Estimating the Camera Response Function of an iPhone 15 Pro to Create New HDR Images

Audrey Lee, and Lycia Tran

Abstract—As technology continues to improve, the want for images to better replicate the human vision grows. High Dynamic Range (HDR) images are images that try to replicate what the human eyes should see by balancing lighting within an image. In recent years, the average person is able to use their mobile phone to capture images, and phones themselves, have been able to create their own HDR images. However, these HDR images do not seem to be significantly better or more balanced. For this project, we propose to estimate the camera response function of the most accessible digital camera – a smartphone – and create our own HDR image pipeline to produce balanced images that can be more interchangeable and modifiable to suit different people's tastes easier to make a more appealing final image.

Index Terms—Computational Photography, High Dynamic Range Imaging, Camera Response Function, Tone Mapping

1 Introduction

Digitized photographs are becoming increasingly important in computer graphics. More than ever, scanned images are used as texture maps for geometric models, and recent work in image-based modeling and rendering uses images as the fundamental modeling primitive. Furthermore, many of today's graphics applications require computer-generated images to mesh seamlessly with real photographic imagery. Properly using photographically acquired imagery in these applications can greatly benefit from an accurate model of the photographic process.

With the rise in use of digital cameras through smartphones, the access to digital photographs and images have become easier. This means that photographs can be taken from different types of cameras and can easily be shared from one device to another. However, all of these cameras have different pipelines to capture and process images.

High Dynamic Range (HDR) images are images that balance lighting within an image to more accurately match what is perceived by the human eye. In recent years, the average person uses their mobile phone to capture images, and while most phones seem to have their own "HDR Mode", these HDR images do not seem to be significantly better or more balanced [1].

Our project aims to estimate the camera response function of the most accessible digital camera – a smartphone – and create our own HDR image pipeline to produce balanced images that can be more interchangeable to suit different people's tastes in a final image. Our project aims to recover the RAW iPhone images by estimating the camera response function of an iPhone to produce our own HDR image pipeline

2 Related Work

Initially, we started with Takamatu's method to recover HDR images using noise variance in a set of images with the same camera settings and exposure values [2]. However, as we looked more into this we chose not to move forward with this method due to the reliance on already having a good estimation of the camera response function in order to solve for the unknowns. For this reason, we had to switch to another method, Debevec's method, to accurately recover our camera response function and our irradiance values [3].

Using Debevec's method on recovering the camera response function, we wanted to see if we could recover the camera response of an iPhone 15 Pro given its limited range of exposure times and camera settings. Debevec's method requires using a series of aligned images of the same scene with each image having different exposure times. The images should range from being fairly underexposed to fairly overexposed. With this method, we are able to utilize a property relating the exposure times to the digital pixel value on what you see in the camera. With this property, we can say that there is a non-linear mapping relating the exposure, X, to the digital value you see on the camera screen, Z. Our exposure value, X, is the product of the irradiance the camera receives, E, and the exposure time, Δt. And our output pixel value, Z_{ij}, is the output of our camera response function, f, for the i^{th} pixel in the j^{th} image of the image series. Since we know the value of Z, we can find the inverse of that camera response function based on our camera response function, f. [3]

$$Z_{ij} = f(\Delta t_j E_i) \tag{1}$$

We can assume that f is invertable, so we can take the inverse and natural log of both sides to get:

$$\ln f^{-1}(Z_{ij}) = \ln \Delta t_j + \ln E_i \tag{2}$$

Finally, we can define g to be the natural log of the inverse of f, so g

$$g(Z_{ij}) = \ln \Delta t_j + \ln E_i \tag{3}$$

We can then put this inverse camera response function, g, into a properly weighted irradiance values equation. We can then establish a scale factor, w, to estimate the pixel values

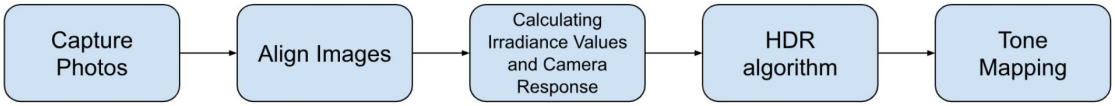

Fig. 1. Flowarchart Diagram of Implementation

Fig. 2. Original set of images with the following exposure times: 1/1 s, 1/5s, 1/13 s, 1/25 s, 1/60 s, 1/100 s.

relative to the minimum and maximum pixel values. If our Z values are closer to the minimum or maximum, $g(Z_{ij})$ will have a steeper curve, so we can introduce a smoothing factor λ to fit the data more. This results in the following objective function:

$$O = \sum_{i=1}^{N}\sum_{j=1}^{P}\{w(Z_{ij})[g(Z_{ij}) - \ln \Delta t_j - \ln E_i]\}^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}[w(z)g^H(z)]^2 \quad (4)$$

We can then solve for our inverse response function, g, as well as our irradiance values, E_i, by framing it as a linear least squares problem to minimize our objective equation and obtain our irradiance mapping and inverse camera response function.

Using our recovered response curve, g, we can now convert our pixel values, Z_{ij}, to relative radiance values. These values will be our linear HDR values, HDR_{lin}, which we can then apply a tone mapping to. We can use an adopted variation of Debevec's method to recover these relative radiance values, HDR_{lin}, with equation 5 [3]:

$$\text{HDR}_{lin_i} = exp(\frac{\sum_j w_j(log(I_{lin_{i,j}}) - log(t_j))}{\sum j w_j}) \quad (5)$$

$I_{lin_{i,j}}$ is our linearized Low Dynamic Range (LDR) value for pixel i, image j. We want to give higher weight to exposures where the pixel value is closer to the center of our response function, thus we can calculate our weights using equation 6:

$$w_{i,j} = exp(-4(\frac{(I_{lin_{i,j}} - 0.5)^2}{0.5^2})) \quad (6)$$

Using equations 5 and 6, we can recover our linearized HDR values. This gets us an HDR image that shows more of the brightness as shown in Figure 6. However, since this HDR image is a 32-bit image, we can use various tonemapping to properly scale and display our images in an 8-bit display while still keeping the high contrast details.

3 Proposed Method

As shown in Figure 1, our pipeline involves aligning images we took from capturing photos with different exposure times, calculating the camera response function for this iPhone camera, creating our own HDR image pipeline, and creating our own tonemapping methodology.

3.1 Camera Response Function Implementation

Using Debevec's method [3], we first loaded the following set of images with the following camera settings: ISO 500, 24 mm, f/1.8. We took 6 images of the same scene using the following exposure times: 1/1 s, 1/5 s, 1/13 s, 1/25 s, 1/60 s, 1/100 s as shown in Figure 2. We used the "ProCam" iOS application to be able to adjust these camera settings, since the defualt Camera application on the iPhone does not allow the adjustment of most of these settings, and would make the project impossible especially if we did not have control nor knowledge of what exposure times we were taking the photos with.

Since we are limited in the amount of compute power and memory we have, we sampled the same are of 50 evenly spaced pixels across each of our original images. These pixels are the used to solve equation 4. The only remaining unknown we have besides $g(Z_{ij})$ and E_i is our smoothing term, λ. To find λ, we solved equation 4 for various different λ values as seen in the plot in Figure 3 and chose the smoothest inverse response curve.

From the plots, we see that with $\lambda = 1000$, we get a smooth response. We can use this to get back our camera response function, f, seen in Figure 4. This resulting camera response function matches what we would expect to see. Camera response function tend to follow this logorithmic shape due to the fact that the human eye can see details in the shadows or dark regions of an image better than we can see details in the highlights or bright spots of an image.

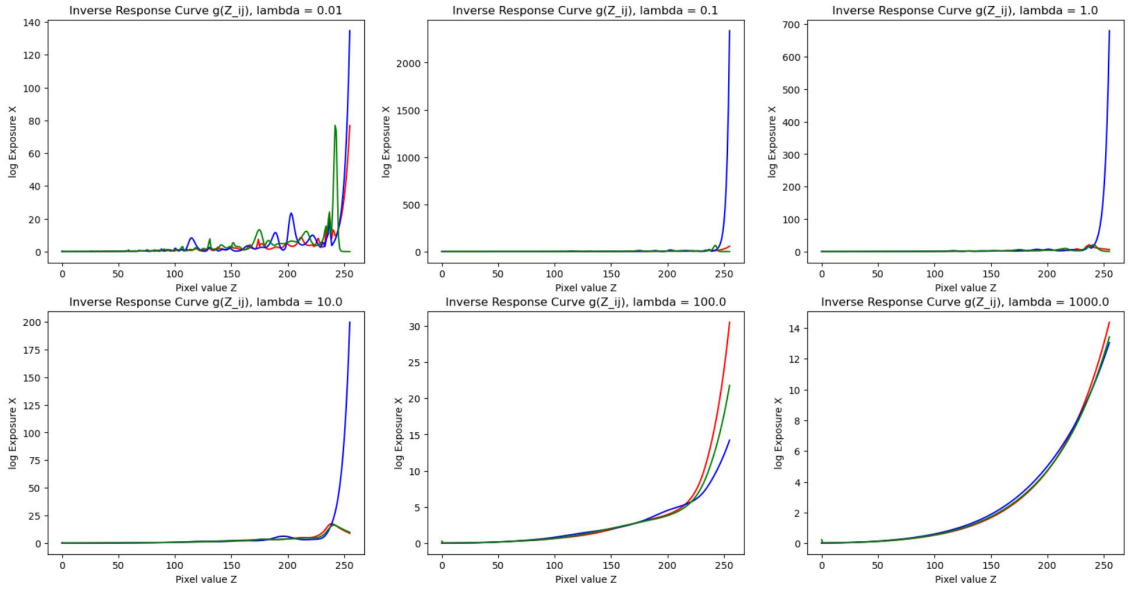

Fig. 3. Inverse Camera Response Function, $g(Z_{ij})$ for different smoothing factors, λ

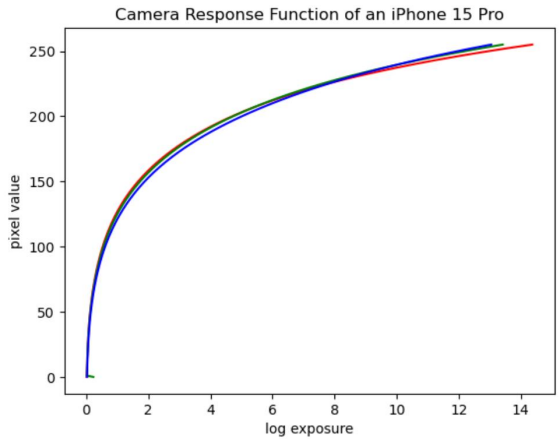

Fig. 4. Camera Response Function for λ = 1000

This is why the camera response function curve has a much steeper slope for lower exposure values, but flattens out for higher exposure values.

Using a λ value of 1000, we can use the corresponding inverse camera response curve we will use to recover our linear LDR images. For each of our original images, we can map our pixel values (Z_{ij}) to our linear LDR pixel values to get the following images shown in Figure 5.

Using the results of this response curve, we can now map our original images back to linearized LDR images which are shown in Figure 5. We use these pixel values, I_{LDR}, to recover our linear HDR pixel values, I_{HDR}, using equations 5 and 6. The resulting image, shown in Figure 6, will be the merged image of all the LDR images weighted based on the confidence that the pixel is well exposed.

3.2 Tonemapping Implementation

After obtaining our linearized HDR image, we can now apply some tonemapping algorithms like Drago's tonemapping [4], Mantiuk's tonemapping [5], and our tonemapping algorithm with the results shown in Figure 7. Tone mapping will allow us to scale the 32-bit HDR image values back into an 8-bit LDR display while still preserving high-contrast image details.

The first tone mapping algorithm we implemented was the equation below where s is our scaling factor and γ is used for gamma correction

$$I_{HDR} = (s * I_{HDR_{lin}})^{\gamma} \tag{7}$$

Although this performed relatively well, we noticed that we could improve our tonemapping algorithm through a use of a bias term. This term would help with the ability to alter the images easier for people who want more control over how their HDR image would look. For our bias term, b, we wanted to see if we could improve our final HDR image. We noticed that there was a bias term used in Drago's tonemapping [4], but instead, we applied it directly to our linear HDR pixel values giving us the following equation:

$$I_{HDR} = (s * I_{HDR_{lin}}{}^{\frac{log(b)}{log(0.5}})^{\gamma} \tag{8}$$

A plot comparing an image of a building at night using our original tone mapping algorithm and our original tone mapping algorithm with bias can be seen if Figure 8. In this figure, we can see a clear difference between the image with bias and the one without.

To test if our camera response function worked with another image dataset, we tested it on a set of images of Stanford University's Hoover Tower. We used the camera response function we found earlier to map these images to linearized LDR images to create our HDR image. We also

Fig. 5. Linearized LDR Images

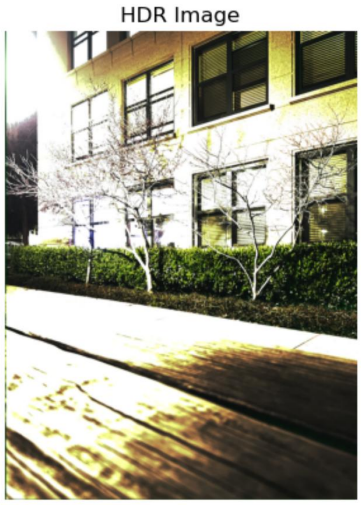

Fig. 6. Resulting linear HDR image

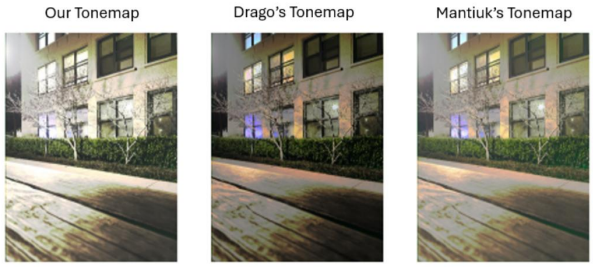

Fig. 7. Tonemapped Images

applied the same tonemapping as was done on the previous images, but we also chose to modify our previous tonemapping by adding a bias term in our original tonemapping. By adding the bias term we are able to see slight improvements in our resulting HDR image.

We now compare our new tonemapping to Drago's tonemapping, Mantiuk's tonemapping, and finally Apple's HDR image pipeline. We used OpenCV's implementation of Drago's and Mantiuk's tonemapping. These images of Stanford University's Hoover Tower can be seen in Figure 9.

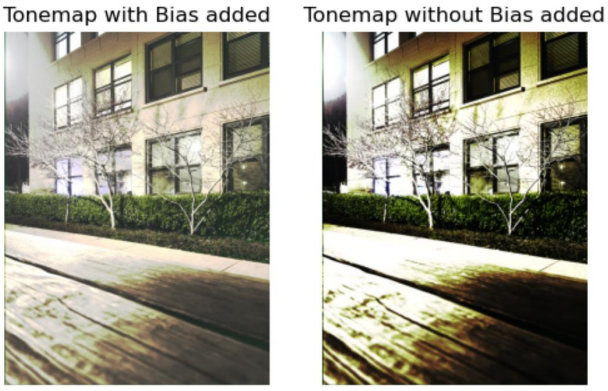

Fig. 8. (Left) Tonemapped image with bias (Right) Tonemapped image without bias

4 Experimental Results

Figure 8 shows the comparison between our original tonemapping without the added bias and our new tonemapping with the added bias term. The images as we can see are vastly different. Our tonemapped image with bias added has more more evenly distributed lighting, and preserves the image details much better than our tonemapped image without bias. This is especially noticable in the bottom right corner of the image where the shadow on the table is. In the tonemapping without bias, this spot is almost completely dark and no details in the table can be seen. While in the tonemapping with bias, we can still see some details in the table, such as the wood grain pattern. We also see that the tonemapping with bias has a much more realistic coloring as compared to the tonemapping without bias. This bias term allows us to create more realistic HDR images and preserve details within in our image better when mapping our 32-bit linear HDR image back to an 8-bit display image.

We also compared the results of new tonemapping algorithm to Drago's tonemapping, Mantiuk's tonemapping, and Apple's iPhone 15 Pro's HDR image in Figure 9. For Drago's tonemapping, we used a γ value of 1, a saturation value of 0.7, and a bias value of 0.85. For Manituk's, we used a γ value of 2.2, a scale value of 1.2, and a bias value of 0.85. In these images, we see that the iPhone's HDR image is cooler toned, while our HDR images are warmer toned. This is due to our white balance setting when capturing our original photos. We chose to capture warmer tone photos, since we tended to prefer warmer images over

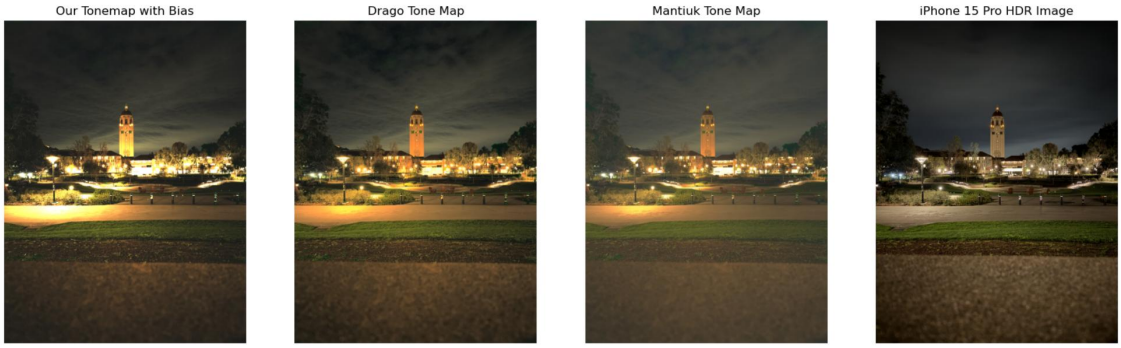

Fig. 9. Comparision of Different HDR Image Pipelines

cooler images. Out of the HDR images we generated, we thought the Mantiuk tonemapped image looked the most accurate to what we observed when we took the photo, but we ended up preferring the Drago tonemapped image. The Drago tonemapped image is much more saturated and vibrant, which we preferred over the more muted tones and colors seen in Mantiuk's tonemap. We did prefer both the Drago, Mantiuk, and our tonemapped images over the iPhone 15 Pro HDR image though as we felt that the iPhone 15 Pro's HDR image was too dark and did not match our perceived vision of what the image should look like.

The results and comparisons of this project have been purely qualitative, since the goal of this project was to be able to create an HDR pipeline for the iPhone that anyone could use to create their own HDR images based on their own personal preferences and style choices.

5 Discussion and Limitations

While our proposed HDR pipeline was successful, there are still some limitations to this pipeline and some things that could be improved. The first limitation is that the iPhone 15 Pro has very limited adjustable camera settings. Even with the use of an additional application, "ProCamera", we were still not able to adjust our exposure time to be above one second. This severely limited how overexposed our original set of images for our camera response function could be, which might have affected our ability to recover the correct camera response function. This was seen when we were trying to recover a camera response function from the set of Hoover Tower images we had. Since we could not get an overexposed enough image, we were not able to recover a full camera response function for all color channels. Some of the color channels did not seem to map log exposure values to the same pixel values we expected from Figure 4.

Another limitation of this project is that the iPhone's default camera application does not allow for adjusting settings such as exposure time, white balance, focal length, or f number. This requires the use of an external application that could give us a different camera response curve than the iPhone's default camera. We also discovered while using an external application, the settings would shift or auto-adjust as we were taking our photos, which rendered some of our datasets useless.

6 Conclusion

Overall, we were successful in obtaining the camera response function of Apple's iPhone 15 Pro and creating our own HDR pipeline to create our own HDR images. We were able to improve our original tonemapping algorithm as well by adding an additional bias term to improving the coloring of our final HDR image.

We believe this project is useful for people who are interested in photography, but may not have the means to buy an expensive or new SLR camera. By following the HDR pipeline proposed in this paper, anyone can create their own HDR images and tune them to their own personal image and color preferences.

While our HDR pipeline did not seem to perform as well as Apple's HDR pipeline in creating accurate HDR images, we do believe that our HDR pipeline allowed us to create HDR images that better suited our personal tastes and image style preferences. We believe that the allowed personalization of HDR images to better match an individual's personal image style and taste is the main benefit of using our HDR pipeline over Apple's default HDR image pipeline.

7 Future Work

Some future work that could done for this project is to see how the camera response function changes as we change our camera settings. All the images used in our HDR pipline had the same white balance value, f number, focal length, and ISO value. We are curious to see how the response curve of the iPhone 15 Pro would differ by changing any of these settings.

We would also like to improve our new HDR tonemapping algorithm. While adding a bias term did seem to improve the image slightly, we would like to test other methods and strategies for our tonemapping algorithm. We could try adding a machine learning (ML) model to better map our linear HDR pixel values to values that better match what we perceive with our eyes. It would be also interesting to train the model on test sets with a specific image style preference such as, warmer toned images, more saturated images, or higher contrast images, to see how that would affect our final mapping. We believe a Convolutional Neural Network (CNN) could be a useful machine learning model

to use, since it is often used and successful in prediction and classification methods and could be useful in our tonemapping algorithm to predict what features humans might like in their images.

ACKNOWLEDGMENTS

The authors would like to thank the EE367 teaching team: Professor Gordon Wetzstein, Axel Levy, and Qingqing Zhao for their help with this project and this course.

REFERENCES

[1] W. Gordon, "How to use hdr mode when taking photos."
[2] J. Takamatsu, Y. Matsushita, and K. Ikeuchi, "Estimating camera response functions using probabilistic intensity similarity," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
[3] J. M. Paul E. Debevec, "Recovering high dynamic range radiance maps from photographs."
[4] T. A. N. C. E. Drago, K. Myszkowski, "Adaptive logarithmic mapping for displaying high contrast scenes."
[5] L. J. K. Rafal Mantiuk, Scott Daly, "Display adaptive tone mapping."