Augmented Reality - Applying noise and blur to the virtual content to make it appear as real objects

Anuj Bhatnagar
Stanford University
Stanford, California 94305

Sreesudhan Ramkumar
Stanford University
Stanford, California 94305

Abstract

Augmented reality (AR) is an interactive experience of a real-world environment where the objects that reside in the real world are enhanced by computer-generated perceptual / virtual objects. One way of experiencing augmented reality is using a handheld device like a smartphone or a tablet and streaming camera feed to the display while augmenting virtual object in it. This is also called as video-passthrough experience. The user views the content rendered to the display where camera feed is rendered from the camera sensor mounted to the device along with the virtual object rendered on top of the camera preview. The cameras present in the handheld devices are small since they need to fit within the form factor of the handheld device. So, the image quality of the camera preview is compromised. The noise in camera preview also worsens if the environment is poorly lit. When a virtual object is rendered in the camera preview, the virtual object looks perfect since the model is generated by a computer. This kills the experience since the virtual object doesn't blend in the surrounding environment and looks obvious to the human eyes that it is artificial. One way of enhancing the experience is to identify the noise present in the camera preview and applying similar noise level to the virtual object so that the virtual object looks like a real object. Another issue with the camera in a smartphone is the limited depth of field within while objects looks sharp and focused when viewed through the camera preview on the display of the handheld device. In general, the depth of field of a smartphone camera is between 20 cm to 2 meters. When the camera is focussed to an object in the near plane, the objects in the infinity are blurred in the camera preview. When a virtual object is rendered near the far plane, they don't blend with the surrounding environment since the real objects surrounding it looks blurred. To improve the visual perception, the virtual object can also be blurred depending on the focus distance being rendered.

1. Introduction

As part of our project, We plan to render the noise and blur to the virtual objects using the noise and blur techniques that we taught in the course lecture (instead of using the APIs). This involves following steps.

1.1. Depth of Field Blur

To render blur to the virtual object, we need to understand the current focal distance at which the camera is focussed. This can be obtained from the metadata of the camera feed. As part of the camera metdata, the camera intrinsic matrix is provided for every frame. This matrix contains the focal length and optical center. From the focal length, we can get rough estimate of the focus distance. When a virtual object is rendered, the focus distance (z-depth assuming camera is the origin) at which the object is rendered is taken. When the object is rendered, a blur filter is applied where the blur strength is proportional to the difference between focus distance and the z-depth of the rendered content. If the difference is zero, then the camera is focussed at the same focal plane at which the virtual content is rendered. So, blur strength is zero and no blur is applied. If the difference is large, blur strength is increased so that virtual content is blurred while rendering.

1.2. Noise

To render noise to the virtual object, We need to understand the noise profile of the camera feed. We plan to analyze the camera image and estimate the noise profile. This would require sampling the texture in the camera feed and look for consistent and uniform region, determine the difference in pixel strength and color across the uniform region. We also plan to use the ISO and lux value provided by the metadata of the camera feed to understand the amount of brightness strength added by the image processing pipeline. This provides some clue about the lighting condition of the environment. When the object is rendered, the noise profile extracted from the camera feed is applied.

Figure 1. llustrating the depth of field blur on the virtual objects in the scene. Source - https://developer.apple.com/videos/play/wwdc2017/604/

2. Related work

Limiting our scope to smartphones and tablets, iOS and Android are the commonly used operating systems in the smartphones. These operating systems have several system frameworks that exposed API's for developers to develop their application. They provide simple and elegant utility functions that performs complex tasks on behalf of the application. Limiting our scope to iOS, ARKit (Augmented Reality Kit) is a framework that provides world tracking and scene understanding capabilities. This framework uses the bayer raw camera mounted to the device along with accelerometer and gyro sensors to predict the world position and orientation of the device in real time. This is obtained by executing VIO and SLAM algorithms. This information is useful to position the virtual object in the scene.

2.1. Depth of Field Blur

Similar to ARKit, SceneKit is another framework that is present in the iOS. This framework hosts a rendering engine and provides elegant API's to the application for creating augmented reality application. This framework exposes the following API.

SceneKit framework

@property(nonatomic) BOOL wantsDepthOfField;

When this property is enabled, the framework renders depth-of-field blur effects to the virtual object.

The following diagram shows an image rendered in virtual reality where the content is blurred depending on the focus distance at which it is rendered.

The same effect can also be achieved when a virtual object is rendered against a camera backdrop.

2.2. Noise

The ARKit framework in iOS provides the following API.

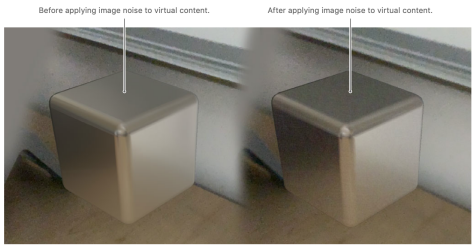

Figure 2. Side by side comparison of rendering content in the camera preview with and without cameraGrainTexture property. Source - https://developer.apple.com/documentation/arkit/arframe/3255173-cameragraintexture?language=objc

ARKit framework

@property(nonatomic, readonly) idMTLTexture $cameraGrainTexture$;

Figure 2 shows the illustration of rendering camera noise to the virtual content. In this figure, the cube is a virtual object. It is rendered against the camera preview in the backdrop. As evident from this figure, the camera preview is noisy. But the virtual object on the left does not show any noise. Hence it looks artificial. When the cameraGrainTexture property is set to YES, the ARKit framework exposes a tileable Metal texture to match the visual characteristics of the current video stream. The texture can be used by the renderer to render noise on the virtual object. This is shown in the right side of figure 2.

3. Method

3.1. Depth of Field Blur

There is a range of distance which keeps objects in focus. Objects outside this range lie in circle of confusion and are out of focus. This size of range is called depth of field. If we place virtual object outside the range of focus, they donot blend well in the scene unless we blur the object to match its surroundings. Therefore to create perfect illusion we should always apply based on focal plane and distance of vitual object in the scene.

In our technique, we are taking the camera image at 60 fps. At the frame rate we generate virtual content at a distance of 1m and render in the world scene. Then we compose the rendered texture by applying blur on objects based on focal length given by the camera and depth of object from the camera. If it lies out of focal range then we apply guassian blur to blend it with surroundings.

Figure 3 shows the depth of blur pipeline that involves grabbing the camera frame and the metadata, generating the

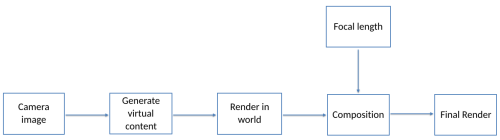

Figure 3. Depth of field blur pipeline

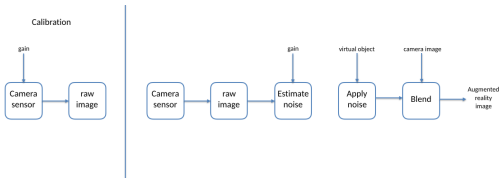

Figure 4. Noise pipeline

virtual content, render it to a world position and compositing virtual content against the camera backdrop.

3.2. Noise

The camera sensors present in the mobile phones have smaller pixel size compared to a DSLR camera since they need to fit within the form factor of the device. So, the image quality is noisy especially in low light scene. When a content is rendered against a camera backdrop in low light scene, the content looks distinct and out of place since the content is rendered artificially. To improve the user experience and create an illusion, We estimate the noise present in the camera image and add similar noise to the rendered content. This involves a calibration process where we capture sensor bayer raw images to estimate signal independent and signal dependent noise for different gain levels. We capture sensor bayer raw image instead of ISP output since ISP tries to remove some of the noise and sharpens edges. So, its difficult to estimate the remaining noise present in the ISP output image. For signal independent noise, We calculate sensor bayer raw image for zero signal and max gain. This amplifies the sensor black level noise and other imperfections. For signal dependent noise, the camera is pointed to a white paper and sensor bayer output is captured for different gain levels for the same scene. We chose 10 different ISO speed ratings from 160 to 1000. We fixed the sensor exposure time to 1/60 fps for all the captures. For these known scenes, we calculate mean, standard deviation and varianc of noise. For a real low light scene running the augmented reality application, We get the gain from the metadata and estimate the noise using bi-linear interpolation of the pre calibrated noise for discrete gain levels. We use this model to apply noise to the current camera image.

Figure 4 shows the noise pipeline that involves the calibration process of capturing known scene for manual gain and estimating the noise for those known scenes. For the real world scene, the pipeline includes grabbing the camera frame and the gain applied in the sensor based on the lighting condition and applying the corresponding noise to the augmented scene and blending it against the camera backdrop.

4. Analysis of other methods

4.1. Depth of Field Blur

Apple released Scenekit as a rendering engine. it provide convenient APIs to enable depth of field blur. We created a small application that uses depth of field technique. It blurs the virtual object similar to our technique. The results when compared visually looked similar.

4.2. Noise

We analyzed the noise grain texture output provided as part of the ARKit functionality. The grain texture is a tileable Metal texture created by ARKit to match the visual characteristics of the current video stream. The camera grain texture is a 3D model that has different noise characteristics for each depth level (z-depth). Along with noise grain texture, ARKit also provides a camera grain intensity that is proportional to the exposure level set by the auto exposure algorithm. For bright scene, the camera grain intensity is lower and for low light the camera grain intensity is higher. A single camera grain 3D metal texture suffices the need for any lighting condition. This camera noise texture and intensity is used by Apple rendering engines - SceneKit and RealityKit. When this feature is enabled, these renderes uses the intensity as z-depth and picks the noise texture and applies it to the rendered content.

The noise grain was rendered to the content for every render pass depending on the camera frame's exposure and the brightness of the scene. The noise was noticeable in low light scene with a rendered content. It also depends on the texture of the rendered content and the background scene.

5. Results

5.1. Depth of Field Blur

The most challenging aspect of the problem is to find the blur radius so that virtual object blends well with other out of focus real world objects. We experimented with 3 techinques 1. Constant blur or low blur 2. Blurring real world objects or high blur 3. Step wise blur

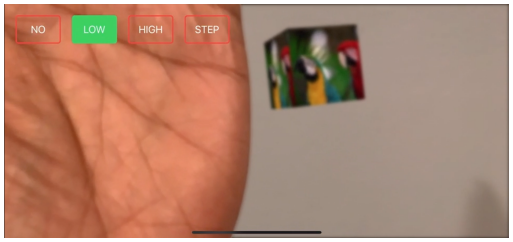

Figure 5. Constant or low blur

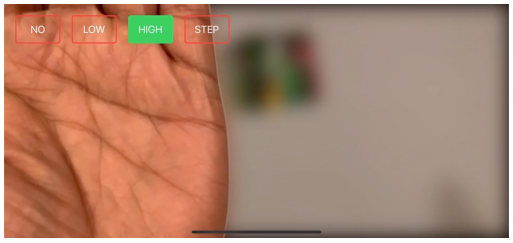

Figure 6. Blurring real world objects or high blur

5.1.1 Constant blur or low blur

In this technique, we applied a constant blur radius for objects outside irrespective of depth of the object. If the virtual object is outside the focal range, then apply a constant blur value. This technique has a problem, the virutal object blur does not match real objects blur which can break the illusion. This is illustrated in figure 5.

5.1.2 Blurring real world objects or high blur

Applying a constant blur, can break the illusion in cases where virtual object are in far plane in comparison to real world objects. Another technique we experimented is to apply a high blur to both virtual objects as well as real objects. This technique applied a double blur to already blurred objects which can break the illusion. This is illustrated in figure 6.

5.1.3 Step wise blur

Another novel approach is to apply different blur radius based on different depths. In this techinque we applied a linear blur radius for monotonically increasing depth, which means if the difference between focal range and depth crosses a threshold, apply a lower gaussian blur and for higher difference, apply a higher guassian blur. Since we apply, different blur values based depth, the virtual content blends well in the surroundings giving a perfect illusion. This is illustrated in figure 7.

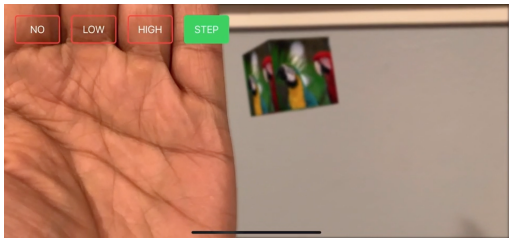

Figure 7. Step wise blur

Type	Value
Aperture	2.52
FNumber	f/2.4
Focal length	52mm
Pixel x value	4032
Pixel y value	3024
Sensor output	Bayer raw RGGB

Table 1. Technical specification for iPhoneX Back Telephoto camera

Type	Value
mean	0.078
standard deviation	1.623e-06

Table 2. Mean and Standard deviation for zero signal

5.2. Noise

We conducted our experiment on iPhoneX (https://www.apple.com/iphone-11/) using back telephoto lens. Table 1 shows the technical specification of the back telephotocamera and lens based on the EXIF metadata provided along with the captured sensor bayer raw image.

Table 2 shows the mean and standard deviation for black image for zero signal and max gain (10x).

Table 3 shows the standard deviation obtained from the calibration process. It is measured for the image captured for a white paper with manual gain.

Figure 8 shows an ARScene rendered to the camera background without noise. The camera background is an indoor low light scene that has visible noise.

Figure 9 shows the same ARScene rendered with noise.

6. User study

We conducted a brief user study to understand the effectiveness of our approach and to quantify how much did the user experience improve with these post effects. For blur, users were able to tell the difference between the 4 approaches and found the step wise blur blends well in the real

ISO speed rating	Standard deviation
16	1.315e-04
50	1.683e-04
100	2.135e-04
200	2.582e-04
300	1.345e-03
400	1.786e-03
500	2.325e-03
640	2.836e-03
800	1.682e-02
1000	1.853e-02

Table 3. Standard deviation for manual exposure

Figure 8. ARScene without noise

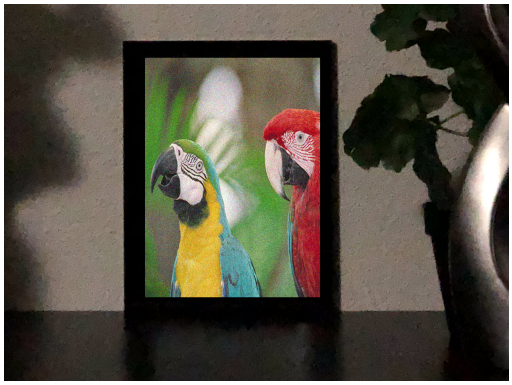

Figure 9. ARScene with noise

world scene. For noise, the users were able to notice the difference between applying noise and the case where no noise is applied to the augmented scene. The experiment also depends on the brightness of the ARScene and the brightness of the camera backdrop since it is easy to distinguish the ARScene if the brightness doesn't match against the camera backdrop.

7. Discussion

7.1. Depth of Field Blur

The blur present in the camera image is a limitation due to the narrow depth of field of the camera lens. For the camera sensor to produce appreciable image quality, the camera aperture should be open wider so that it can take more natural light. But, as the aperture size increases, the depth of field narrows. As the depth of field narrows, the camera can only focus on a limited range. So, the camera sensors present in mobile devices have auto focus feature where the lens moves to focus between near an far plane. The auto focus feature ensures that the camera is always focussed on the right object which is prominent in the scene. The natural blur of the out of focus area present in the camera sensor output depends on the lens characteristics and sensor characteristics. The camera lens exhibits different distortion model depending on the lighting condition, focus distance and the scene. So, different portion of the image has subtle different in the amount of natural blur. The current blur technique adopted in this project assumes equal blur for the entire rendered content. One way of extending this technique is to understand the lens and sensor characteristics and adjust the blur depending on the area where the rendered content is placed against the camera backdrop. Another extension would be to improve the edges of the rendered content. Also, the rendered content can also be transparent. If so, the rendered content can have different virtual object in the background. Or, it may have camera backdrop. The rendering engine applies pre-mul alpha rendering to figure out the value of the final pixel. The blurring after pre-mul alpha needs to take care of the surrounding pixels in the vicinity to give realistic blur effect.

Another major area of research in rendering is lighting and shadows. If an artificial light is added to the scene, the texture of the rendered object is changed based on the lighting. So, special care needs to be taken if we decide to blur the rendered content. Same applies to shadows where the shadow of the rendered content is drawn against the camera backdrop depending on natural and artificial lighting. The blurring of shadows need further study and analysis.

As an extension, the camera image also exhibits natural blur effect if the camera is shaken or moved. This is due to the rolling shutter effect where each pixels integrates over a period of time. This depends on the lighting condition. In low light condition, the sensor is exposed over a longer period of time causing larger motion blur even though the camera is focussed at the right distance. For this scenario, we can also apply blur to the rendered content to improve the user experience.

7.2. Noise

Noise coming from the camera sensor can be categorized into sensor black level amplification noise, sensor defect pixel noise, noise due to phase shifting of auto focus pixels, noise due to sensor heat dissipated in the circuit, sensor speckle noise. The noise characteristics for each category can be studied further to understand how noise shows up visually for different lighting condition. We have broadly classified noise into signal independent and signal independent noise. For the signal dependent noise, better calibration process under different lighting condition can lead to a more accurate noise modelling. Also, the noise characteristics is non linear as gain increases. In the current process, the standard deviation is calculated based on bi linear interpolation between known noise models. Further study can be done to see whether this modelling is effective. Apart from adding noise to the scene, the AR scene can have different brightness characteristics. So, adjusting the brightness of the ARScene can also help to improve the user experience. Also, this process can be extended to apply noise to each scene in camera preview instead of taking one image and applying noise to it. In that scenario, the noise texture applies each frame should be randomized because the noise present in the camera feed is not same for each pixel. A simple non uniform offset in both x and y direction can be used as a variable before calculating noise for each scene.

8. References

1. Dissecting the Camera Matrix, Part 3: The Intrinsic Matrix, http://ksimek.github.io/2013/08/13/intrinsic/

2. WWDC 2017, SceneKit: What's New, https://developer.apple.com/videos/play/wwdc2017/604/

3. Understanding Depth of Field – A Beginner's Guide, https://photographylife.com/what-is-depth-of-field

4. Hayato Ikoma, Michael Broxton, Takamasa Kudo Gordon Wetzstein, A convex 3D deconvolution algorithm for low photon count fluorescence imaging, 2018

5. Ryan D. Gow and all, A Comprehensive Tool for Modeling CMOS Image-Sensor-Noise Performance, 2007

6. Yeul-Min BaekJoong-Geun KimDong-Chan ChoJin-Aeon LeeWhoi-Yul Kim, Integrated Noise Modeling for Image Sensor Using Bayer Domain Images, Springer