Implementation of High Dynamic Range (HDR) Photography

Huayu Zhang

University of Wisconsin-Madison

Introduction

- Dynamic Range: ratio of the largest illuminance to smallest illuminance.
- Task: Capture a series of images with different exposures, and merge them into an HDR image.

Figure: Source: http://farbspiel-photo.com/learn/hdr-pics-to-play-with

Process

- Preprocessing – Image alignment [2] (omitted)
- Radiance estimation
- Tone mapping

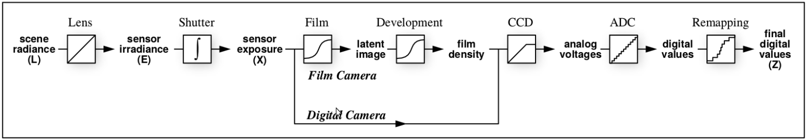


Figure: Image formation [1]

Nonlinear relationship between scene radiance and pixel values.

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

$i \in \{1, \ldots, n\}$: index of a pixel, $j \in 1, \ldots, m$: index of an image, Z_{ij}: pixel value, E_i: actual illumination of pixel i, Δt_j: exposure time of image j, $f : \mathbb{R}^+ \mapsto \{0, 1, \ldots, 255\}$ a nonlinear function. Rewrite Eq. 1,

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

$g : \{0, 1, \ldots, 255\} \mapsto \mathbb{R}^+$, $E_i : i \in \{1, \ldots, n\}$ unknown

$$\underset{g,\boldsymbol{E}}{\arg\min} \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(w(Z_{ij})\left(g(Z_{ij}) - \ln E_i - \ln \Delta t_j\right)\right)^2}_{\text{radiance fidelity}} + \lambda \underbrace{\sum_{z=1}^{254}\left(w(z)\left(g(z-1) - 2g(z) + g(z+1)\right)\right)^2}_{\text{smoothness constraint}}$$

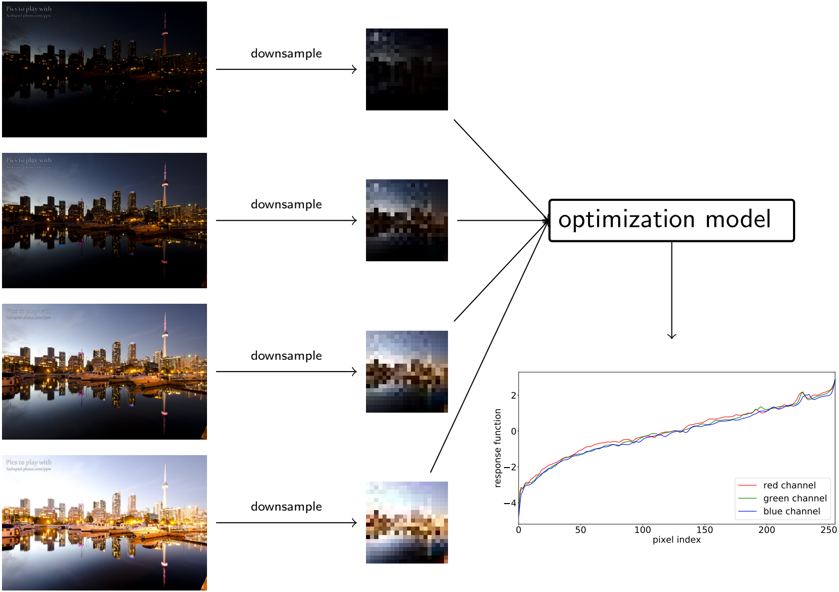
$$\text{s.t.} \quad g(128) = 0$$

- λ: trade-off between the radiance fidelity and the smoothness.
- $w(z) = 1 - \frac{|z-128|}{130}$ a weight function.
- Let the first derivative of the object function w.r.t. $\ln E_i$ and $g(z)$ equal to 0.

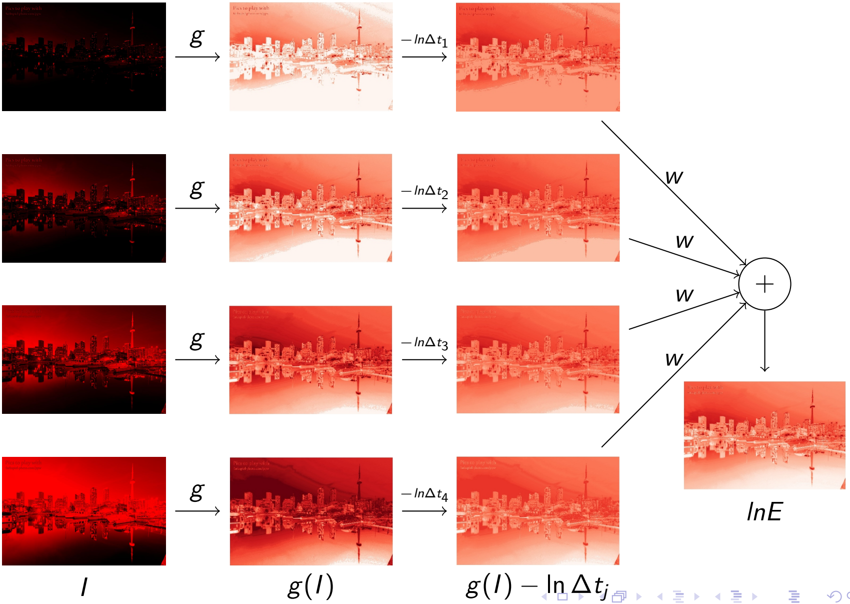
$$\begin{aligned} w(Z_{ij})g(Z_{ij}) - w(Z_{ij})\ln E_i = w(Z_{ij})\ln \Delta t_j &\quad \forall i \in \{1,2,\ldots,n\}, j \in \{1,2,\ldots,m\} \\ \lambda w(z) - 2\lambda w(z)g(z+1) + \lambda w(z)g(z+2) = 0 &\quad \forall z \in \{1,2,\ldots,254\} \end{aligned}$$

Radiance estimation

estimate response function g

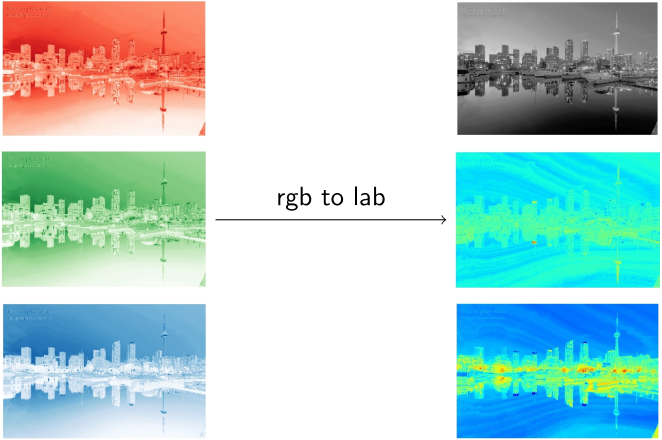


Radiance estimation
Red channel



- Radiance range [0.12656, 667.55] ([-2.0670, 6.5036] log scale).
- Dynamic range 5274.4.
- Compress the dynamic range to 255. $\rightarrow$ Tone mapping.

Tone mapping
RGB to LAB



RGB color space, Scaled to [0, 1]   LAB color space, channel 1 scaled to [0,1]

- The first channel of LAB color space is the image lightness while the other two channels contain color information.
- Only process the L channel.

Contrast Limited Adaptive Histogram Equalization [3]

- Redistribute the image range based on histogram.
- Higher frequency $\rightarrow$ wider range.

Figure: L channel

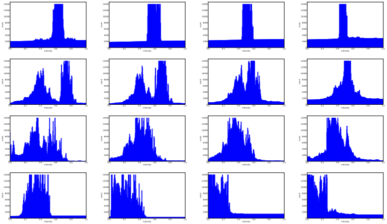


Figure: Histogram (Clipped)

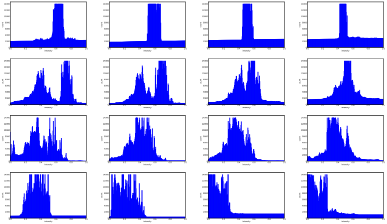


Figure: Histogram

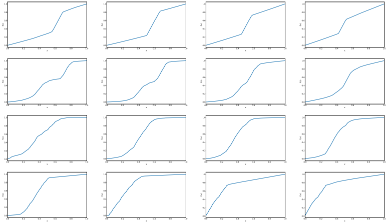
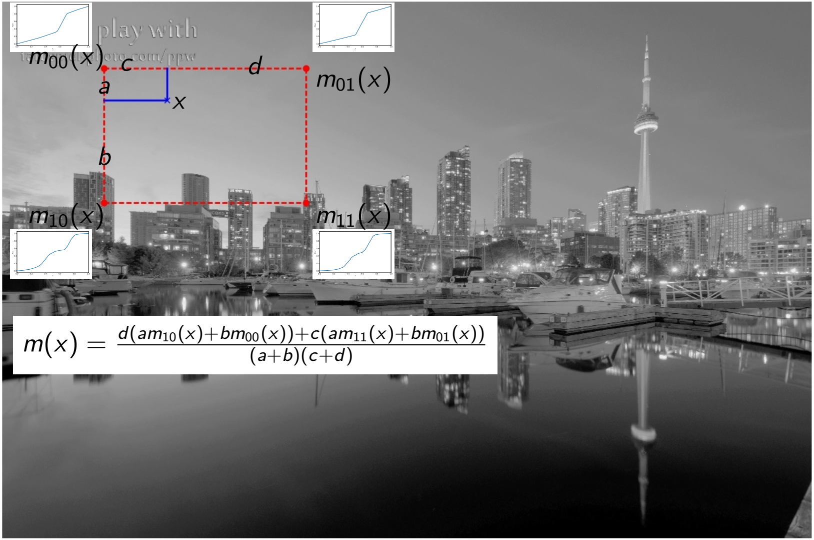


Figure: Tile mappings

mapping $\longrightarrow$

Tone mapping

Eliminating discontinuity – Bilinear interpolation



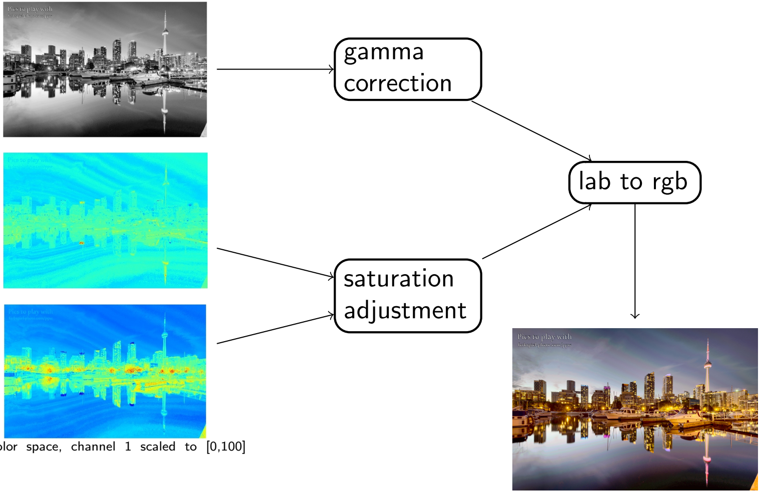
$$m(x) = \frac{d(am_{10}(x)+bm_{00}(x))+c(am_{11}(x)+bm_{01}(x))}{(a+b)(c+d)}$$

Tone mapping

Effect of CLAHE

Figure: L channel (before CLAHE)

Figure: L channel (after CLAHE)



LAB color space, channel 1 scaled to [0,100]

Example (I)

Figure: Source: http://farbspiel-photo.com/learn/hdr-pics-to-play-with

Figure: HDR

Example (II)

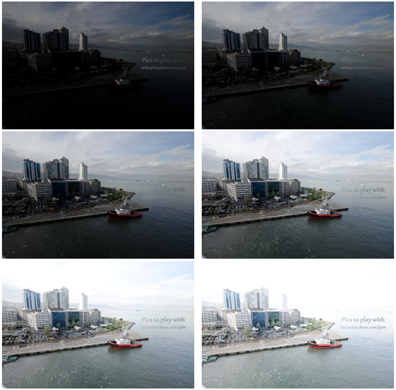


Figure: Source: http://farbspiel-photo.com/learn/hdr-pics-to-play-with

Figure: HDR

HDR on Android phones

- Build a complete CV system.
  (1) Image capture
  (2) HDR reconstruction
- Not trivial
  (1) Complex Android API: camera2
  (2) Limited resource: JVM 64MB memory for each app
  (3) Low quality of images compared with DSLR.
  (4) Difficult to fine tune parameters: no ground truth.
- Not very successful. A prototype can hardly be used in reality.
  (1) slow (30 seconds for 1920×1080)
  (2) not robust to vibration.
  (3) surrealistic.

HDR on Android phones
Camera2 API

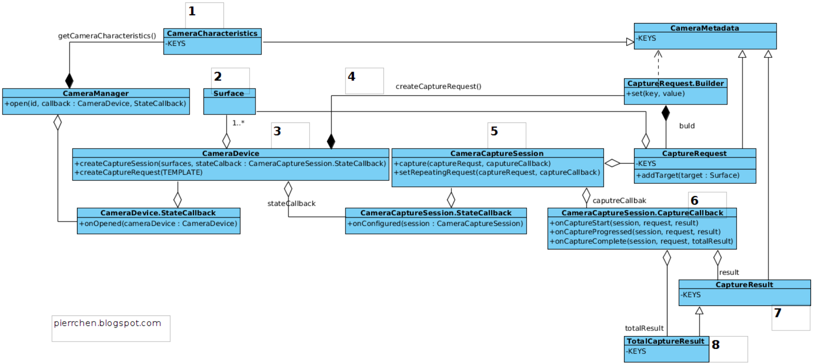


Figure: Image credit:`http://pierrchen.blogspot.com/2015/01/android-camera2-api-explained.html`

A complex framework,

- How to control focus, exposure, aperture, ISO, white balance?
- How to consecutively capture a set of bracketing exposure images?

- Use the existing camera app : Opencamera.
- Hacking Opencamera
  - (1) Opencamera takes care of auto focus, auto exposure, bracketing capture, fixing aperture, ISO and WB.
  - (2) Opencamera already has an HDR function in HDRProcessor.java `private void processHDRCore(List<Bitmap>`
  - (3) Replace it with my algorithm.
  - (4) Record the exposure time on image capturing.
- Implementing my algorithm with Java? No enough memory (64MB limit) → endless garbage collection when memory usage approaches the 64MB limit.
- Final solution: Native C++.

Figure: Opencamera interface

Figure: HDR

HDR on Android phones
CS Dept (2)

Figure: HDR

HDR on Android phones
CS Dept (3)



Figure: HDR

HDR on Android phones

Compared with Google HDR+, Ultimate HDR, Opencamera

(a) Normal image (reference)

(b) My implementation

(c) Google HDR+

(d) opencamera

- Much worse than DSLR+PC post-processing
  - ⋆ Limited exposure time ($\leq 0.25s$)
  - ⋆ Small aperture
  - ⋆ Limited number of burst images (3)
  - ⋆ Vibration
  - ⋆ Limited computation resources.
- Possible improvements
  - ⋆ Image alignment/registration/deblur
  - ⋆ Fine tune parameters: smoothness weight, saturation, gamma correction, histogram upper bound
  - ⋆ High performance frameworks (GPU/multithreading), e.g. RenderScript

Main References

Paul E Debevec and Jitendra Malik. "Recovering high dynamic range radiance maps from photographs". In: *ACM SIGGRAPH 2008 classes*. ACM. 2008, p. 31.

B Srinivasa Reddy and Biswanath N Chatterji. "An FFT-based technique for translation, rotation, and scale-invariant image registration". In: *IEEE transactions on image processing* 5.8 (1996), pp. 1266–1271.

Karel Zuiderveld. "Graphics Gems IV". In: ed. by Paul S. Heckbert. San Diego, CA, USA: Academic Press Professional, Inc., 1994. Chap. Contrast Limited Adaptive Histogram Equalization, pp. 474–485. ISBN: 0-12-336155-9. URL: http://dl.acm.org/citation.cfm?id=180895.180940.