International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 8, Number 2, pages 205–217, July 2018

HDR Image Processing from Embedded Cell Phone Camera

Andrej Mihálik
Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava
842 48 Bratislava, Slovakia
Email: andremihalik@gmail.com

Pavol Kunovský
Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava
842 48 Bratislava, Slovakia
Email: palo.oneill@gmail.com

and

Roman Ďurikovič
Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava
842 48 Bratislava, Slovakia
Email: durikovic@fmph.uniba.sk

Received: January 18, 2018
Revised: April 22, 2018
Accepted: June 5, 2018
Communicated by Susumu Matsumae

Abstract

In photo-realistic image synthesis the incoming light from the environment is particularly important. In this work we focus on capturing the incoming light in the form of the radiance map using a common mobile device. This involves the reconstruction of the spherical panorama sky-dome in high dynamic range (HDR) quality and save it to a usable data format. Our setup requires a common smartphone devices with an additional fish-eye lens attached to the camera. In this paper we discuss the calibration of our setup and the implementation of selected tone mapping operator (TMO) allowing satisfactory display of HDR images on the mobile device screen. Built-in cameras on mobile devices do not generally capture the HDR image. In this work, we describe an algorithm for capturing an HDR image on the Android platform. Using an optimization method we are able to acquire the camera response curve needed in the reconstruction of the HDR image from multiple snapshots. Subsequent HDR panoramas represent radiance sphere-maps of the incoming light from an environment. These radiance sphere-maps are useful in realistic image synthesis to illuminate the objects in the 3D scenes.

Keywords: HDRI capturing, Android panorama, Camera response curve

1 Introduction

Realistic image synthesis is a prevailing scope in computer aided visualization. Fields like architecture, products design and prototyping or entertainment industry significantly benefit from 3D graphics capabilities. A crucial aspect of photo-realistic image synthesis is real world lighting. In this work we attempt to capture the light condition from arbitrary environments by capturing high dynamic range (HDR) images using a mobile device. More specifically, it is a reconstruction of HDR spherical panorama sky-dome from images captured by a mobile photo camera. Subsequently, the resulting HDR panorama is saved in a usable data format. Contrary to the RGB format, where values range from 0 to 255 for each channel, an HDR image contains arbitrary floating point values. The HDR image can be visualized by a suitable tone-mapping operator on the mobile display. Later output data can be used to illuminate objects in a 3D scene.

When creating a spherical panorama from multiple images, we have to choose the proper technique of blending all the images needed. Unlike the previous methods we avoid the complex stitching technique [1]. To create a spherical panorama, we will use the fish-eye lens that captures the 360o $\times$ 180o scene.

2 Previous Works

There are a lot of similar HDR image creation applications. Most of them, however, run on computers that have the necessary performance and capacity. We are trying to create an HDR spherical panorama directly on a mobile phone. Previous solutions to this type of application are not that many. We found the article [6] that focuses on creating an HDR spherical panorama on a mobile device powered by NVIDIA Tegra. The purpose of their application is to allow the user to easily capture the spherical HDR panorama and interact with it. Using OpenCV with Stitcher class, they effectively optimized the calculations to take place as quickly as possible. Image alignment resolves using Inertial Measurement Unit (IMU) sensors to help determine the location of the camera against previous photos and thus speeds up the algorithm. The HDR spherical panorama was created with individual HDR images. The advantage was that the spherical panorama created was of high resolution. To calculate the response curve, they chose an algorithm that iteratively modulates the response curve based on the article [7].

Creating the spherical panoramas by stitching the multiple images is a common technique. However, one panorama requires 30 to 40 snapshots to be taken, which creates particular memory requirements for the embedded device. Considering the HDR images, the memory consumption triples. Stitching algorithms are also computationally intensive, therefore not suitable for low end mobile devices.

Another solution is to send a sequence of images from the device to the processing server. The mobile device is not used in calculations and the total processing time takes a few minutes.

The article [8] presents a new approach to creating panoramic mosaics from image sequences. Unlike the current panoramic stitching method, which typically requires a clear horizontal panorama of the camera, their system does not require any controlled movements or constraints when used. The algorithm presented is very simple and robust because it directly uses 3D rotation instead of the general 8 parametric transformation of the imaging plane.

3 Sky-dome capturing work flow

Here we propose the essential steps of our method as described in the following pipeline, see Fig. 1 In the first step, we should take at least three RGB panorama snapshots that are stored in a temporary folder. The next step is the camera calibration, see Sec. 5, where we choose the appropriate number of random pixels from the central image area to calculate the camera response curve and set the static camera parameters. The third step is the conversion step of RGB panorama snapshots to HDR values, see Sec. 4.2. Finally, we get a single HDR panorama image exported to the output

HDR format, see Sec. 7. Global tone mapping operator is used to project the HDR values to the mobile display RGB space and to save it as JPG image.

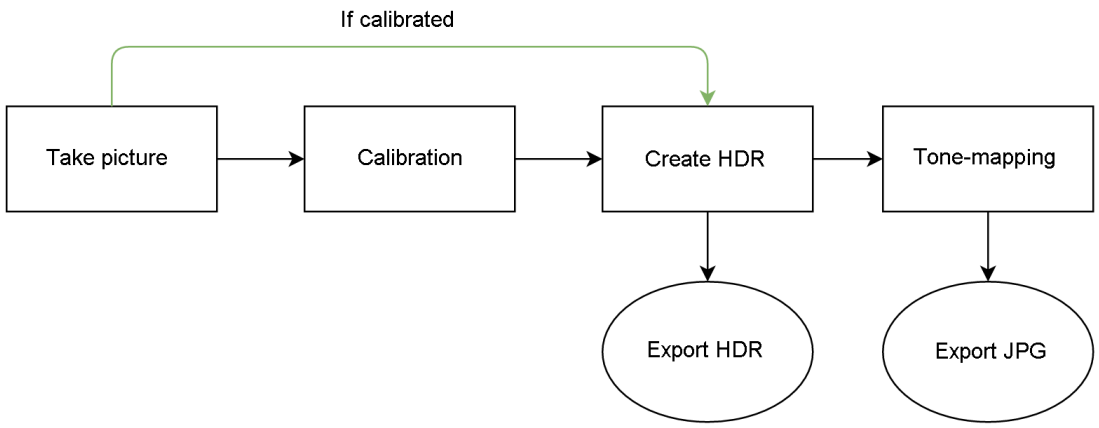

Figure 1: Pipeline for capturing the HDR panorama on a mobile device with few shots.

4 Response curve

The HDRI technique attempts to record a larger range of radiance and increase the resulting quality in the dim parts of the image. The resulting photo can be displayed after processing with the tone-mapping operator or stored in a usable Radiance RGBE HDR data format. To take a single spherical panorama image, it is best to use a fish-eye lens (fish eye) that captures the $360° \times 180°$ scene. By this way we avoid the necessity to stitch particular images to form a single panorama RGB image. If we want to create an HDR image, we need to capture the scene with all levels of radiance. This could be achieved by capturing a sequence of photos with different exposures. Some photos will have underexposed or overexposed spots. We need at least three frames with different exposure to create the HDR image on a mobile device.

4.1 Capturing multiple exposures

The camera should be set to capture multiple exposures. This means that all parameters, except for the shutter time, are constant. The relevant parameters that must be same for all frames are level of sensitivity (ISO) and white-balance. When shooting, the mobile device should be laying on a flat surface and it should not move.

4.2 Obtaining HDR Intensity from RGB Color

When we capture an image from the camera, the RGB values do not truly represent the radiance. If one pixel of captured image has twice the brightness value of another, it is unlikely that it has twice the radiance in real. There is usually a nonlinear mapping that determines how radiance in the scene becomes an RGB value. Most significant nonlinearity in the camera response curve [2] is at its saturation point, where any pixel with a radiance above a certain level is mapped to the same maximum RGB value. This produces images in limited dynamic range, where we have to choose the range of radiance values that are of interest and determine the exposure time suitably. To produce HDR image with accurate radiance values, we can incorporate the multiple exposure approach to our acquisition system. The device's camera during the process of measurement has fixed focal ratio, white balance and level of sensitivity (ISO). The only variable parameter is exposure time. We make

three shots per sample with variable shutter time. Since, our purpose is to capture environment light conditions, only details in low, mid and high exposure are sufficient to construct the sky-dome.

The varying exposure values (EV) are set to -2, 0 and 2 for each sample. Since the camera response curve is not linear, we have to acquire the response curve beforehand. Computation of the response curve is required only once for every device by capturing three photos and specifying one hundred random pixel position in the image. For each position, we fetch three-pixel values from three captured photos. All calculations are performed in particular RGB color channel separately. We have three photos indexed by $i \in \{1, 2, 3\}$. Each photo is taken with different exposure time Δt_i. For the particular pixel position at the coordinates (u, v), the camera response curve is defined as a function:

$$f(E_{u,v}\Delta t_i) = I^i_{u,v}, \tag{1}$$

where $I^i_{u,v}$ is the pixel value in the i-th photo. E is radiance incoming to the camera aperture from the particular point. We assume that function $f(x)$ is smooth and monotonic. Pixel values are integers ranging from 0 to 255. To acquire the function $g(I^i_{u,v}) = ln\ f^{-1}(I^i_{u,v}) = \ln E_{u,v}\Delta t_i = \ln E_{u,v} + \ln \Delta t_i$, we solve a linear least squares problem of minimizing the following quadratic objective function:

$$\begin{aligned} \sigma &= \sum_{u,v}\sum_{i=1}^{3}(g(I^i_{u,v}) - \ln E_{u,v} - \ln \Delta t_i)^2 + \\ &+ \sum_{z=1}^{254}(g(z-1) - 2g(z) + g(z+1))^2. \end{aligned} \tag{2}$$

The first term of the equation is the fitting function and the second term is the approximation of the second derivative of g in the discrete form it ensures that the function g is smooth. The consequent system of linear equations is solved using the singular value decomposition (SVD) method.

The response curve is estimated for each color *(Red, Green, Blue)* channel separately. The method requires at least two images to process, but more images give better results due to noise sensitivity. The input images should meet common shooting conditions such as stabilization, ISO value, aperture, white balance, and focus on all images.

To speed up the process, instead of summing over all pixels coordinates (u, v), we randomly choose one hundred pixels within the image. To ensure even distribution of pixel positions, we use 10×10 grid to divide the photo. Then we choose a random pixel position from each cell. Finally, we can calculate the incoming radiance in particular pixel position by the following equation:

$$E_{u,v} = e^{\frac{1}{3}\sum_{i=1}^{3}(g(I^i_{u,v}) - \ln \Delta t_i)}. \tag{3}$$

We apply this equation to each color channel separately.

4.3 Fish-eye

A fish-eye lens is a wide-angle lens that produces visual distortion intended to create a wide panoramic image. In our experiment, we have used a clip-on lens (see Fig. 2) that can capture a range of 235° and works with any device lens, as long as its no larger than 13 millimeters in diameter.

5 Calibration

5.1 Camera setup

Our mobile camera should be set to the unchanged settings during shooting, so we've chosen these settings as follows:

- ISO, the international standard for sensitivity determination value set to 200.
- The aperture is set to F / 2.6.

Figure 2: Fish-eye lens attachment.

- We set the white balance to Cloudy day light.
- Focus is set to manual.
- Exposure time is selected with aperture priority.

5.2 Response curve

As we mentioned in Sec. 4.2, since the camera response curve is not perfectly linear, we need to calculate it by an optimization method, which sets the weights for individual pixels, so that the average exposed pixels are more important than pixels that are close to the beginning and end of the range. Since the mobile devices are limited in performance, we need to optimize these calculations. One-time calibration is required before the calculation of response curve for particular camera. Selected number of input samples are captured in order to perform the calibration process. In order to calculate the response curve efficiently, we select 100 random pixels from the inside of the spherical panorama marked by the circle mask evenly distributed in a regular grid, see Fig. 3.

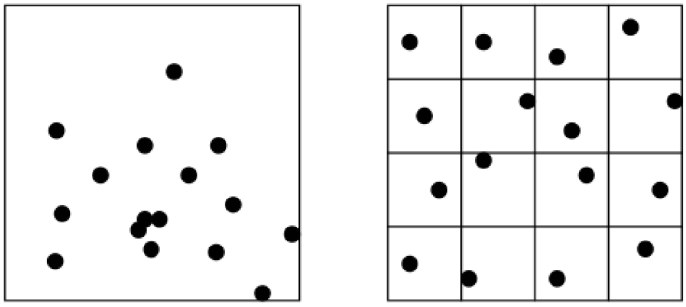

Figure 3: Calibration samples selected from the inside of the spherical panorama. Left: Random selection. Right: Evenly distributed random sample selection in a regular grid.

Since we capture natural environment rather than uniform patterns, we utilize random sample distribution. In our case, we chose a 10×10 grid inside the circle mask. If we have 100 random pixels selected, one random pixel is selected in each cell. If we had the maximum of 600 random pixels, six random pixels would be selected in each cell. The grid is dynamically calculated based on the user's resolution of the output image.

6 Clipping

In this section, we estimated the mapping function for the fish-eye lens to determine the radius we want to crop the image so that the resulting spherical panorama is 180°. This step has to be repeated for newly attached lens.

The scale we used measured angles and was applied to the transparent box, see Fig. 4. The height of the box was 10.5 cm, the base was 12.4×9.9 cm and the centre of the box was $S = [6.2; 4.95]$. The scale is oriented in a way that 0° is at the top of the box (above the lens) and the centre of the projection is located at the point S at 2.3 cm above the base. The fish-eye lens captured the

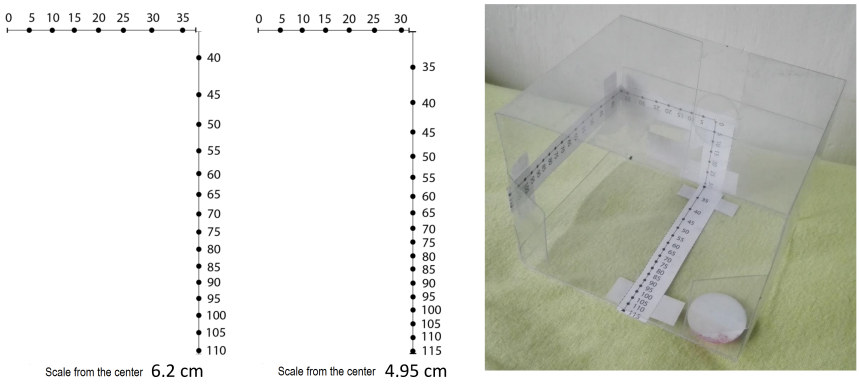

Figure 4: Scale in angles.

image approximately at an angle of $100° - 105°$, see Fig. 5. Since, we do not know the mapping

Figure 5: Image captured inside the box. The shown scale is in degrees.

function of our fish-eye lens, we have to find the mapping to hemispherical coordinates. We assume that mapping is a stereographic projection. Figure 6 shows the mapping of a box (in green) onto the lens and then on the planar image sensor. The circle center S (lens) represents the sphere center to which the image is mapped through the center of the projection S. The south pole N on the sphere is the center of another projection, that maps the points projected on the sphere to the plane (sensor) α. The box is labeled as K. Based on this mapping, we've obtained a new scale (blue points) that we've embedded in our photo captured from the box.

Spherical panorama of 180° is needed as an output format. Unfortunately, the fish-eye lens has the angular field of view (FOV) 235° and multiple reflections inside the lens. We need to crop the picture by a circle to 180° FOV, but first, we need to find the mapping from our fish-eye lens to 180° FOV. It was done based on the projection measurements of the transparent box with the lens.

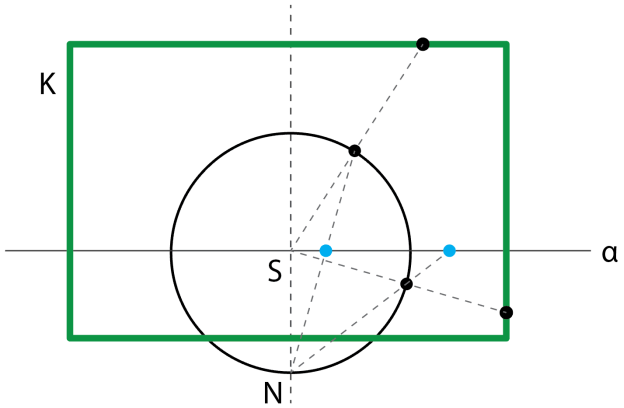

Figure 6: Mapping of the fish-eye.

We can calculate the resolution of photo camera and resolution of a mobile phone display and then determine the radius of the circle boundary in pixels that corresponds to 180° view angle instead of 235° view. We display the clipping boundary as the inner circle representing the recalculated portion that displays 180° view, where the outer circle represents a portion of the fish-eye lens, see Fig. 7. The inner circle forms the circular mask used to crop the image.

Figure 7: Clipping. Left: Clipping area is limited by inner circle while outer circle corresponds to the visible fish-eye area of 235° view. North direction is marked as a blue dot and the sun position is marked as a yellow dot. Right: The final clipping mask.

7 HDR calculation

In order to create the HDR image on a mobile device, we need to store three photographs with different exposure time. Each pixel of the resulting HDR image is calculated using the following equation:

$$E_{u,v,j} = e^{\frac{1}{P}\sum_{j=1}^{P}(g(Z_{uvj})-\ln \Delta t_j)}, \tag{4}$$

were $E_{u,v,j}$ represents the (u, v) pixel brightness for particular color channel, $g(Z_{uvj})$ is the response curve for the particular color channel at pixels with coordinates (u, v) in the exposed image with index j and the $\ln \Delta t_j$ is shutter time. The number of input photos is $P = 3$.

7.1 Output Format

The output HDR image is stored in the Radiance RGBE image format that stores 4 bytes for each pixel. Three bytes store the mantissa for each color channel (red, green, blue) and fourth-byte stores the shared exponent for all channels. The advantage of this format is that it can store the higher range of pixel values. Due to the use of a shared exponent, we do not lose precision with bright and dark pixels.

Listing 1: Example of HDR header

```
#?RADIANCE
#SUN=-Y 0 +X 0
#NORTH=-Y 259 +X 82
FORMAT=32-bit_rle_rgbe
-Y 2448 +X 2448
```

We placed additional information into the header (see Listing 1). When capturing the HDR image, we record the date and time, the position of the sun (SUN) on the sky and the orientation of the HDR image relative to the North (NORTH) orientation. The sun's position is defined by a user double-clicking on the display at given location. The sun is displayed as a yellow dot. The North is captured based on the mobile phone orientation sensor. The sensor returns value 0 for North, 90 for East, 180 for South and 270 for West. From the sensor, we choose the azimuth that is between the north and show the north point location.

Listing 2: Conversion of north coordinates to the position of blue dot

```
float azimuth = event.values[0];
North.x = (float)((width / 2) + ((height / 2) - sizeOfCircle)
* Math.sin((double)(-azimuth)/180*3.143));
North.y = (float)((height / 2) - ((height / 2) - sizeOfCircle)
* Math.cos((double)(-azimuth)/180*3.143));
```

We can project the azimuth coordinates x and y to outer circle in spherical panorama, see Listing 2. North is marked as a blue dot on the device display, see Fig. 7.

7.2 Tone-mapping

Tone-mapping allows us to display the HDR image on the device screen. First, we have to find out maximum and minimum intensity value for each color channel in the HDR image. Subsequently, we can normalize each pixel value for the particular color channel in the HDR image. The remapping of a value $x \in [min, max]$ to the interval $[0, 1]$ is done by $f(x) = \frac{x-min}{max-min}$. Normalized values are then multiplied by the value of 255 to get the pixel values in the image format using 8-bit per color channel.

Listing 3: Tone-mapping code snippet.

```
Mat img = Mat.zeros(hdr.size(),hdr.type());
for(int i = 0; i < hdr.rows(); i++) {
 for(int j = 0; j < hdr.cols(); j++) {
  double[] rgb = hdr.get(i,j);
  rgb[0] = ((rgb[0] - minR)/(maxR - minR))*255;
  rgb[1] = ((rgb[1] - minG)/(maxG - minG))*255;
  rgb[2] = ((rgb[2] - minB)/(maxB - minB))*255;
  img.put(i,j,rgb);
 }
}
```

8 Implementation

We have implemented the method, with the Android Studio [4] development environment, including the OpenCV [5] library and JAMA [9] math library to solve the optimization using the Android 4.1.2 operating system.

The application interface consists of a screen showing the image from the camera and the main menu. We have shown a green and grey circle at the image from the camera demonstrating the clipping according to the fish-eye lens, yellow paint (sun) and blue point (north pole estimation).

The application captures three photos with exposure -2, 0 and 2 and saves the current sun position, orientation, and image preview resolution of the cell phone in memory. These photos are then used to create an HDR image or to calibrate camera response curve. The HDR image is generated provided the smathphone's camera response curve. After the HDR image is created, a tone-mapping algorithm is applied to prepare the image for a preview.

9 Results and Validation

Several examples of captured spherical panoramas with our method in HDR format on the mobile device are shown in Fig. 8.

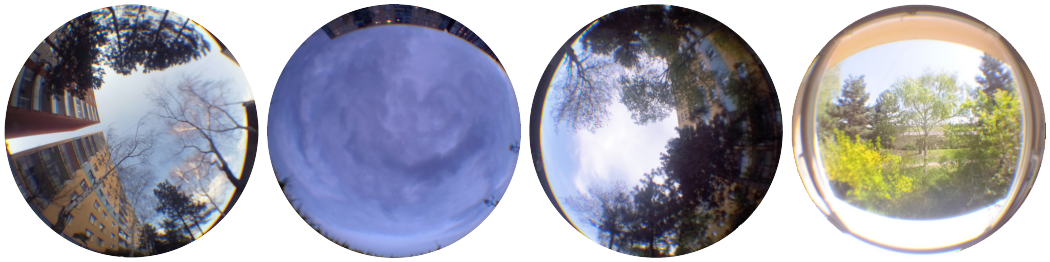

Figure 8: Resulting HDR spherical maps with resolution 2448x2448 pixels after application of "tone-mapping" for visualization purposes.

In this section, we make comparisons between the Matlab application and the mobile application on the mobile device. Since computational power of embedded devices is limited we prototyped our algorithms in Matlab environment and then optimized it to the mobile device purposes. We set constraints such as the number of input pixels in mobile version and compared resulting curves with Matlab output. We were optimizing the pixel setting for the response curve, based on a comparison response curve in a mobile application and the application in Matlab environments. Then we visually compared HDR image panoramas.

9.1 Measurement of external influence

We generated 3 HDR images in a row with our method, and measured the difference between them by means of the mean quadratic error. The difference in the HDR images could have been caused by various external factors like wind, clouds, light, air movement or slight movement of objects. If the external factors are constant then the error images should be the same.

In the test, we searched for a difference between images one and two, and to compare the difference between images one and three. The motion picture test was done indoors without the fish-eye attached (see Fig. 9, 10) and outdoors (see Fig. 11, 12) with the fish-eye attached. The standard deviation between images was computed using the following formula:

$$\rho = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - x_i')}, \tag{5}$$

where N is the total number of pixels in the image, while x_i and x_i' is the intensity of the particular pixel in the first and the second image respectively. Note that while using the fish-eye, the deviation may be even lower due to the stretching and cropping of the image.

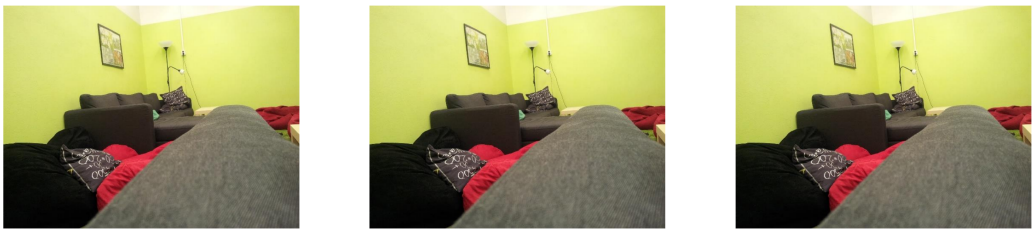

Figure 9: Indoors images taken in a short time span.

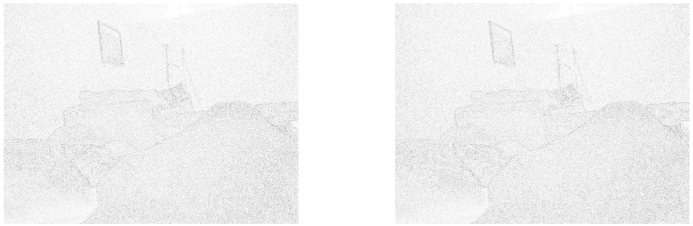

Figure 10: Difference images. Left: The difference image between first two above images with standard deviation $\rho = 0.0164$. Right: The difference image between the first and third image with standard deviation $\rho = 0.0113$.

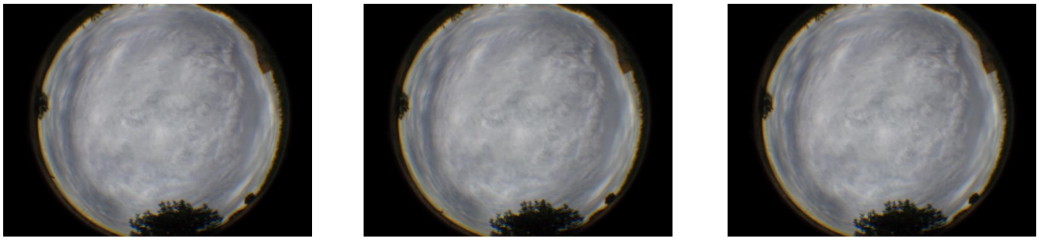

Figure 11: Outdoors HDR images were taken in a short time span.

The result of this test was that the deviation in both the interior and the exterior is for the purpose of our application negligible, and we can use individual images to compute response curves and also to create an HDR image. Time between shots is too short to significantly distort the actual image, so this does not affect the computation of broad environment light conditions.

9.2 Response curves for different input random pixels

As input for the response curve, we gave a different number of random pixels. We have evaluated the robustness of the response curve fitting different number of random pixels and visualized the resulting HDR image after the "tone-mapping" algorithm.

We tested our algorithm on a desktop computer using the Matlab environment and compared the response curves for 100, 300, 400 and 600 random input pixels from different images while visualizing the response curves. We observed that the method works well and is robust if we use at least 100 pixels for estimation of the response curve based on the observation.

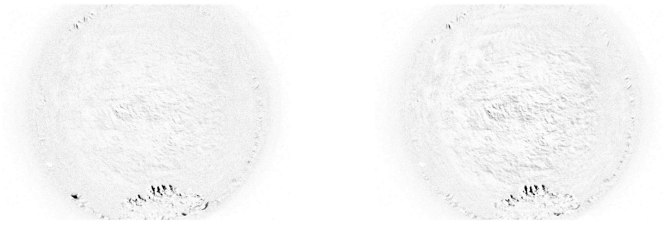

Figure 12: Difference images. Left: The difference image between first two outdoors images with standard deviation $\rho = 0.0149$. Right: The difference image between the first and third image with standard deviation $\rho = 0.0142$.

Then we render the HDR images using 100 random pixels in the mobile version and compared with the the same image in the Matlab environment giving a standard deviation less than 0.015.

With less than 100 random pixels, we noticed color bleeds. The colors in the resulting HDR image after applying the "tone-mapping" algorithm in Fig. 13 did not fit with real colors at all.

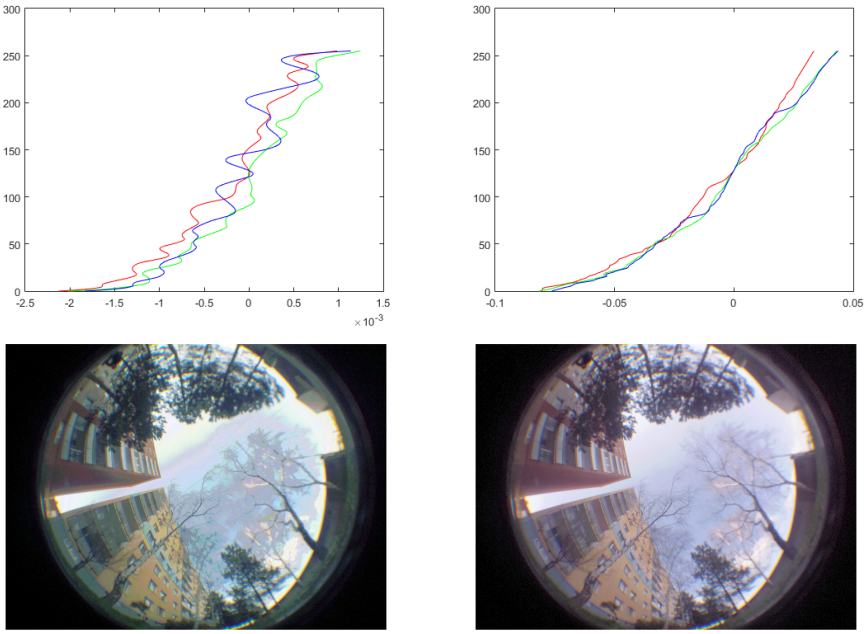

Figure 13: Response curves for different input random pixels. Top left: The response curve estimated based on 30 random pixels. Top right: The response curve estimated based on 400 random pixels. Bottom row: The HDR spherical panorama calculated with response curve for 30 and 400 random pixels, respectively.

10 Conclusion

We have developed the mobile sensor capturing the radiance map of a sky-dome environment using a common mobile device. Captured incoming light is particularly useful in the material appearance simulation or the entertainment industry. With the additional surface reflectance measuring techniques [10], we are able to display 3D objects under arbitrary lighting. This allows us to add

objects into captured scenes while preserving realistic appearance [6]. Our portable method is highly flexible and allows us to capture lighting in virtually all environments. In order to handle all the calculations of the mobile device we estimate the camera response curve from a few randomly selected points. Captured panoramas containing HDR data are utilized to produce radiance sphere-maps (see Fig. 14) for realistic image synthesis [3] displaying objects in the 3D scene under particular illumination. Finally, we have validated the method by comparing the final HDR images to the off-line Matlab implementation.

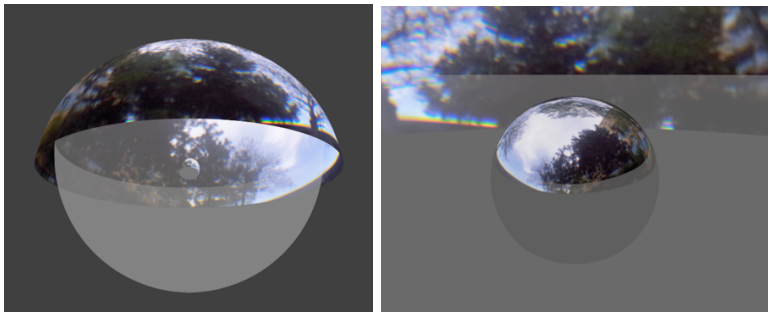

Figure 14: Applying captured HDR data as a radiance map to the 3D scene in the computer image synthesis. Left: Our panorama mapped to the spherical environment map. This environment map serves us as the light source in the form of the sky-dome. Right: Environment map applied to the spherical metallic object in the 3D scene.

References

[1] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, Aug 2007.

[2] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, pages 31:1–31:10, New York, NY, USA, 2008. ACM.

[3] P. Dutre, P. Bekaert, and K. Bala. *Advanced Global Illumination, Second Edition*. CRC Press, 2016.

[4] Google. Android Studio. `http://developer.android.com/sdk/index.html`. 1. july 2017.

[5] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015. 30. jun 2017.

[6] Peter Kán. Interactive HDR Environment Map Capturing on Mobile Devices. In *Eurographics 2015 - Short Papers*, pages 29–32. The Eurographics Association, 2015.

[7] Mark A. Robertson, Sean Borman, and Robert L. Stevenson. Dynamic range improvement through multiple exposures. In *In Proc. of the Int. Conf. on Image Processing (ICIP99*, pages 159–163. IEEE, 1999.

[8] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 251–258, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[9] The Mathworks Inc. JAMA: A Java Matrix Package. `http://math.nist.gov/javanumerics/jama`. 28. jun 2016.

[10] Roman Ďurikovič and Andrej Mihálik. Modeling the brdf from spectral reflectance measurements of metallic surfaces. 312:87 – 90, 09 2014.