Realtime Object Occlusion In Augmented Reality Environments.

Niyati Shah
Department of Computer Science
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623
nxs6032@cs.rit.edu

Abstract—In the world of augmented reality, the real world is purely a background and the virtual objects are superimposed as the foreground objects of this world. These virtual objects appear in front of any real-world object when rendered as they don’t have any sense of depth. For a more realistic augmentation of objects you want them to interact with the real-world object, cast shadows just like the real-world object and reflect light like the real-world objects. This project deals with the problem of occlusion by making virtual objects closely resemble the real objects by using the depth and color information in the scene to clip the virtual object to create a sense of occlusion between the objects.

Keywords—Augmented Reality; Occlusion;

I. INTRODUCTION

Augmented reality (AR) is a modification of Virtual Environment (VE). In virtual reality, the user is completely immersed in a virtual world whereas in Augmented reality the user in immersed in a world where virtual objects are combined or superimposed on the real world. “Augmented Reality” the word was defined by Ronald T Azuma in his paper ‘A survey of augmented reality. Presence: Teleoperators and Virtual Environments’. According to him “The ultimate goal of augmented reality will be to generate virtual objects that are so realistic that they are virtually indistinguishable from the real environment” - Ronald T Azuma 1997 [1].

Until now most of the research on augmented reality has been on creating head mounted devices to produce augmentation. To improve the illusion of augmentation, researchers are now looking at how different objects interact with each other in real world and apply those learnings to the virtual world to produce realistic virtual objects that interact with real objects. The different interactions that are considers are collisions, shadows, lighting, and occlusions between the real objects and virtual objects.

Occlusion occurs when the view of one object is obstructed by another object. But in augmented reality, the virtual object is just overlaid on the real scene that is captured by the camera. So even when objects are placed behind real object they appear to have been placed in front of them when viewed from the front as seen in figure 1. Where we can see from the side view, figure 1.a, that the heart is placed behind the box but from the front view, figure 1.b, it is seen rendered in front of the box instead of behind it. This is the problem of occlusion in augmented reality and this paper will be focusing on handling this kind of occlusions between the real and virtual objects.

II. PRIOR WORK

Various approaches have been suggested by researchers that create an augmented reality experience using personal devices like mobiles and tablets and also using head mounted devices like HoloLens and other smart glasses. But most of
these do not handle occlusions between real and virtual objects in real time or correctly. Some of the ways that these devices handle occlusion are discussed below.

A. HoloToolKit

HoloLens uses spatial mapping to detect the surrounding environment before rendering the virtual objects on it. Spatial mapping is a mesh that is created by a device by discovering the environment around it. It lets you understand the area around you by working with the data given to it or getting the data from the natural physical interactions in the world. HoloToolKit and Asobo Studios have made it easy for users to scan the environment for features in a few seconds and use it to create a mesh. [2]

The two objects used for spatial mapping are Spatial Surface Observer and Spatial Surface. The spatial observer gives us one or more bounding volumes which defines the region in which the spatial mapping should start and each mapping is stored in the bounding volumes. These spatial surfaces give us the constraints around the environment which can be used to make the users interaction more natural.

Occlusion is one of the most important expectation of a user as it is based on the natural interaction the user faces. Although the spatial data from HoloLens creates a very neat mesh of the environment, as seen in figure 2.a [3], it still does not handle occlusions very well. As seen in figure 2.b, the interaction between the table and the star is not perfect. This is because the mesh of triangles created is pretty large and it therefore gives us a jagged edge instead of actual smooth edge which is seen in real life object occlusions.

B. ARKit

It uses Visual Inertial Odometry, where it combines iOS device’s motion sensing hardware data with computer vision analysis of the world visible to the device’s camera. [4] It then detects distinguishable features points in the world, follows the differences in the positions of those key points across the frames, and compares that information with motion sensing information as seen in figure 3.a. This then results in a very precise model of the device’s motion and position. [4]

Although a very powerful algorithm, there are a few things that don’t make the occlusion handling realistic. It does not handle occlusion in real time. The way that ARKit handles occlusion is by asking the user to map the area, i.e. tell it where the walls objects are and then an appropriate mask can be built that will occlude the virtual object as seen in figure 3.b. [5] Mapping areas of few big objects like a sofa, wall or a door is easier but a real world contains so many more objects
which are tiny and cannot be easily mapped specially if you keep moving and the scene changes.

Also, right now we can only detect horizontal planes, which means a lot of these objects will not be detected and mapped easily.

C. Other Work

There are many other different ways proposed by researchers to handle occlusions for various toolkits and standalone ways for augmented reality. These use sensing, reconstruction and rendering to achieve occlusion.

Sensing the 3D world around us requires depth sensors, which could be structured light, time of flight or stereo cameras. Structured light projects IR light pattern onto the object and uses the distortions of the pattern formed on the object to construct the 3D object. Time of Flight constructs 3D objects by calculating the depth from the difference in delay of reflections of IR light emitted by it. Stereo cameras use two cameras to triangulate the distances between the points in the world.

All these techniques construct a very good 3D scene of the real world but they all work only for a few meters. The IR based sensors do not work well in outdoors in sunlight and stereo cameras only work in well-lit areas with high contrast. Even if all the conditions are met, these sensors cannot create a perfect depth map which is needed to handle occlusions as seen in figure 4. [6]

III. PROPOSED SOLUTION

To handle occlusions, I will be following the algorithm of bounding volume proposed by A. Fortin, P & Hebert, P as shown in figure 5 and also try to improvise on their algorithm. The algorithm works as follows: [7]

- There are two cameras used, one is a tracking camera and the other is performance camera. The tracking camera will look at the scene and process the objects in it while the performance camera creates the augmented viewpoint for the user.
- We extract the silhouette of the object by performing background subtraction of the object.
- Next, we render the image obtained by the performance camera to a color buffer.
- Then we initialize the depth buffer with an presumption that every pixel is originally located at depth $z = \infty$.
- Then we record a binary mask plotting the derived silhouette to a stencil buffer and enable a stencil test.
- Next, we disable the color buffer and add the bounding volume in the depth buffer where the stencil test succeeds. This will segment the original bounding volume to the shape of the silhouette obtained from background subtraction process.
- Finally, we enable the color buffer and draw the virtual object where the depth test succeeds.
As the paper by A. Fortin, P & Hebert, P was written in 2006, it did not have as much information about the surrounding as we get right now in 2018. So as part of my improvement, I would like to incorporate the information available to me like the spatial mapping of hololens in the implementation. I would also not need two different cameras as I will be working with the hololens which has an inbuild camera tracking and a renderer screen that can replace the two cameras in the algorithm. This algorithm uses standard bounding volumes on the objects, instead I would also like to experiment with different bounding volumes created using 3D convex hulls to save on space and probably some processing time which is definitely required when working in real time.

IV. DESIGN

The project was planned to be implemented in fifteen weeks from scratch. After doing a lot of brainstorming and reading of different papers I decided to go with the algorithm proposed by A. Fortin, P & Hebert, P as a template to build my system. Things like what systems to use and how will I improve the current system in the augmented reality development kits were taken into account. Given below is the final design flow of the system and figure 6 shows the algorithm as a flowchart.

- The input to the system is frames of images of the scene that is being viewed by the user. This frame is then cleaned and prep by performing noise cleaning using Gaussian and smoothing filters for further processing.
- Next we wait for the user to select the regions of surrounding and the object behind which he/she wants to place the virtual object.
- Once the regions are selected we can apply any segmenting algorithm that gives us complete contours to create a mask by filling the surrounding in black i.e. with all zeros and the selected object in white i.e. with all ones. After trying many different algorithms like canny, sobel etc I found that the watershed algorithm [8] worked the best for my system to give me a complete and smooth mask without any jagged edges.
- Once we have the mask we move onto creating depth images from the scene. HoloLens creates spatial data using the four cameras that continuously scans the environment. One other way of doing this is to use a depth sensing device like kinect that is the perfect substitute of the HoloLens, as the Kinect was the precursor to HoloLens Spatial maps. We can also get a similar depth data by using only software and some extra input from the user. This is done by asking the user to continue looking at the scene and move left a few steps or move right a few steps. This helps us create a stereo system to calculate depth. [9]
- The depth data can be collected beforehand by scanning the room or the scanning the area where behind which we want to place the object to emulate the HoloLens.
- Generate depth information using any of the above techniques from the scene and divide the depth information in 5 levels using histograms to create an array that stores the values by which the depth is divided.
- Next ask the user to click the character ‘p’ on the keyboard to draw the virtual object. This virtual object is first drawn in front of all the different objects in the scene. The user can then navigate the object throughout the scene using ‘w’, ‘s’, ‘a’, ‘d’ keyboard keys to move the virtual object up, down, left and right respectively. The characters can be either in upper or lower case.
- According to the movement of the virtual object the segmentation mask and the depth masks are combined and applied to the virtual object and the result is rendered in the scene. The segmentation mask clips the virtual object and creates the illusion of occlusion in the scene.

V. IMPLEMENTATION

A. Assumptions and constraints

There are some assumptions made while building the application along with some constraints. This project is mainly a proof of concept on how the current technology and a couple of algorithms can be combined to create a seamless mix of virtual and real objects.

- It is assumed that we already have the depth data of the scene like spatial map or stereo depth data.
- Currently the user needs to select the object behind which he/she wants to place the virtual object and also select the surrounding.
- As most augmented reality kits have advanced tracking system and include geo-anchor points to place the virtual objects so when the user looks around and comes back to the point where he/she had placed a virtual object earlier they find it exactly there. This project does not use these kits but it has its own augmented reality kit, so an assumption is made that we have the same technology and can extract that data from the system.
To create an environment similar to the development kits as part of the project the user is not allowed to look around and is asked to keep only the view of the current scene with the selected object which will be processed to perform occlusion between the virtual and selected object.

As part of the virtual object the user can only choose a cube object.

The depth has only 5 levels instead of being continuous till infinity. The 5 levels are derived by dividing the maximum and minimum distance of the depth into 5 levels for faster processing. In future, we can let depth be $\infty$ and each distance have a continuous depth level instead of only a selected depth level.

B. Working

When the program starts, the camera is the viewing device for the augmented reality system. You can look around with it and then fix it to the place where you want to place the virtual object from. Then the user selects the object behind which he/she wants to place their virtual object and the rest of the surrounding.

Once the selection is done, the user clicks “P” to draw the virtual object. First the virtual object is drawn at the center of the scene. Then the user can use ‘w’, ‘a’, ‘s’, ‘d’ keyboard keys to move the object up, left, down and right respectively. Then the user selects the depth level from 1 to 5 and places the object accordingly at that depth in the scene. Here 1 is completely in front of the scene and 5 is furthest away from the camera view. Figure 7 shows the working process of this system.

C. Experiments

To test the implementation, project was run through the Middlebury’s stereo dataset from the year 2001[9], 2003[10], 2005[11][12] and 2014[13]. The dataset included images and its depth information. The dataset contained images with many different shapes of objects placed at different distances from the camera. It contains left and right image to create a stereo set. As in the assumption section, we have stated that we already have the depth information from the scene, this scene fit the project perfectly. It provided extensive set of data to test the project. Results of the experiments can be seen in table 1.

Through the dataset, we could see how this system worked in different kinds of environment and its shortcomings. The depth data is prone to errors due to occlusions and the distance between the camera and objects as shown in figure 8. The error causes some holes in the image which show up while processing the data to create mask as seen in figure 9. Other issues have been segmenting objects from the surrounding. Many objects have similar texture and color as its surrounding or they are affected by their surrounding environmental aspects like lighting shadows etc. which impact it being segmented by the surrounding as seen in figure 10. If there are issues in getting depth data and segmenting the surrounding, the resultant mask will also be incorrect and giving an erroneous clipping and occlusion as seen in figure 11.

D. Challenges

Throughout the implementation of the project I have faced quite a few challenges with the software and hardware of the system. As the latest technology of Augmented Reality is quite new and not mature enough, only some versions of IDE’s support them and there is no proper backward compatibility between the versions.

Initially I had planned to integrate my program with at least one of the augmented reality devices like the HoloLens to show how my program works as compared to the inbuilt
occlusion facility of these devices. To do that I needed my C++ program to interface with Unity, which is the primary developer IDE for HoloLens and other augmented reality devices. Unity only supports C# and JavaScript as its programming language and Open CV only supports C++, Java and Python as its programming language. So, this was one of the biggest issue I had to find a way to integrate both. After trying a logical few ways, researching Unity support, Open CV support and other developer forums, I was unable to find a good way to integrate both. There was an asset in Unity called ‘Open CV for unity’ [14], but it did not have all the functionality and libraries needed from Open CV for my project. Also with a price tag, most reviews showed that the author was not responsive when and when responsive and it had barely any documentation to go about it. So, I decided to create my own virtual object using C++ graphic capabilities.

Apart from this installing and working with Open CV was also a major issue. Installing Open CV using the binaries or executables was also very troublesome as it dependent on a lot of different environment variables that had to be set and following installation tutorials did not necessarily give the right results.

After Open CV was installed on my personal laptop smoothly when I tried to run some of the basic Open CV commands of reading in and displaying an image, I got an error, ‘BUG: The current event queue and the main event queue are not the same. Events will not be handled correctly. This is probably because _TSGetMainThread was called for the first time off the main thread.’.

After searching on the different Open CV forums and also in general on the internet and trying the different solutions provided there I contacted the Open CV team who said that it wasn’t an Open CV problem and I should contact Apple or my IDE company as the things with the main thread of the system were not handled by them. After going to all of them, talking to different professors in college who work with Open CV and looking at the different posts online by people who had faced the same error, I realized it wasn’t really an Open CV bug or even an Operating System Bug as many people who had got this error had different operating systems and were also using different applications. As this bug could not be recreated else where it was impossible for any of the teams to really work on it and find a solution for it. So, after spending a lot of time on this I finally tried installing Open CV on a different machine and started working on that machine.
### Table 1: Result Table

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Selected Image</th>
<th>Color Segmentation</th>
<th>Depth Clustering</th>
<th>Depth Level</th>
<th>Final Mask</th>
<th>Object Clipping</th>
<th>Final Scene</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Image 1" /></td>
<td><img src="image2.jpg" alt="Image 2" /></td>
<td><img src="image3.jpg" alt="Image 3" /></td>
<td><img src="image4.jpg" alt="Image 4" /></td>
<td>1</td>
<td><img src="image5.jpg" alt="Image 5" /></td>
<td><img src="image6.jpg" alt="Image 6" /></td>
<td><img src="image7.jpg" alt="Image 7" /></td>
</tr>
<tr>
<td><img src="image8.jpg" alt="Image 8" /></td>
<td><img src="image9.jpg" alt="Image 9" /></td>
<td><img src="image10.jpg" alt="Image 10" /></td>
<td><img src="image11.jpg" alt="Image 11" /></td>
<td>2</td>
<td><img src="image12.jpg" alt="Image 12" /></td>
<td><img src="image13.jpg" alt="Image 13" /></td>
<td><img src="image14.jpg" alt="Image 14" /></td>
</tr>
<tr>
<td><img src="image15.jpg" alt="Image 15" /></td>
<td><img src="image16.jpg" alt="Image 16" /></td>
<td><img src="image17.jpg" alt="Image 17" /></td>
<td><img src="image18.jpg" alt="Image 18" /></td>
<td>3</td>
<td><img src="image19.jpg" alt="Image 19" /></td>
<td><img src="image20.jpg" alt="Image 20" /></td>
<td><img src="image21.jpg" alt="Image 21" /></td>
</tr>
<tr>
<td><img src="image22.jpg" alt="Image 22" /></td>
<td><img src="image23.jpg" alt="Image 23" /></td>
<td><img src="image24.jpg" alt="Image 24" /></td>
<td><img src="image25.jpg" alt="Image 25" /></td>
<td>1</td>
<td><img src="image26.jpg" alt="Image 26" /></td>
<td><img src="image27.jpg" alt="Image 27" /></td>
<td><img src="image28.jpg" alt="Image 28" /></td>
</tr>
<tr>
<td><img src="image29.jpg" alt="Image 29" /></td>
<td><img src="image30.jpg" alt="Image 30" /></td>
<td><img src="image31.jpg" alt="Image 31" /></td>
<td><img src="image32.jpg" alt="Image 32" /></td>
<td>3</td>
<td><img src="image33.jpg" alt="Image 33" /></td>
<td><img src="image34.jpg" alt="Image 34" /></td>
<td><img src="image35.jpg" alt="Image 35" /></td>
</tr>
<tr>
<td><img src="image36.jpg" alt="Image 36" /></td>
<td><img src="image37.jpg" alt="Image 37" /></td>
<td><img src="image38.jpg" alt="Image 38" /></td>
<td><img src="image39.jpg" alt="Image 39" /></td>
<td>2</td>
<td><img src="image40.jpg" alt="Image 40" /></td>
<td><img src="image41.jpg" alt="Image 41" /></td>
<td><img src="image42.jpg" alt="Image 42" /></td>
</tr>
<tr>
<td><img src="image43.jpg" alt="Image 43" /></td>
<td><img src="image44.jpg" alt="Image 44" /></td>
<td><img src="image45.jpg" alt="Image 45" /></td>
<td><img src="image46.jpg" alt="Image 46" /></td>
<td>1</td>
<td><img src="image47.jpg" alt="Image 47" /></td>
<td><img src="image48.jpg" alt="Image 48" /></td>
<td><img src="image49.jpg" alt="Image 49" /></td>
</tr>
<tr>
<td><img src="image50.jpg" alt="Image 50" /></td>
<td><img src="image51.jpg" alt="Image 51" /></td>
<td><img src="image52.jpg" alt="Image 52" /></td>
<td><img src="image53.jpg" alt="Image 53" /></td>
<td>3</td>
<td><img src="image54.jpg" alt="Image 54" /></td>
<td><img src="image55.jpg" alt="Image 55" /></td>
<td><img src="image56.jpg" alt="Image 56" /></td>
</tr>
<tr>
<td><img src="image57.jpg" alt="Image 57" /></td>
<td><img src="image58.jpg" alt="Image 58" /></td>
<td><img src="image59.jpg" alt="Image 59" /></td>
<td><img src="image60.jpg" alt="Image 60" /></td>
<td>2</td>
<td><img src="image61.jpg" alt="Image 61" /></td>
<td><img src="image62.jpg" alt="Image 62" /></td>
<td><img src="image63.jpg" alt="Image 63" /></td>
</tr>
</tbody>
</table>
VI. RESULTS

The virtual object is placed behind the selected real object in the scene successfully. There are times when many real objects occlude each other, they do not have any clear boundaries. This results in incorrect segmentation of the object and eventually incorrect occlusion of the object. Even with depth maps, if the user does not take proper images left and right images by moving either left or right correctly we do not get correct depth information, and so the algorithm fails. Also as there are only 5 depth levels, if there is some object is only partially available at a certain depth level and the virtual object is placed beyond it then too the virtual object is clipped. This is currently due to the limitation of 5 levels. If these levels became continuous then we would not face the same problem.

Table 1 shows the results of different scenes where a virtual object is placed in the scene. The first column is the original frame of the scene, the second column is the scene where the user selects the object and the background. The third column shows the mask of color segmentation. The fourth column clusters depth data into 5 levels depending on the distance of objects from the camera. Fifth column gives the depth level selected. Sixth column shows the final mask created from the color mask and selected depth level. The seventh column shows the clipped virtual object according to the final mask in column six. The eighth column shows the final result of the virtual object occluded behind objects depending on the depth it is placed in. A working example of the algorithm can be viewed at : https://youtu.be/ZUZNooDo4eU and the project can be found at https://github.com/iamniyati/OcclusionInAR

VII. CONCLUSION

This was mainly a proof of concept for virtual objects interaction with real objects. Even after quite a few hurdles, this project was completed. This project brings Ronald T Azuma’s vision of indistinguishable real and virtual objects a little closer to reality. As right now there are only a few depth levels we can see some issues with the results, but with complete integration of this system with any of the augmented reality development toolkits like the one in Hololens we can take advantage of the their depth data like spatial maps and create a more seamless results.

VIII. FUTURE WORK

The current work was an attempt to understand how we can use the surrounding information and depth information to create a cohesive scene of real and virtual objects. The project can be extended further in many aspects given as follows:

- Augmented reality heavily relies on scene processing to place objects or even create objects. Many toolkits like Vuforia, Merge Cube use detecting and tracking of objects to create the augmented reality scene. So integrating OpenCV and Unity to combine both their strengths together to create a system that can work in image/scene processing and also in three dimensional system can help us to get a cohesive look of virtual and real objects.
- The depth maps created by our system are prone to error due to the fact that the user may shake a lot or not move enough to give us a right and left image to create a depth map. So instead we can take advantage of the spatial maps created by Hololens to completely remove our depth sensing technique to get accurate depth maps of the scene.
- Right now we only have 5 depth levels, in reality we have a contiguous depth so we need to update the system by using the depth information from the toolkit and make it one contiguous depth that depends on where and how the object is placed instead of dividing it into a few levels.
- Right now the user is not allowed to look anywhere once he/she has selected the object. Here, we can take advantage of the powerful detection and tracking system provided by the toolkits so even if the user looks around when he or she comes back the real object is again detected and the occlusion can take place between the virtual and real objects.
- The virtual object in the current system is not anchored to the scene location and will move around with you in future so we can take advantage of the geo location techniques of the toolkits to place objects in the scene and certain places and it will stay there and not move around with you.
- In future we can use advanced neural networks to segment the objects in the current scene so the user does not have to select real object behind which the virtual object needs to be placed. This can happen automatically.

REFERENCES

Why is occlusion in augmented reality so hard? – hacker noon


