Project Description

High dynamic range (HDR) images have much larger dynamic range than traditional images' 256 brightness levels. In addition, they correspond linearly to physical irradiance values of the scene. Hence, they have many applications in vision. In this project, you are expected to finish the following tasks to assemble an HDR image.

0. Reading.

Read Chapter 10 (HDR section) from Szeliski book and/or the Debevec/Malik HDR paper (see references below).

1. Taking images.

Take a series of photographs for a scene under different exposures. As discussed in the class, changing shutter speed is probably the best way to change exposure for this application. For that, you need a digital camera that allows you to set exposures. (Note that not every camera allows a user to manually set exposures.) You can use your own camera or borrow a Canon PowerShot A640 (Links to an external site.) or Canon PowerShot SX110 (Links to an external site.) from us. If you need to borrow one camera, please check out the schedule on Canvas and send Jia Xu an email. We strongly recommend that you use a tripod when you take pictures. Again, you can use your own or borrow one from Jia. We will provide flash memory. However, you will need your own batteries (4x AA Batteries, Alkaline Ni-MH) to operate the camera.

One thing to note is that you should avoid moving your camera during this process so that all pictures are well registered. At the same time, you should try, if possible, to ensure that there is minimal change in the scene (people walking in front of the camera causing occlusions). Some digital cameras have their own programs which allow users to remotely control the shutters via their USB cables. Using such programs prevent you from shaking the camera while pressing the shutter. For some Canon camera models, there are third party programs, for example, AHDRIA, which allow you to set exposures and release shutter remotely from a PC or a laptop. Cameras of other brands might have a similar solution for remote capturing, but you have to discover on your own. You are welcome to write down your findings for that matter in your report.

If you decide to manually change exposures, it would help if you align your images before proceeding to the next step. Photomatix (Links to an external site.) has a free version for HDR creation and image alignment, Photomatix Pro 2.5.2 (Links to an external site.). RASCAL (Links to an external site.) also has a utility called imageAlignment for aligning images. You can download from their website. You may also manually align images using a program like GIMP (Links to an external site.). You are also welcome to write a program for alignment as bonus points.

2. Write a program to assemble an HDR image.

Write a program to take these captured images as inputs and output an HDR image as well as the response curve of the camera. You will use Debevec's method. Please refer to his Debevec's SIGGRAPH 1997 (Links to an external site.) paper. The details of this algorithm has been discussed in class. Additional details and pointers to other papers are in the Szeliski book (drafts are available online). The central piece of the implementation will solve the over-determined linear system. You can use a linear algebra package:

- LAPACK (Links to an external site.) (recommended, with C++ (Links to an external site.) and Java (Links to an external site.) wrappers)
- GSL (Links to an external site.)
- Boost (Links to an external site.)
- ATLAS (Links to an external site.)
- Pardiso (Links to an external site.)

In Matlab you may use the backslash operator (Links to an external site.). In fact, Matlab has already implemented (Links to an external site.) the Debevec's HDR algorithm. You can work in matlab to simplify image I/O issues, but you need to use Matlab's basic functions to reimplement this algorithm. I trust you that you will not just blindly copy the example code on lecture slides. You can refer to Chapter 2 of Numerical Recipes (Links to an external site.) if you want to write your own linear solver.
Matlab is not the only option. You can try OpenCV or Python too. For the linear system solver, we encourage you to experiment. For example, you can make use of a linear system solver in CUDA.

If you want to test your program before taking pictures of your own, you can download test images from

- Prof. Li Zhang (Links to an external site.)
- Debevec (Links to an external site.)
- RASCAL (Links to an external site.)

You may also use existing libraries to perform I/O from image files. We recommend that you output your radiance map as a Radiance RGBE image (.hdr). C code to write RGBE files is provided by Greg Ward Larson (Links to an external site.). Reading the input PNG or JPG images can be done with any libraries, including

- OpenCV (Links to an external site.) (recommended) (note: OpenCV has a bit of a learning curve)
- CImg (Links to an external site.)

Matlab provides the imread (Links to an external site.) function.
If you do not want to use Matlab, we provide a **windows-only** C++ image class called gil which supports I/O for many image formats for both unsigned-char images and float-point images including RGBE.
Click here (Links to an external site.) to download the gil library and source code showing you how to use it (this version should work on CSL Windows machines). To compile and build the code in MS Visual Studio, you may need to change the "Include" and "Library" directory settings for the gil library.
3. Develop your radiance map using tone mapping.
Load your radiance map into HDRShop (Links to an external site.) (windows-only) [Linux links TBA] and use Reinhard's tone mapping algorithm to develop your radiance map into a usual image. Reinhard's algorithm has been implemented as a free HDRShop plugin (Links to an external site.). You may also use PFStmo (Links to an external site.) (linux-only), which implements several tone-mapping algorithms including Reinhard's.
Some other tone-mapping software:

- Photomatix (Links to an external site.)

- logview (Links to an external site.)
- Fast Bilateral Filtering (Links to an external site.)

You must setup a team. Find your partner on Canvas. (see below.)

For your experiments, pick out scenes where the perceptual difference between a "regular" image and your HDR version is significant. This /will/ count towards your grade.

Bonus?

You must implement at least one (preferably two) of the following extensions: **implementing one or more functionality from this section is essential for full points**.

- Image alignment methods for HDR imaging. A good suggestion is Ward's MTB algorithm (Links to an external site.).
- Other HDR creation methods such as Mitsunaga and Nayar's algorithm. Please refer to their CVPR 1999 paper, Radiometric Self Calibration (Links to an external site.). The main difference between this method compared with Debevec's method is that this method assumes that the response curves are high-order polynomials. RASCAL (Links to an external site.) uses this method. Another popular method is Robertson's method (Links to an external site.). It is more robust.
- Your own tone mapping implementation. Any tone mapping algorithms other than Reinhard's are counted as bonuses. More tone mapping algorithms are listed in the reference at the bottom of this page.
- Read Chapter 10 in the Szeliski book, and implement one or more extensions listed in the HDR section.
- Run a small experiment with HDR video, see this paper (Links to an external site.) or find others from Google scholar
- We will link to your results online (see below) and other students from this class will vote for the best results on Canvas. The top three winners will get extra credits of up to 10 additional points

You are strongly encouraged to try any other extension you can think of related to high dynamic range images. The bonus depends on how useful and difficult these extensions are. We will be generous with partial credit even if the functionality you were attempting does not work extremely well. Please be creative.

Groups

Two students must form a group to do this project. If you are really low on time or have some other extenuating circumstances, you can have a group of three people (with prior approval). If you *really* want to practice on everything, you can do it individually.

Quantitative measurement of progress and contribution

On the day you start work on this project, setup a version control system (e.g., git, svn). We expect to see *exactly* how much code commits each member in the team is responsible for. If you are unable to submit a working project in time, your code commits may help with partial

credit. If your submission is merely "satisfactory" and the progress was driven entirely by one team member, credit will NOT be equally distributed among all team members. If your submission is among the best in class, each team member will receive full credit.

Submission

Package your code and report in a zip file, and upload from this Canvas. Please also include one good "representative" image of your work which should be considered for voting.

Your submission must also include

- You full codebase.
- A clearly written documentation in PDF or html or wiki form (for an example, see this (Links to an external site.))
- A representative image for voting. Please name it as to_be_vote.png. This should be from the scenes you photographed.
- Credits: How each member of your group contributed

Grading

Submitting a bare minimum version will map to **~70 out of 100 points**. The best submission will map to ~110. The grade for all other submissions will be interpolated between these two extremes.

Reference software

- Photomatix (Links to an external site.): a high-quality commercial program to create HDR images and tone mapping. It has a free version with basic operations.
- HDRShop (Links to an external site.): a program to create, view and manipulate HDR images.
- AHDRIA & AHDRIC (Links to an external site.): a program to automatically create HDR images including camera support.
- RASCAL (Links to an external site.): a HDR creation program with image alignment capability.

References (more coming soon)

HDR

- Paul E. Debevec, Jitendra Malik, Recovering High Dynamic Range Radiance Maps from Photographs (Links to an external site.), SIGGRAPH 1997.
- Tomoo Mitsunaga, Shree Nayar, Radiometric Self Calibration (Links to an external site.), CVPR 1999.

Alignment

- Greg Ward, Fast Robust Image Registration for Compositing High Dynamic Range Photographs from Hand-Held Exposures (Links to an external site.), jgt, 2003.

Tone Mapping

- Jack Tumblin, Greg Turk, LCIS: A Boundary Hierarchy for Detail-Preserving Contrast Reduction (Links to an external site.), SIGGRAPH 1999.
- Sumanta Pattanaik, Jack Tumblin, Hector Yee, Donald Greenberg, Time-Dependent Visual Adaptation for Fast Realistic Image Display (Links to an external site.), SIGGRAPH 2000.
- Raanan Fattal, Dani Lischinski, Michael Werman, Gradient Domain High Dynamic Range Compression (Links to an external site.), SIGGRAPH 2002.
- Fredo Durand, Julie Dorsey, Fast Bilateral Filtering for the Display of High Dynamic Range Images (Links to an external site.), SIGGRAPH 2002.
- Erik Reinhard, Michael Stark, Peter Shirley, Jim Ferwerda, Photographics Tone Reproduction for Digital Images (Links to an external site.), SIGGRAPH 2002.
- Michael Ashikhmin, A Tone Mapping Algorithm for High Contrast Images (Links to an external site.), EGWR 2002.
- F. Drago, K. Myszkowski, T. Annen, N. Chiba, Adaptive Logarithmic Mapping for Displaying High Contrast Scenes (Links to an external site.), Eurographics 2003.
- Erik Reinhard, Kate Devlin, Dynamic Range Reduction Inspired by Photoreceptor Physiology (Links to an external site.), IEEE TVCG 2005.
- Hwann-Tzong Chen, Tyng-Luh Liu, Tien-Lung Chang, Tone Reproduction: A Perspective from Luminance-Driven Perceptual Grouping (Links to an external site.), CVPR 2005.
- Grzegorz Krawczyk, Karol Myszkowski, Hans-Peter Seidel, Lightness Perception in Tone Reproduction for High Dynamic Range Images (Links to an external site.), Eurographics 2005.