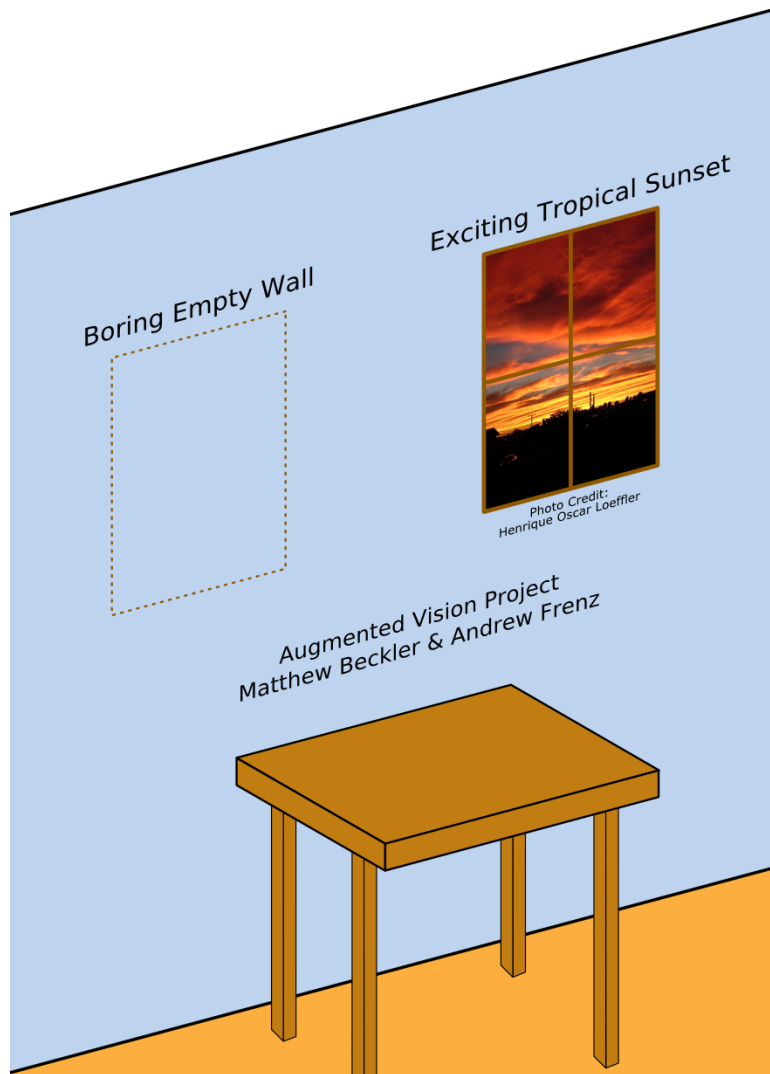


Figure 1: Project Demonstration

2 Problem Description

For our Computer Vision project this semester, we will develop a vision system to embed images and video within the real world environment. The natural place to start will be to augment pre-recorded reality by editing video that has already been shot. This will allow us

to develop our routines and test out our system. Then as we progress and our augmentation system code is optimized, we can try to implement it in real time, which is of course the ultimate goal.

In order to tackle this problem, we will begin with simple cases and work our way up. Our first goal is to simply place a colored sheet of construction paper on a white wall. We will film the wall and then perform post-processing on the video to replace the rectangle with an image. Next, we will film the same wall but with a person walking by. We will develop code to only replace the wall with the image in the areas where the wall can be seen. This way, when the person walks by and covers up parts of the wall, it will look like the image is on the wall in the background and so it will be partially covered up as the person walks by. The next step will be to do the same image augmentation but with the camera at different angles and perhaps part of the rectangle out of the view of the frame. This will require that we scale and rotate the image so that it has the correct perspective, just like a picture hanging on the wall would look if you looked at it from different angles. From here, we can then replace the rectangle with video. A great application of this is to replace the paper on the wall with a window frame and video behind it creating a virtual window in a place where no window exists. We plan to have demonstration videos of this available in the classroom.

Once we receive millions of dollars of funding to start a business to use this same technique for billboard replacement, we will apply the same concept to billboard detection and replace the billboards with custom images and video tailored to each car owner's particular buying habits. But we do not expect to get to this point by the end of the semester.

3 Prior Work

Using standard PC hardware, stereo cameras, and retroreflective markers, Klaus Dorfmueller of the *AGDV Computer Graphics Center* has developed a system to do augmented reality. [1] A related challenge with real-world camera work is the distortions caused by perspective, with work done by Australians Shaw and Barnes [2], where they use detected lines that converge at a common vanishing point to find transformed rectangles along corridor walls. When dealing with the actual overlays of data onto a video stream, a number of important issues are considered by Drascic and Milgram, including how to provide for the appearance of proper depth and shadows to drawn images. [3] For international soccer (football) events, many advertisers wish to target local viewers. For these purposes, Medioni et al of USC have developed a system of real-time replacement of billboards from soccer stadiums without requiring the cooperation of the camera operators or the billboards. [4] Many of the same authors of [4] have also been granted a patent [5] describing the hardware and methodology of perform their developments of [4].

4 Project Solution

We have made considerable progress towards our goals. Starting from a suggestion by the professor, we began the project using a solid color background to define a region that we wanted to replace with video. We decided to use a rectangular sheet of paper instead of simply using markers. This decision was made to enhance our occlusion detection because the color difference of objects in front of the rectangle would be greater than if the wall color was used as the original backdrop. We first went and purchased a few large sheets of brightly colored paper. The selection was somewhat limited but we picked out a large neon orange sheet and a large green sheet. Next, we taped the sheets of paper to a wall and filmed them with a video camera at different angles, with people walking in front of them, and with varying lighting conditions.

Using the video that we captured, our next task was to configure Matlab so that we could load and edit the video in it. The built-in Matlab routines did not have support for diverse enough video types, but code was found on the Matlab website to be freely available that allowed virtually any video file to be loaded into Matlab. Using this routine, we loaded the video that we shot of the colored rectangle. This allowed us to play with the frames and begin developing a method to detect the rectangle and augment it.

The natural way that we found to process the video was by going frame by frame through it, converting it to an image, performing the processing and augmentation, and then converting it back to a movie frame. We decided to use the first frame to detect the marked overlay region, as a pre-processing step. From this frame we determine the parameters of the rectangle, which are simply the locations of the four corners. We started out by using the video of the rectangle viewed straight on and not at an angle or skew. After detecting the rectangle in the first frame, we use the information to replace the rectangle in subsequent frames with an image. If the frame has an occlusion, we determine the part of it that has an object in front of it and replace the rest with the image. This way, if someone walks in front of the rectangle, they will appear in the foreground and the image augmented into the video will appear in the background. In order for this to work, we chose to use video shot on a tripod so the rectangle was always in the same position in subsequent frames.

To find the rectangle in the first frame, we do thresholding on the RGB colors of the image. We set the threshold to be a broad range around the color of the rectangle we are trying to find. This thresholding allows us to find all of the points that are a part of the rectangle. Then, we can simply use max and min functions on these points to determine the four corners. These four corners give us the location of the rectangle. In order to pick out the edges, we buffer this rectangle by 5 pixels because the edges usually show up very dark and do not make the thresholding. When this is done, it produces a cleaner augmentation. Then, we save these coordinates along with the pixel information for each pixel inside the rectangle.

For every subsequent frame, we concern ourselves only with the pixels inside the rectangle. The first thing we do is get rid of all the other pixels by using the rectangle coordinates

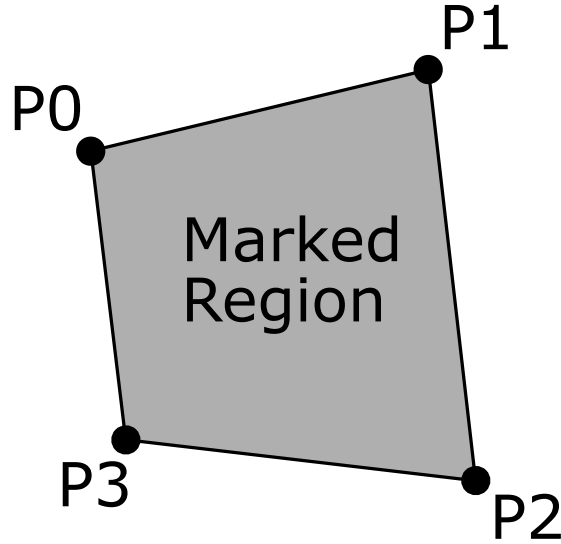


Figure 2: Marked Points for Projective Transformation

to select those we care about. Next, we determine the Red, Blue, and Green difference of each current pixel from the original image. We then do thresholding on these differences. For the neon orange rectangle, we put most of the weight on the Red pixel difference. We then look at max of the difference image and if the max is over a certain threshold, then it means that there is an occlusion. If there are no pixels picked by this thresholding, then there is no occlusion and we simply replace the entire rectangle with the image. If there is an occlusion, then we replace the rectangle only in those parts whose difference is below a certain threshold (meaning the pixels are not part of an occlusion). This produces an augmented rectangle on the wall that shows an image different than is really there and it even shows up in the background as people or other objects go in front of it!

The following chart shows the Red, Blue, and Green difference values plotted out over time for a video clip where initially the frame has several stationary objects including the colored orange rectangle then someone enters the frame, walks in front of the rectangle, and then leaves the frame. We plotted the max difference values for each color and the plots are color coded based on what color they represent.

There are a couple of interesting things with this plot. First of all, you can clearly see when the occlusion occurs. From approximately frame number 60 - 80 there is a hill that indicates huge differences in the current frame's pixel values compared to the original frame's pixels. This corresponds to when the person walks in front of the colored rectangle. This max difference plot serves as our occlusion detector. You may notice that the maximum difference is seen in the Red color which makes complete sense since the rectangle used in the trial was orange which shows up in the camera with a large amount of red. The other two colors also have a significant color difference from this occlusion but not as much as the

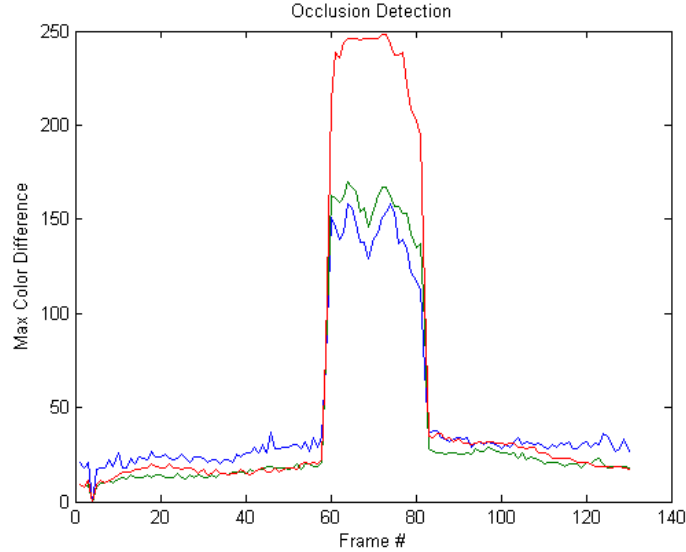


Figure 3: Split-channel Occlusion Plot - Example 1

red color. Another interesting thing to note from this chart is that when no object is in front of the rectangle, there is still some difference noise. This noise is actually much greater than we would have thought and is about 25 for blue and 10-15 for red and green. This was a source of difficulty at first because we expected that the difference would be only a few values when no object was in front of it and so we set our thresholds considerably too low when looking for an occlusion. After constructing plots similar to this, we realized that we needed to raise our threshold so that the noise of frame-to-frame with no occlusions was eliminated. Once we did this, then our occlusion detection was much more robust.

Another example of our occlusion detection mechanism is shown in figure 4 on page 7 in a more involved example. In this sequence, there are three different occlusions and the movie ends with a fourth occlusion.

Once again, we place the majority of the weight on the red because of the rectangle's color. Using thresholding on this image, we get a very definitive occlusion detection mechanism. By using this fast occlusion detection scheme, we don't have to consider the specifics of where the occlusion is if there isn't one detected. This of course allows us to process many frames very fast. An interesting note here is that the blue this time has a no-occlusion difference noise of almost 40. The video used for this plot was taken in the night-time which probably resulted in more blue-like conditions which the video camera varied much more than in the first example which included brighter conditions and consequently less noise in the blue range.

We used the following method to replace the rectangle with an image (to do the actual

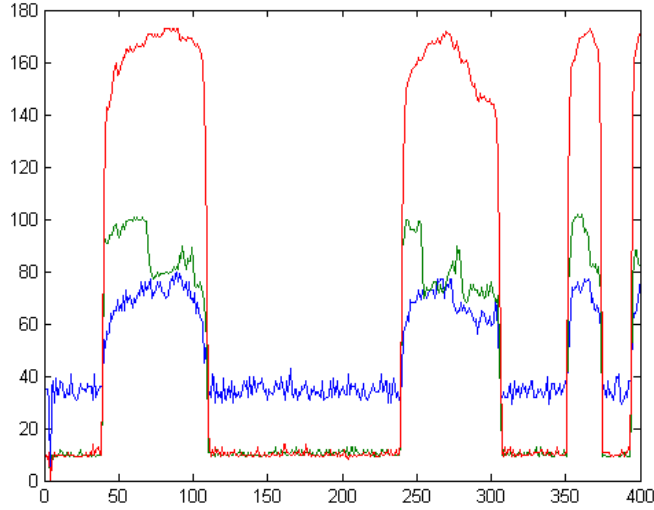


Figure 4: Split-channel Occlusion Plot - Example 2

augmentation). An input image is loaded into the routine. Then after the parameters of the rectangle are calculated in the pre-processing, this image is transformed to fit the rectangle size and orientation. As discussed previously, we have already detected the four corners of the parallelogram that is the overlay area. We use these four corners to transform the input image or video stream. A routine has been developed that will scale, rotate, and otherwise transform a source rectangular image into any regular quadrilateral, perfect for pasting into another image. The reasoning behind development of this algorithm is that often times, the area to be replaced is not a perfect rectangle, but rather a rectangle that has been skewed and transformed by the angle of the camera's view. For instance, if we desire to replace this piece of classic artwork on the living room wall with something more interesting, such as this vintage Star Wars movie poster, we can't just paste the movie poster over the artwork, as the viewing angle has transformed the art from rectangular into a more complex quadrilateral.

To perform the actual image transformation, we use the *Projective Transformation*, which is a superclass of the *Affine Transform*[6]. The difference between these two types of transformation is visible when comparing their 4×4 transformation matrix:

$$T_{affine} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{projective} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix}$$



Figure 5: Original Artwork



Figure 6: “Augmented” Artwork

Having more elements variable, the projective transformation has more freedom to manipulate images. We use the following Matlab code to create the desired transform based on the desired input and output coordinates:

$$\text{tform} = \text{maketform} \left(\begin{bmatrix} 1 & 1 \\ S_{x_{ov}} & 1 \\ S_{x_{ov}} & S_{y_{ov}} \\ 1 & S_{y_{ov}} \end{bmatrix}, \begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix} \right)$$

Where $S_{x_{ov}}$ and $S_{y_{ov}}$ are the dimensions of the overlay image. x_i and y_i are the coordinates in the main image where the overlay should be placed. They are provided in the order described in an image earlier in this document.

After every frame is processed, we compile the new frames to create a new movie. This movie can then be viewed or saved to disc. After initially starting out by augmenting the rectangle with the same image from frame to frame, we decided to tackle the issue of augmenting the rectangle with video. Our design for the still image turned out to be very conducive to augmenting with video, and after some short code development we were able to augment a rectangle with video.

To do this, we load an additional video to serve as the video to replace the rectangle. Then, as we increment through the video frames, we increment through the second video frames as well. Each time, we take each overlay video frame, convert it to an image, and then do the same processing that we initially did with just the still image. This includes scaling it and handling other changes to account for perspective. Then that individual frame of the overlay movie is used to augment the individual frame of the main video. In this fashion we augment the video with video. Some frames of one test video are shown to demonstrate our augmentation.

These frame-shots of the output video demonstrate our augmentation algorithm. First of all in the top frame set, the orange rectangle is augmented to show video of a Star Wars space battle. The second clip set shows how when an arm is moved in front of the rectangle, selective augmentation determines where exactly the occlusion takes place and it augments the rectangle only where there are no occlusions so the rectangle remains in the background as it should. The third clip set demonstrates how, when a person walks by between the camera and rectangle, nearly all of the rectangle has an occlusion and does not show up except for the small sections that are not obstructed by the person.

Throughout the development of this project we have encountered many different obstacles. Early on we had to learn how to deal with lighting to eliminate shadows and make sure the background was well lit. Some of our test video with shadows was a nightmare to try out because it was impossible to distinguish between shadow and real objects. We improved our

Before and After

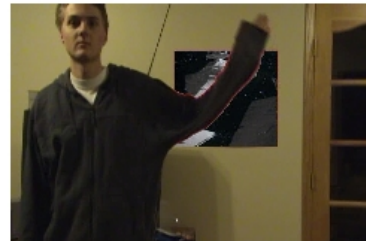
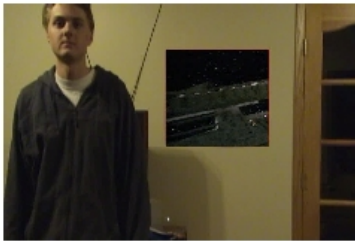


Original Frame



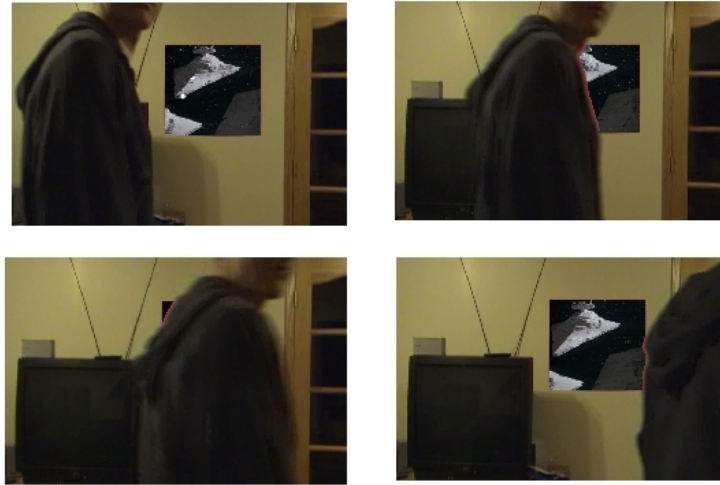
Augmented Frame

Demonstration of Occlusion Detection and Selective Augmentation



video by learning how to arrange the lighting such that shadows were minimized. Another decision we made while developing the code was to shoot the video on a tripod. Initially it was done by hand, but this meant that the rectangle moved around constantly and as soon as it became occluded, it was impossible to determine the exact parameters of the rectangle well enough since if only two corners were still visible the exact size of the rectangle could not be determined. By using a tri-pod, we could determine initially the orientation of the rectangle by requiring the first few frames to have no occlusions. Then, we could use this information to subsequently produce realistic partial augmentation in the presence of occlusions. A third issue we dealt with was specific thresholding techniques and values. Many different schemes were tried, but in the end the difference method of each pixel that we use turned out to work the most robustly and produce the best results. We tried other schemes which looked at absolute values or averaged over the rectangle, but often times the color varied drastically from one side of the rectangle to the other which caused many problems. Another issue we have

Near-Complete Occlusion: Person Walking By



is with a thin orange glow appearing around occlusion objects that you may have noticed in the images above. We have not yet fully resolved this glow although we have minimized it.

This concept of replacing a skewed rectangle with a matching skewed image was extended to video for both the original scene and the augmentation. A rectangle was filmed at an angle. The rectangle's corners were detected and the augmentation video was transformed to have the correct view angle. The following screenshot shows how a virtual paper-thin flat monitor television was augmented into reality such that it is hanging on a door. The virtual television is playing a light saber duel from Star Wars. Notice how the Star Wars scene appears with the correct perspective.



This technique was also tested to the extreme. We filmed the same situation but at a

much more drastic angle and from low to the ground looking up. You can see that the virtual television displays the correct perspective even from this severe viewing angle.



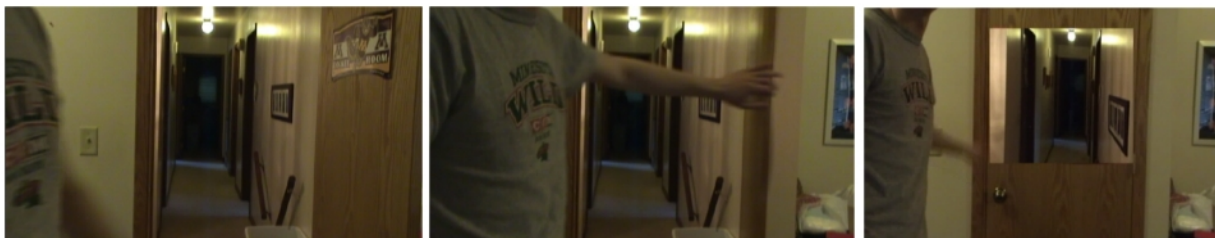
Another trial augmentation that was run was to see how well the system did for fast moving thin occlusions. To test this, we filmed a rectangle head on and the subject twirled around a meter stick in front the rectangle. A video stream was augmented in replacement of the rectangle and occlusion detection was done so that the augmented video appeared always in the background. The following screenshots show the results:



And here the meter stick was held so the thin side was facing the camera. The occlusion detection is able to successfully pick out the thin occlusion as well as the entire arm and hand and display it correctly.



Besides augmenting video streams into reality, this same system can also be used for creating augmented invisible walls. To achieve invisible walls, the same technique of augmentation and occlusion detection is used, but instead of any old video stream being added, a video stream of what is behind that wall is augmented. This produces an effect as if the viewer is looking through the wall because they see whatever is behind it. To demonstrate this technique, we placed the rectangle on a door. This way the viewer can see what is behind the door when it is open, and when the door is closed we can make a portion of the door invisible to the viewer. The following screenshots show this effect.



The door looks like a normal door but then as it swings shut it, a large portion of it becomes invisible, much like the doors with a top that opens up. The viewer can see out the door and down the hallway despite the door being closed. There are many potential applications for this. A high tech security system could place video cameras on the outside of a building facing outwards from the exterior walls. A projector could then project video on the interior walls showing the outside video. Or even better, a headset could also be worn by a security guard so that he could walk around through the building and whenever he looked

at an exterior wall, his headset display would make the wall invisible and allow him to see through it just like you can see through the door in this example. The same augmentation technique as used here could thus be directly applied for this security application as well as many other uses for invisible walls.

5 Conclusion

The methods developed through this research have worked admirably. We are able to detect regions of interest and transform an image or video to be overlaid on a background video stream. We are able to adjust for differing perspectives and appearances of the overlay region, as well as being able to successfully handle occlusions over the overlay image. Split color channels were used to perform occlusion detection and thresholding, which allowed us to adjust the process to differing lighting conditions and occlusion objects. In the end, we were able to successfully add video and images to reality, such that they integrated with the environment and had correct properties of perspective and occlusion.

6 Future Work

As noted above, future work could look at improving the reduction in edge glow around an occluding object. We would also like to investigate techniques for detecting the desired overlay area by using indicators other than color alone. Corner or edge markers would be a good area to focus on, as they could potentially be made small enough for humans not to notice them. The use of infrared-sensitive materials would also be a very good way to hide markers from human eyes, as most camera CCD elements are sensitive to infrared light, invisible to the naked eye. Another area of improvement would be to relax the tripod restriction. We could use more advanced tracking techniques, such as using the optic flow to record the camera's movement relative to the wall, and to use these velocities to predict where the camera is moving, to allow for smoother tracking. Adding an accelerometer to the camera would allow it to use proprioceptive measurements to track its movement. We could use a Kalman filter to handle the uncertainty inherent in all of these measurements to produce the best possible estimate of camera motion. Another area of improvement would be a reduction in the computational complexity of the method, as currently the system is not real-time. It would be relatively simple to use the C programming language instead of MATLAB, and custom hardware could be developed to quickly compute the image transformations.

7 References

- [1] Dorfmler, K. "Robust tracking for augmented reality using retroreflective markers", Computers & Graphics, 1999.
- [2] "Perspective Rectangle Detection", by David Shaw and Nick Barnes, Applications of

Computer Vision, Workshop at the European Conference on Computer Vision (ECCV), May 2006.

- [3] Drascic, D, and Milgram, P. “Perceptual issues in augmented reality”, SPIE, 1996
- [4] G. Medioni, G. Guy, H. Rom, and A. Francois. Real-time billboard substitution in a video stream. In Proceedings of the 10th Tyrrhenian International Workshop on Digital Communications, Ischia, Italy, 1998. <http://pollux.usc.edu/~afrancoi/pdf/TIWDC1998-paper.pdf>
- [5] G. Medioni, G. Guy, H. Rom, 'Video Processing System for Modifying a Zone in Successive Images', U.S. Patent # 5,436,672, July 1995.
- [6] Weisstein, Eric W. “Affine Transformation.” From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/AffineTransformation.html>