FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

HDR Image Acquisition for Augmented Reality

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik

eingereicht von

Martina Rasch
Matrikelnummer 0925447

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Dipl.-Ing. Mag.rer.soc.oec. Martin Knecht, Bakk.techn.

Wien, 16.09.2013

(Unterschrift Verfasserin) (Unterschrift Betreuer)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

HDR Image Acquisition for Augmented Reality

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics

by

Martina Rasch
Registration Number 0925447

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Dipl.-Ing. Mag.rer.soc.oec. Martin Knecht, Bakk.techn.

Vienna, 16.09.2013

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Martina Rasch
Webergasse 5, 2225 Zistersdorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasserin)

Abstract

In this thesis I present a method for calculating high dynamic range images in a mixed reality system. Cameras and monitors usually have a lower dynamic range than we encounter in the real world, e.g. the sun. While pictures have a maximal contrast of 1:500, real world scenes often have a contrast of 1:100 000. An image taken of a scene with a higher dynamic range than our camera can capture will have regions that are too bright or too dark. With a higher exposure time more details will be visible in dark regions and with a lower exposure time more details will be visible in bright regions. Since our camera cannot create an image preserving details in both dark and bright regions we have to calculate one using the images our camera can actually produce. The method described in this thesis is based on the work of Debevec and Malik [2]. It takes several images taken with different exposure times and combines them to a high dynamic range image, leading to a better viewing experience in our RESHADE framework, a mixed reality framework, for which this method was implemented.

Keywords: high dynamic range image, mixed reality, tone mapping, characteristic curve, RESHADE framework

Contents

CHAPTER 1

Introduction

Light sources in the real world, like the sun, usually have a higher dynamic range between the highest luminance density and the smallest luminance density than monitors are able to produce and cameras are able to capture, since they usually operate in low dynamic range (LDR). This leads to images taken with cameras or shown on monitors to have lower contrast or regions which are too dark or too bright. However, for certain application scenarios, such as visually plausible mixed reality, it is important to know the high dynamic range (HDR) values in the scene in order to produce convincing results.

Since many cameras only have a limited dynamic range all pixels below or above a certain brightness value will be either black or white. Which pixels will be clamped to black or white depends on the exposure time the camera is set to. If an image is taken with a low exposure time, the details in the bright regions will be preserved and the dark regions will be nearly black. A high exposure time will lead to bright regions becoming white while preserving the details in dark regions. What we want to achieve is a picture which preserves the details in both the dark and the bright regions. Such an image is called HDR image, since it has a higher dynamic range than usual images.

In mixed reality setups HDR images can be used to estimate the correct lighting for objects virtually added to the scene. The term mixed reality is used for environments where the real world and a virtual world are merged, e.g. adding of virtual objects. For a convincing integration of virtual objects into a real scene realistic lighting is important. Virtual objects should be lit consistently with their surrounding objects and reflect light just like a real object would. However, for distant objects in the scene the light reflected from the new object would not make a big difference for the lighting of the scene object. Therefore it would suffice to know the radiance of the distant scene objects, which can be accurately detected by creating a HDR image of the scene. An image taken with a conventional LDR camera would lead to an incorrect estimation of the lighting since the capabilities of the camera are limited, as can be seen in Figure 1.1.

There are several methods for creating a HDR image. For example, a HDR image can be synthetically generated on a computer or a special HDR camera can be used which already

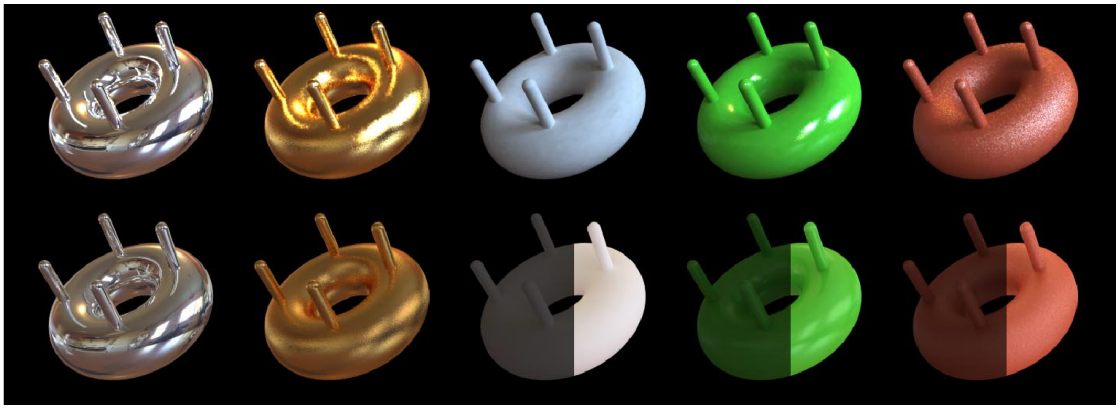

Figure 1.1: The images in the top row were synthetically illuminated using a panoramic radiance map and the RADIANCE global illumination algorithm [6], the bottom row was lit using a single LDR photograph. This image is taken from Debevec. [1]

creates a HDR image at capture time. Another way is to take several images with a LDR camera and to calculate a HDR image out of these images afterwards. Our approach belongs to the latter category and is based on the work of Debevec and Malik [2]. In this method several images are taken with different exposure times and combined to one HDR image. This way both the dark and the bright regions are visible in the same image. However, the images usually cannot be combined by just multiplying a factor to the brightness values, since most digital cameras apply a nonlinear mapping. To encounter this problem we calculate the characteristic curve of the camera and use it to correctly determine the brightness values.

Furthermore the HDR image usually cannot be viewed directly because monitors, like cameras, only work with a limited dynamic range. If the image would be displayed directly, again the pixel values below and above certain thresholds would be clamped to black or white. Scaling can be used to determine in which parts of the image details will be visible, but some black or white regions will remain. One solution that works well is to create an image which is closest to human perception. These algortihms are known as tone mappers.

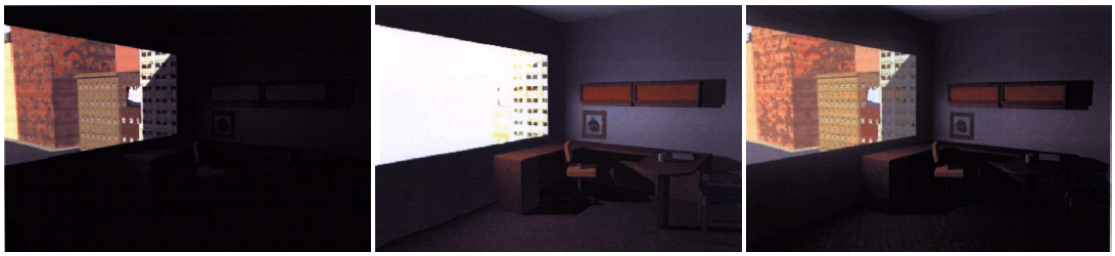

Figure 1.2: The left image is scaled in a way that the brightest pixel is the pixel with the maximum color value, the picture in the middle is scaled with a different scaling factor and the right image is tone mapped. The image is taken from Larson et al. [3].

In the following we will first present related work. Afterwards the method and the implementation are described. Finally we discuss the results of our work and how the images could still be further improved.

CHAPTER 2

Related work

There have been several publications concerning high dynamic range images. Many of them describe how to create a HDR image. Debevec and Malik [2] have proposed a method which allows to create a HDR image from several images taken with a conventional imaging device at different exposure times. To measure the true radiance values in the scene it is necessary to know how the radiance values are mapped to pixel values during the imaging process. Their paper describes how the response function of the imaging system can be obtained and how the HDR values can be calculated using the taken images. The mathematical explanation can be found in section 3.2.

Other papers discuss how HDR images can be displayed on conventional displays. Most displays only operate in LDR and therefore are unable to correctly display HDR images. Hence a method is needed to bring the HDR image into an appropriate color range. These algorithms are called tone mappers and usually take human perception into account. Larson et al. [3] came up with a tone mapper using histogram adjustment. They take advantage of the fact that luminance values of a scene are non-uniformly distributed throughout the dynamic range, resulting in clusters of luminance levels. Furthermore, the eye cannot distinguish absolute differences in luminance values. Under the condition that areas which were brighter than others in the scene are still brighter in the final image, the actual absolute difference values become irrelevant. Therefore gaps between occuring luminance values can be closed, which allows displaying a seemingly higher dynamic range than the monitor is actually able to reproduce. Larson et al. achieve this by approximating the brightness values with the logarithm of the luminance, building a histogram and a cumulative distribution function and using them to map the luminance values to display values. The result is further adjusted using human perception models. For instance, it is known that the human eye does not perceive changes in luminance values equally at different brightness levels. With the respective perception model the histogram is adjusted to preserve contrast.

Some papers show applications of HDR images. Debevec [1] and Nishina et al. [4] for example used HDR images to calculate the lighting of virtual objects merged into real world scenes. Debevec devides the scene into the distant scene, the local scene and synthetic objects.

In his method the HDR image is used to estimate the radiance of the distant scene. This is possible since the influence the light reflected from the virtual objects has on the appearance of the distant scene is very small and can be neglected. In this paper Debevec uses a light-based model by placing a steel ball near the location where the virtual object should appear and photographing it. Afterwards he maps the measured radiance values to the distant scene. With this method, some artifacts can occur, like photographing the camera and the photographer with the scene. Those artifacts can be reduced by taking several pictures from different directions and combining them or using editing software to remove them. Nishina et al. [4] calculate a HDR environment map and use it to render the reflectance of virtual objects. They use two cameras, one to capture the user's viewpoint and one to calculate a HDR image. Similar to the other papers the HDR image is created using several images at different exposure times. Meanwhile the virtual objects are geometrically registered using the user's viewpoint camera and special markers. For the shading and the shadows of the virtual objects the directions of the light sources are calculated. Moreover, for the reflectance of the virtual object an environment map is calculated using the previously calculated HDR image. At the end of each iteration the amount of images and the exposure times needed in the next iteration are estimated using the intensities of inappropriate black and white parts of the image and the maximum and minimum albedo value of the virtual object.

CHAPTER 3

Method

In this chapter we describe the method to capture a HDR image using a conventional LDR camera. To get HDR images in real-time we calculate them in a loop. Our algorithm takes images in a loop and in each iteration combines the new image with the combination of the previous images. The images are taken with different exposure times. When enough images were taken to get a correct HDR image or the limit of the camera was reached, the combined image is declared as the finished HDR image and the loop starts the calculation of the next HDR image.

3.1 The main HDR capture loop

In our loop to calculate a single HDR image first an image of the scene is taken. This image will probably contain pixels which are too bright or too dark. Therefore we define two thresholds, one defining which pixel values are too bright and one defining which are too dark. Then the minimum and the maximum color values of the image are calculated. If the maximum values are above the upper threshold another picture is taken with the halved exposure time. This step is repeated until the maximum color values of the new image are below the upper threshold. Next the exposure time is resetted and the algorithm checks if the minimum values are below the lower threshold. If they are, the exposure time is doubled and another picture is taken. Again this step is repeated until the minimum values are above the lower threshold. This procedure is visualized in Figure 3.1.

3.2 Combining the images

Whenever a new picture is taken it is combined with the previous images with exception of the first image of the loop. The procedure is the following: If the exposure time was halved before taking the new picture, because the maximum values where above the upper threshold, all the pixels with RGB values over the threshold from the new image are taken. The pixels with values

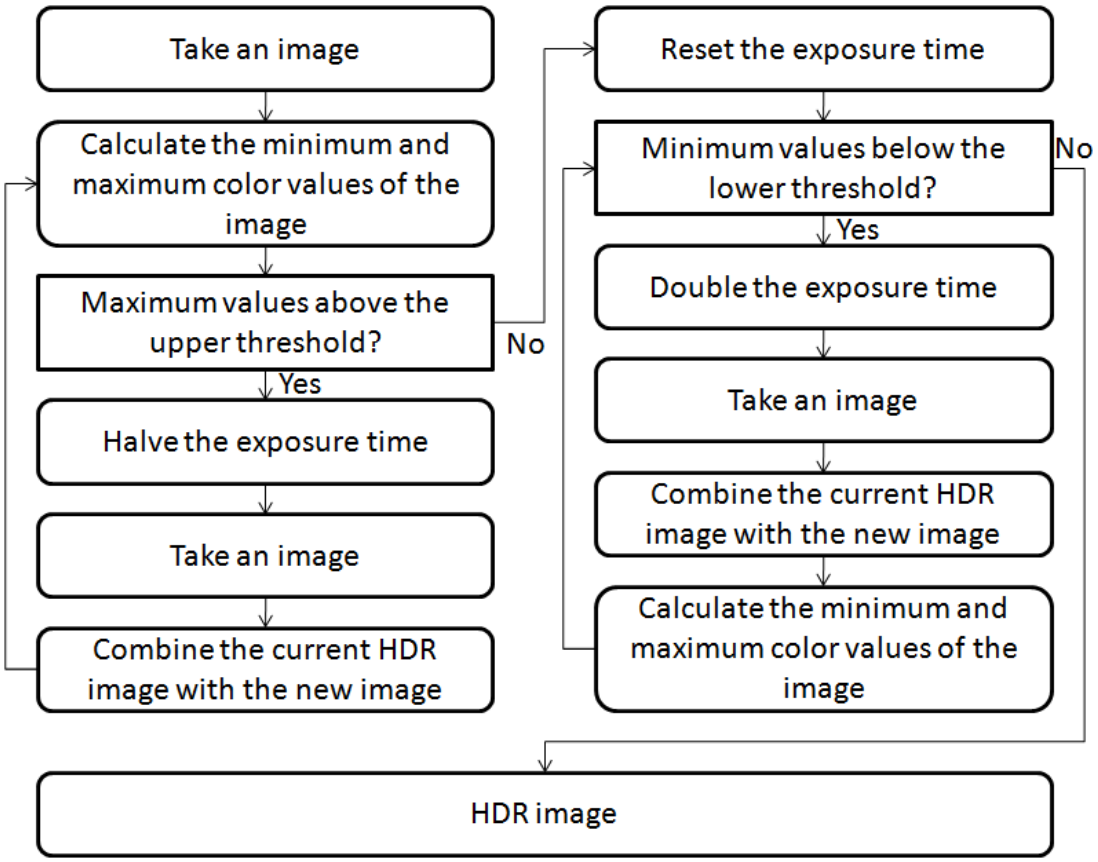

Figure 3.1: This figure visualizes the loop for taking a new HDR image. First images are taken with low exposure times to cover the bright areas. Then the process is repeated with high exposure times to capture the dark areas.

below the threshold are taken from the combination of the previous images (or the first image, if this is the second image taken). If the exposure time was doubled, the pixels which are taken from the new image are those that are below the lower threshold. Since the images are taken with different exposure times, the pixel values from the new image are not simple copied to the combined image. Instead a new value is calculated for the HDR image using the pixel value from the new image and the characteristic curve of the camera.

Characteristic curve

Since the images are taken with different exposure times, the RGB values cannot be simply taken from the two images and written to the new image. Multiplying the values of one image with the factor by which the exposure time differs between the two images also does not work. The reason is that most digital cameras apply a nonlinear mapping. When taking a picture, the camera measures the exposure X for each pixel. This equals the irradiance E multiplied with the

Figure 3.2: The left image visualizes how the images are combined to one HDR image. The green pixels are the pixels taken from the first image of the loop, the red pixels are from images taken with a lower exposure time which cover the bright areas, the blue pixels are from the images with higher exposure times which cover the dark areas. The brighter the color, the lower the exposure time and the higher the luminance of the scene. The right image is a picture of the scene taken with the same exposure time as the first image and should serve as a reference.

exposure time X = E · Δt. After processing the data the camera saves the pixel value c for X. Since the mapping is usually nonlinear, twice the pixel color value does not mean the camera measured twice the irradiance.

The correct brightness values can be calculated if the function of the camera is known. Debevec and Malik describe a method to calculate the characteristic curve of the camera in [2]. They take several images taken with different exposure times as an input. Z_{ij} stands for the pixel values, where i is a spatial index over pixels and j an index for the exposure times. The film reciprocity equation is Z_{ij} = f($E_i \Delta t_j$). After inverting f, taking the natural logarithm of both sides and simplifying ln f^{-1} as g, we get: g(Z_{ij}) = ln E_i + ln Δt_j. g(Z_{ij}) is the characteristic curve we want to find. This problem can be solved in a least squared sense.

Instead of calculating the characteristic curve we used the tool $HDRshop^{©}$ [5] to generate it. We used images differing by half an fStop as an input and got a matlab file with the values of the camera curve. The values needed for our HDR image are the values of the irradiance E. They can be calculated as $E_i = e^{g(Z_{ij}) - ln \Delta t_j}$.

3.3 When to stop changing the exposure time

The available exposure times are limited due to the capabilities of the camera. If the camera is set to a value which is too high or too low the exposure time of the camera will not actually change. Hence the algorithm stops halving or doubling the exposure time if it exceeds the exposure range of the camera.

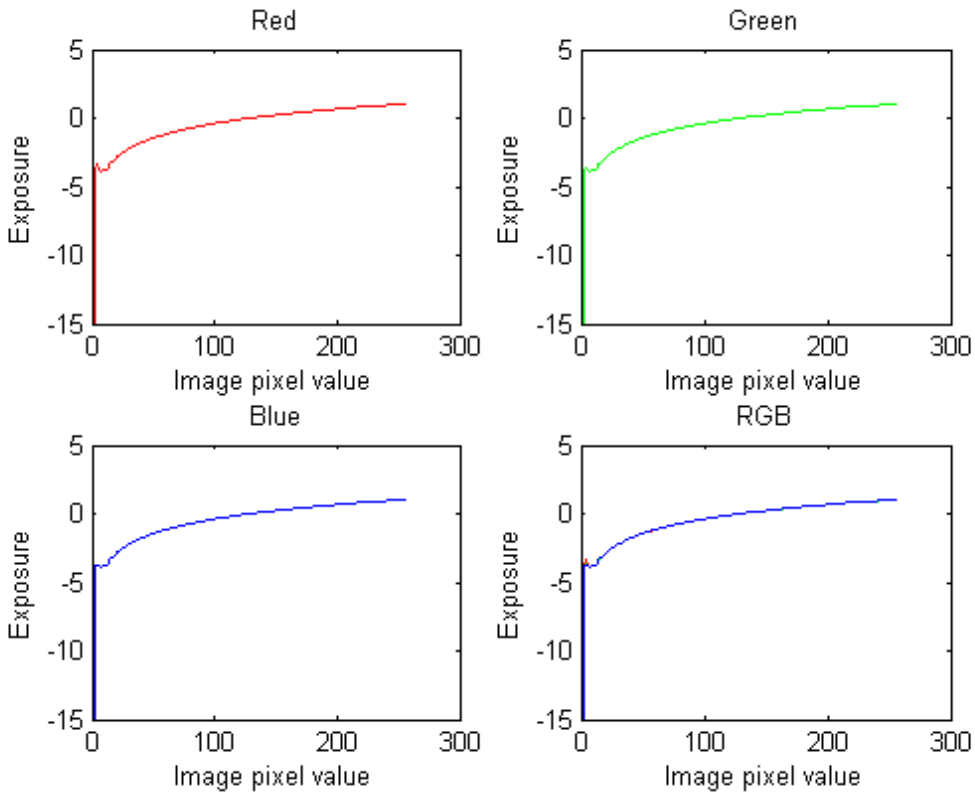

Figure 3.3: Characteristic curve of the camera, the first 3 graphs show the response functions of the red, green and blue color channel, the last graph shows an overlay of all 3 color channels

CHAPTER 4

Implementation

This section describes how the method is implemented in the RESHADE framework which is written in C# and uses DirectX.

4.1 The camera

The camera used for calculating the HDR image is a uEye camera from IDS with a fisheye lens attached. Our framework supports two modes as an enumeration: In *normal mode* the images of the cameras are displayed unaltered, in *HDR mode* a HDR image is calculated before displaying it on the screen. The images displayed on the screen are saved in a 2D Texture called rgbTexture. This is usually a texture in B8G8R8A8_UNorm format with the color values in the range between 0 to 255. When the mode is changed to HDR, the texture is replaced with a texture in the R32G32B32A32_Float format. This is necessary since we need more than just 256 different brightness values for HDR images.

4.2 Important variables

For our implementation several textures are needed. The Texture2D rgbTexture is the one holding the final image. The texture rgbTexture2 is only needed in *HDR mode*. The new image is copied to this texture instead of copying it to the rgbTexture which is displayed on the screen. During the calculation the current HDR image is saved in an array of two ping pong textures called hdrTextures. The term ping pong textures describes two textures that are used for a calculation on the GPU by taking one texture as an input and the other one as an output, then swapping and using the other one as input and the first as an output and so forth until the calculation is finished (see Fig 4.1). Ping pong textures are also used for calculating the minimum and maximum color values of the pixels, called minTextures and maxTextures. The characteristic curve of the camera is saved as a 1D texture containing 256 floating point triples.

Additionally two thresholds are defined: an upper threshold and a lower threshold. These will be used to determine which pixel values will be replaced by values from the new image.

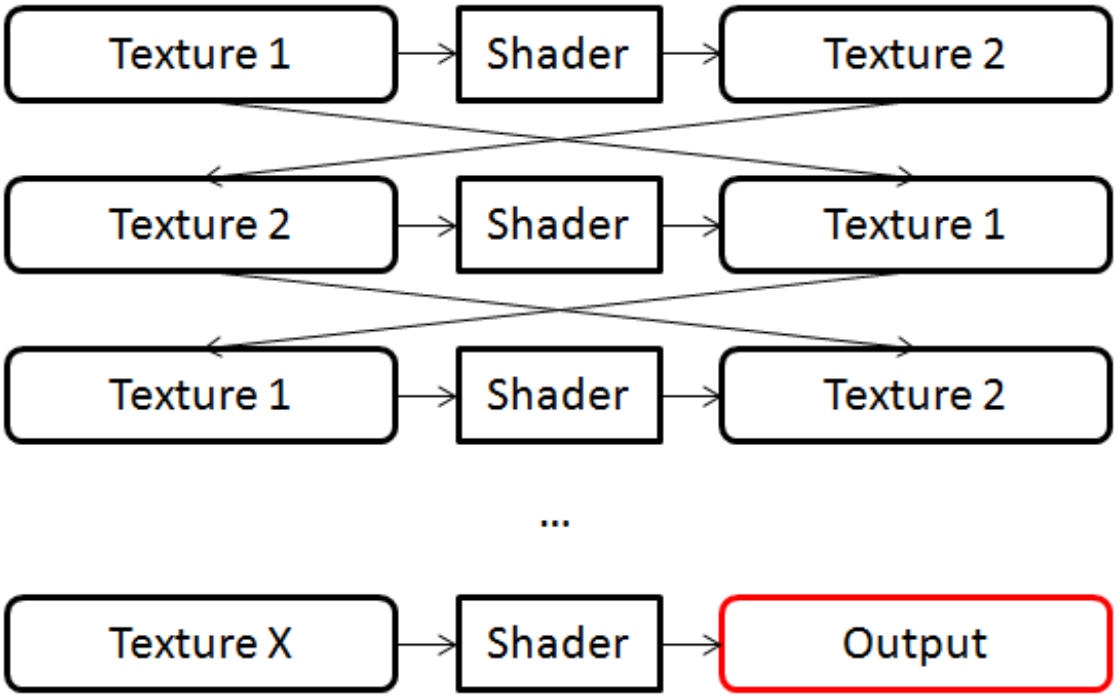

Figure 4.1: This figure shows how ping pong textures are used.

4.3 Calculating the minimum and maximum values

First the image is copied to the ping pong textures. The minTextures are used for calculating the minimum, the maxTextures for calculating the maximum. The calculations are performed in a HLSL shader. The values are calculated by taking four values of the image, calculating the minimum and the maximum of these values and writing it to the other texture, but one mipmap level higher. Then the minimum and the maximum values from this texture one mipmap level higher are calculated and written to the first texture, but another mipmap level higher. This is repeated until the mipmap level with the width and height of one pixel is reached. This pixel contains the wanted minimum and maximum values. Finally the minimum and the maximum are read from the respective textures.

4.4 Calculating the HDR image

Since the RESHADE framework shows a video and not just a picture, the HDR images are recorded in a loop. At the beginning an image is taken and copied to the rgbTexture2. Next the minimum and maximum values are calculated. If the maximum values are above the upper

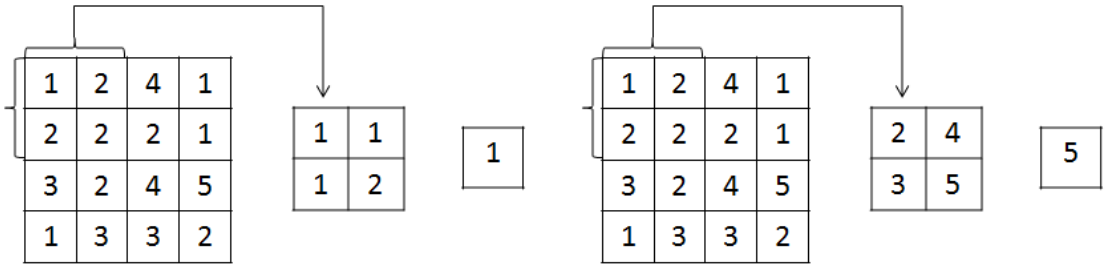

Figure 4.2: Calculating the minimum and maximum values

threshold, the exposure time is halved. Else if the minimum values are below the lower threshold, the exposure time is doubled. In the case that all the values are between the two thresholds the image is copied to the rgbTexture and the next HDR image is calculated.

While the maximum values are above the upper threshold new images are taken with the halved exposure time. Then the new image is combined with the combination of the previous images in a HLSL shader. All the values above the upper threshold are taken from the new image in rgbTexture2, the other values are taken from the previous image. To combine the images two ping pong textures called hdrTextures are used. One of them holds the previous image while the other one is the render target of the shader. After combining the images they are swapped. Then the algorithm checks if the values are still above the upper threshold. This is repeated until the values are not above the upper threshold anymore or the exposure range of the camera is exceeded. In this case it is checked if the minimum values are below the lower threshold. If they are above the threshold the image is copied from the current hdrTexture to the rgbTexture and all values are resetted.

If the values are below the lower threshold a similar procedure as the one where they are above the upper threshold is started. First a new image is taken with the doubled exposure time. Then the two images are combined. This time the values below the lower threshold are taken from the new image. Like before this is calculated using the hdrTextures and a HLSL shader. As soon as all the values are above the lower threshold or the exposure time is above the maximal exposure time of the camera the finished image is copied to the rgbTexture and the next HDR image is calculated.

4.5 Problems on the border

Due to the fact that the camera used for the calculation has a fisheye lens attached, the image of the camera does not always fill the whole texture. How much of the texture is covered depends on the camera settings. Often the image of the camera is only in a circle in the middle of the texture and the border is black. If all the values of the texture would be used for calculating the minimum values, the result would always be 0 (black). To prevent this we define the center and the radius of a circle and use only the texture values inside this circle for the calculation.

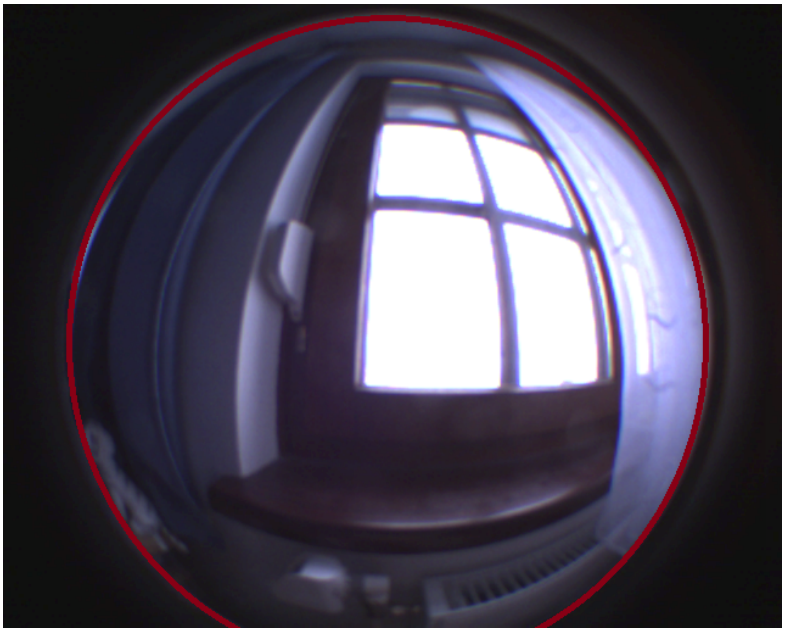

Figure 4.3: Since the image is black appart from a circle in the middle, we define a circle and use only the pixels inside of this circle for the minimum and maximum calculation

4.6 Displaying the image

Since most monitors only have low dynamic range, the HDR image will not be displayed with the correct brightness values. The best solution is to use a tone mapper.

CHAPTER 5

Results

The following images show the results of calculating HDR images with a uEye camera in the RESHADE framework. Figure 5.4 and 5.6 are HDR images produced with our implementation, figure 5.3 and 5.5 show the images used in the process. Unfortunately our implementation induces artifacts in the images, especially in bright regions and on the edges between image areas taken from different images.

Since our monitors only work with LDR the HDR images cannot be viewed directly. Therefore we dislplayed them using the formula log(multiplication factor * pixel color value + 1) and different values for the multiplication factor. This is not the optimal way of displaying HDR images, but it serves our purpose. However, this formula is not able to reproduce an image with details visible in all areas. Because of that figure 5.4 and 5.6 show several images instead of just one HDR image. A better tone mapper would enhance the visual quality of the image even more.

For our first HDR image showing a living room pictures taken with the exposure times of 0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32 and 64 ms were used in the calculation. The HDR image was displayed with the multiplication factors 1, 10, 25 and 50. The HDR image is shown in figure 5.4 and the corresponding image sequence in figure 5.3.

The next image sequence shows the view out of a window. The images were taken with exposure times of 1, 2, 4, 8, 16, 32 and 64 ms, for the displayment of the HDR image multiplication factors of 5, 25, 50 and 100 were used.

For a better comparison figure 5.1 shows a single image of the camera taken at an exposure time of 16 ms and a HDR image displayed with a multiplication factor of 50. In the HDR image the dark areas appear darker and bright appear brighter. In this specific image the details can only be seen in the dark areas, since the used tone mapping is not optimal. Figure 5.4 though shows that using a different multiplication factor will make details in other areas visible. Additionally the colors are more saturated in the HDR image. The artifacts probably have a negative effect on the aesthetics of the HDR image, but many people find HDR images visually more appealing than LDR images.

Figure 5.1: A comparison between a single picture taken with the exposure time 16 ms on the left and a HDR image display with the formula log(50 * pixel color value + 1) on the right.

Figure 5.2: The left image is a HDR image of Debevec and Malik [2], the right image is a HDR image calculated with our implementation.

Compared to the results of the algorithm of Debevec and Malik [2], the images do not look as nice, especially because of the artifacts, but they can be dynamically generated. Figure 5.2 shows both an image of Debevec and Malik [2] and of our implementation.

Figure 5.3: A sequence of images used to calculate the HDR image in figure 5.2. The images were taken with the exposure times of 0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32 and 64 ms.

Figure 5.4: A HDR image of a living room. It was displayed using the formula log(multiplication factor * pixel color value + 1) and the multiplication factors 1, 10, 25 and 50.

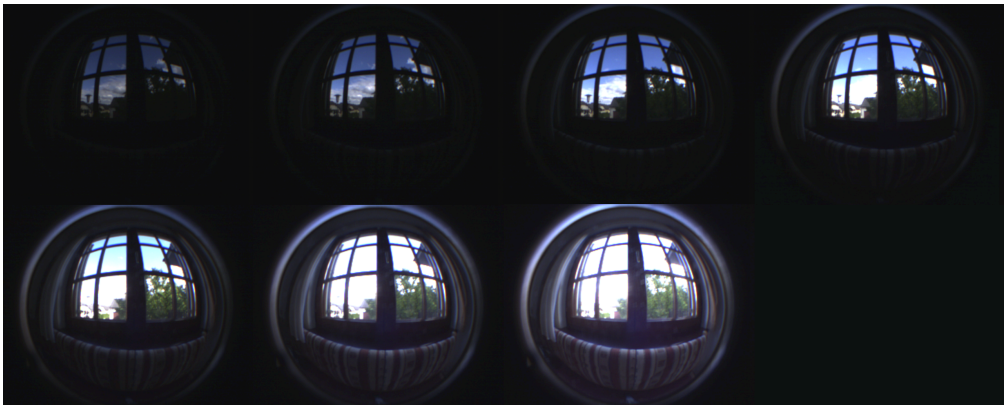

Figure 5.5: A sequence of images used to calculate the HDR image in figure 5.4.

Figure 5.6: A HDR image of the view out of a window. It was displayed using the formula log(multiplication factor * pixel color value + 1) and the multiplication factors 5, 25, 50 and 100.

CHAPTER 6

Conclusion

We presented a simple method for calculating a HDR image in a mixed reality system by taking several images with different exposure times and combining them to a HDR image. The result are HDR images in our RESHADE framework. Ideally it should be combined with a better tone mapper to create an even better viewing experience.

The implementation could still be improved by getting rid of the artifacts which appear mostly in bright areas or in on the edges between pixels calculated from differently exposed images. Another possible improvement is the calculation of the minimum and maximum values. Since the fisheye camera often produces pictures with the taken image in a circle in the center and black pixel values on the border, we defined values for the center and the radius of the circle and just used values inside the circle for the calculation. For these values we simply chose values that seemed appropriate. This could be improved by dynamically calculating which pixel values actually belong to the image of the real world.

CHAPTER 7

Appendix

7.1 Example shader codes

```
1  PS_OUT output = (PS_OUT)0;
2  float4 rgb = float4(rgbTexture.SampleLevel(pointSampler, input.uv, 0).rgb, 1.0f);
3  if rgb.x >= treshold && rgb.y >= treshold && rgb.z >= treshold then
4  |   float4 hdr = float4(rgbTexture.SampleLevel(pointSampler, input.uv, 0).rgb, 1.0f);
5  |   float r = cameraCurve.SampleLevel(pointSampler, hdr.r, 0).r;
6  |   float g = cameraCurve.SampleLevel(pointSampler, hdr.g, 0).g;
7  |   float b = cameraCurve.SampleLevel(pointSampler, hdr.b, 0).b;
8  |   hdr.r = exp(r-exposureTime);
9  |   hdr.g = exp(g-exposureTime);
10 |   hdr.b = exp(b-exposureTime);
11 |   output.hdrColor = hdr;
12 else
13 |   output.hdrColor = float4(hdrTexture.SampleLevel(pointSampler, input.uv, 0).rgb,
   |   1.0f);
14 end
15 return output;
```

Algorithm 7.1: Combining 2 images

```
1  PS_OUT output = (PS_OUT)0;
2  float3 minValue = float3(1.0f, 1.0f, 1.0f);
3  float3 maxValue = float3(0.0f, 0.0f, 0.0f);
4  if sqrt(pow(camCenterX - (input.uv.x * windowWidth + 0.5), 2) + pow(camCenterY -
   (input.uv.y * windowHeight + 0.5), 2)) < camRadius || mipmaplevel > 0 then
5      minValue = minTexture.SampleLevel(pointSampler, float2(input.uv.x + 0.5 / width,
       input.uv.y + 0.5 / height), mipmaplevel).rgb;
6      maxValue = maxTexture.SampleLevel(pointSampler, float2(input.uv.x + 0.5 / width,
       input.uv.y + 0.5 / height), mipmaplevel).rgb;
7  end
8  if sqrt(pow(camCenterX - (input.uv.x * windowWidth - 0.5), 2) + pow(camCenterY -
   (input.uv.y * windowHeight + 0.5), 2)) < camRadius || mipmaplevel > 0 then
9      minValue = min(minValue, minTexture.SampleLevel(pointSampler, float2(input.uv.x
       - 0.5 / width, input.uv.y + 0.5 / height), mipmaplevel).rgb);
10     maxValue = max(maxValue, maxTexture.SampleLevel(pointSampler,
       float2(input.uv.x - 0.5 / width, input.uv.y + 0.5 / height), mipmaplevel).rgb);
11 end
12 if sqrt(pow(camCenterX - (input.uv.x * windowWidth + 0.5), 2) + pow(camCenterY -
   (input.uv.y * windowHeight - 0.5), 2)) < camRadius || mipmaplevel > 0 then
13     minValue = min(minValue, minTexture.SampleLevel(pointSampler, float2(input.uv.x
       + 0.5 / width, input.uv.y - 0.5 / height), mipmaplevel).rgb);
14     maxValue = max(maxValue, maxTexture.SampleLevel(pointSampler,
       float2(input.uv.x + 0.5 / width, input.uv.y - 0.5 / height), mipmaplevel).rgb);
15 end
16 if sqrt(pow(camCenterX - (input.uv.x * windowWidth - 0.5), 2) + pow(camCenterY -
   (input.uv.y * windowHeight - 0.5), 2)) < camRadius || mipmaplevel > 0 then
17     minValue = min(minValue, minTexture.SampleLevel(pointSampler, float2(input.uv.x
       - 0.5 / width, input.uv.y - 0.5 / height), mipmaplevel).rgb);
18     maxValue = max(maxValue, maxTexture.SampleLevel(pointSampler,
       float2(input.uv.x - 0.5 / width, input.uv.y - 0.5 / height), mipmaplevel).rgb);
19 end
20 output.minValue = minValue;
21 output.maxValue = maxValue;
22 return output;
```

Algorithm 7.2: Calculating the minimum and the maximum

Bibliography

[1] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 189–198, New York, NY, USA, 1998. ACM.

[2] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[3] Gregory Ward Larson, Gregory Ward Larson, Holly Rushmeier, Holly Rushmeier, Christine Piatko, and Christine Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3:291–306, 1997.

[4] Yusaku Nishina, Bunyo Okumura, Masayuki Kanbara, and Naokazu Yokoya. Photometric registration by adaptive high dynamic range image generation for augmented reality. In *7th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2008, Cambridge, UK, 15-18th September 2008*, pages 53–56. IEEE, 2008.

[5] USC Institute for Creative Technologies. http://www.hdrshop.com. Accessed: 2013-08-02.

[6] Gregory J. Ward. The radiance lighting simulation and rendering system. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 459–472, New York, NY, USA, 1994. ACM.