High dynamic range imaging

Digital Visual Effects

Yung-Yu Chuang

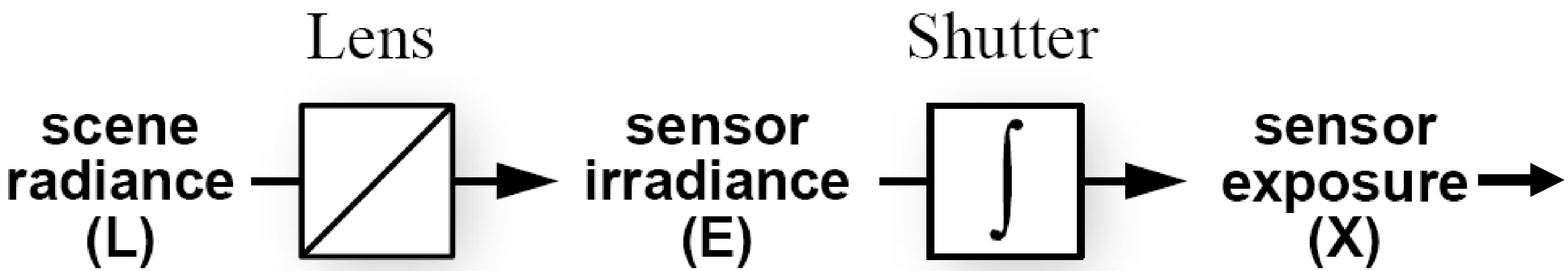
with slides by Fredo Durand, Brian Curless, Steve Seitz, Paul Debevec and Alexei Efros

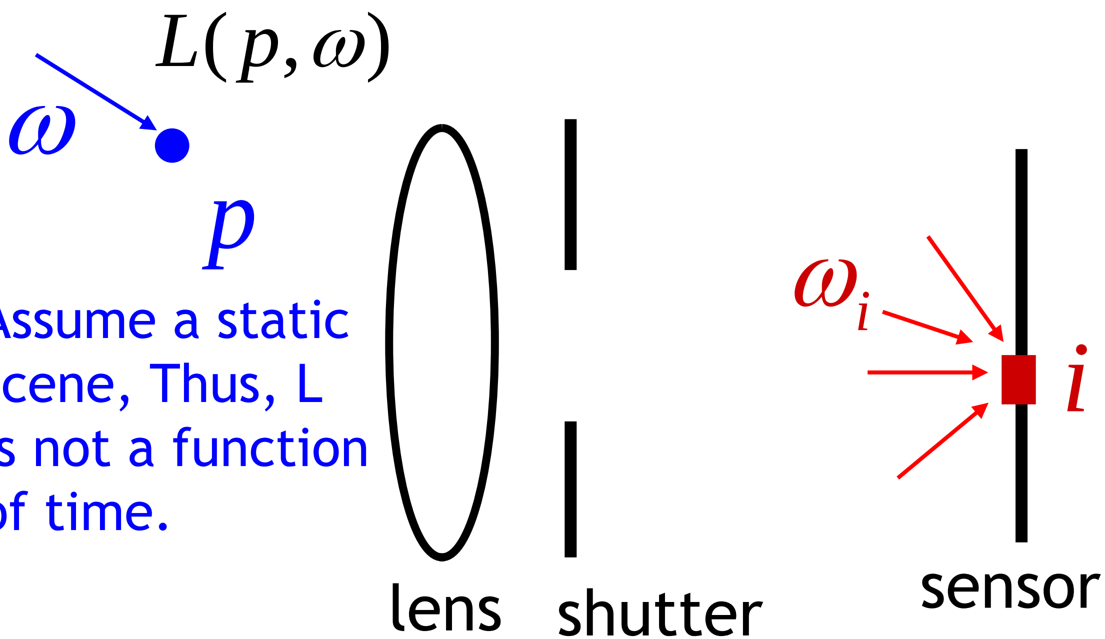
Camera is an imperfect device

DigiVFX

- Camera is an imperfect device for measuring the radiance distribution of a scene because it cannot capture the full spectral content and dynamic range.
- Limitations in sensor design prevent cameras from capturing all information passed by lens.

Camera pipeline



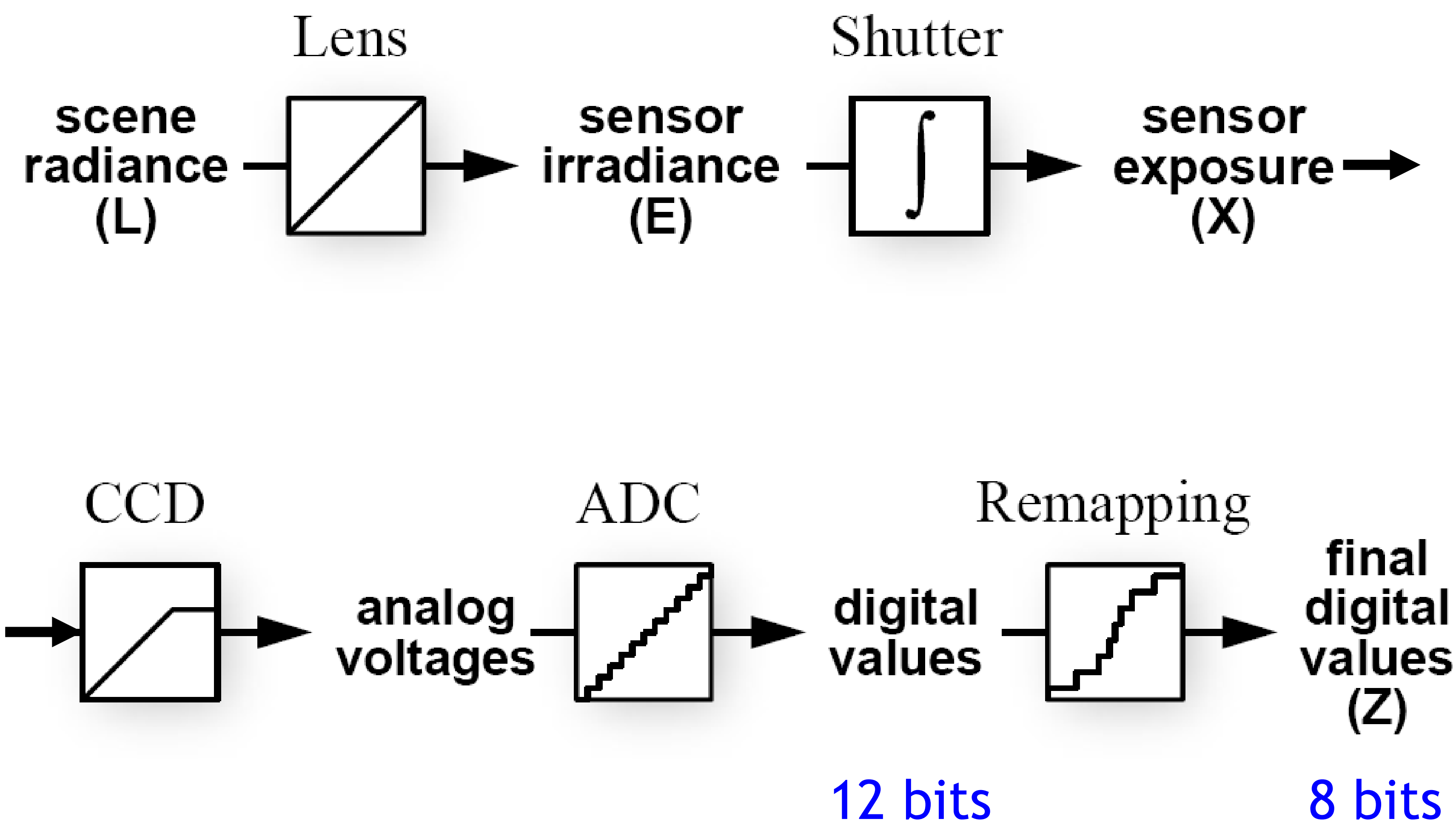


Assume a static scene, Thus, L is not a function of time.

$$E_i = \int_\Omega L(i,\omega_i)\,d\omega$$

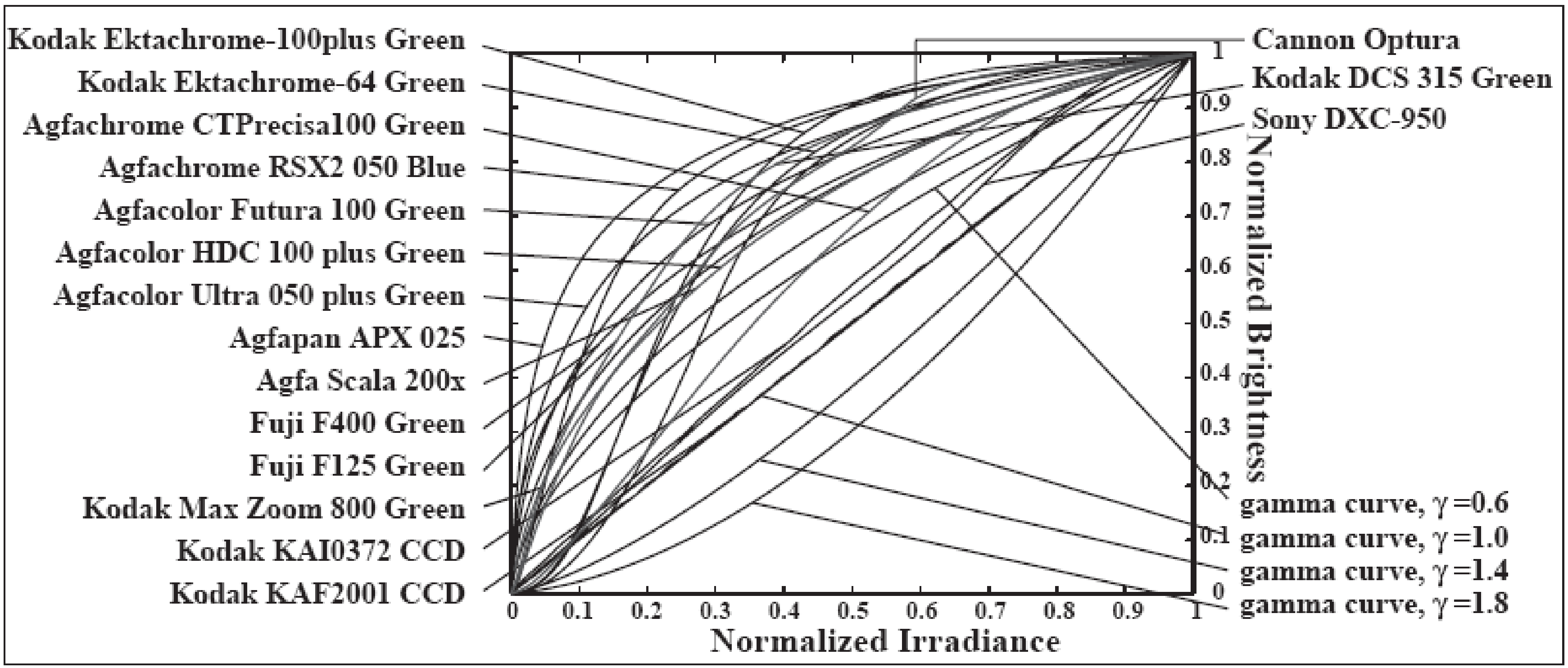
$$X_i = \int_{t=0}^{\Delta t} E_i\,dt = E_i \Delta t$$

Camera pipeline



Real-world response functions

In general, the response function is not provided by camera makers who consider it part of their proprietary product differentiation. In addition, they are beyond the standard gamma curves.



The world is high dynamic range

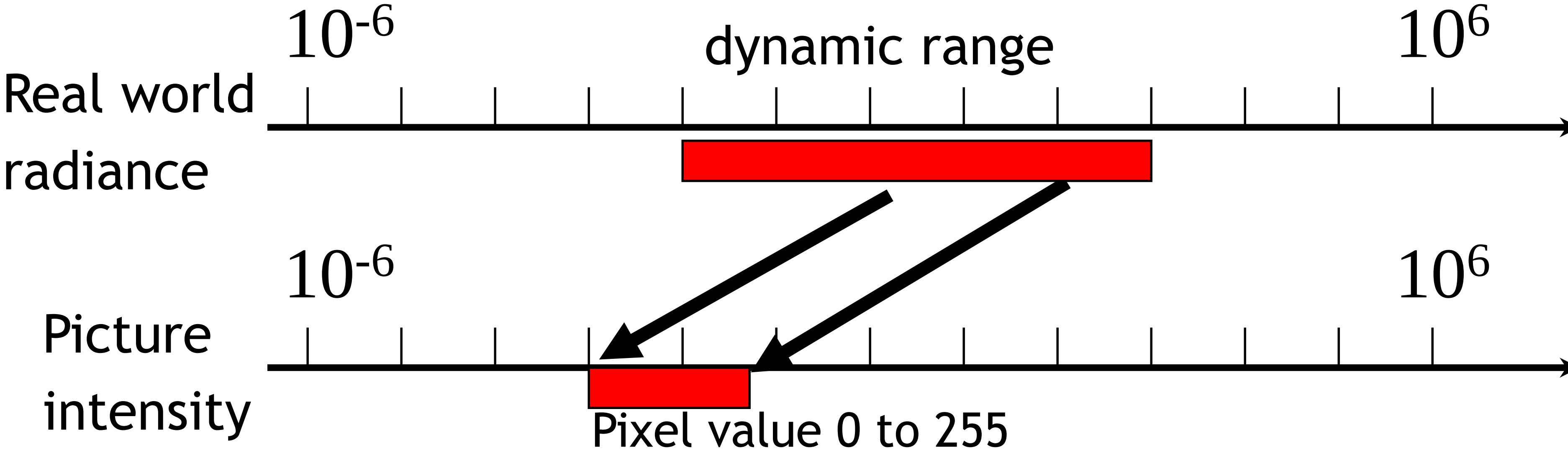
The world is high dynamic range

Real world dynamic range

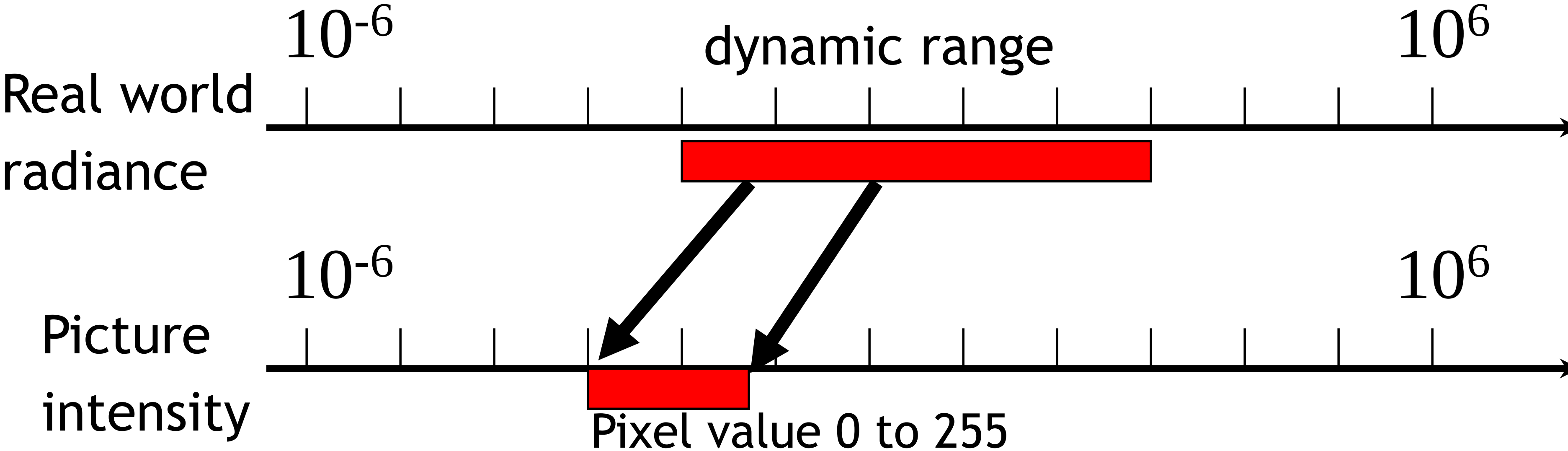
- Eye can adapt from ~ 10^{-6} to 10^{6} cd/m^2
- Often 1 : 100,000 in a scene
- Typical 1:50, max 1:500 for pictures

spotmeter

Short exposure



Long exposure

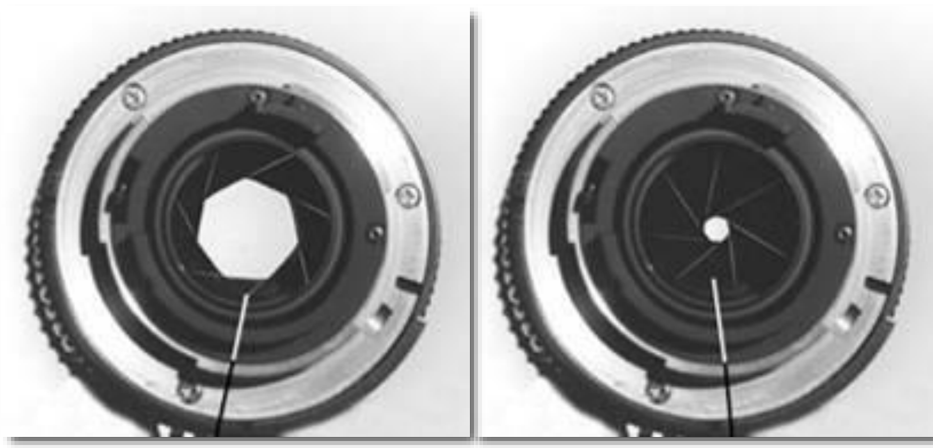


Camera is not a photometer

- Limited dynamic range
  $\Rightarrow$ Perhaps use multiple exposures?
- Unknown, nonlinear response
  $\Rightarrow$ Not possible to convert pixel values to radiance
- Solution:
  - Recover response curve from multiple exposures, then reconstruct the ***radiance map***

Varying exposure

- Ways to change exposure
  - Shutter speed
  - Aperture
  - Neutral density filters



Shutter speed

- Note: shutter times usually obey a power series – each “stop” is a factor of 2

- ¼, 1/8, 1/15, 1/30, 1/60, 1/125, 1/250, 1/500, 1/1000 sec

  Usually really is:

  ¼, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512, 1/1024 sec

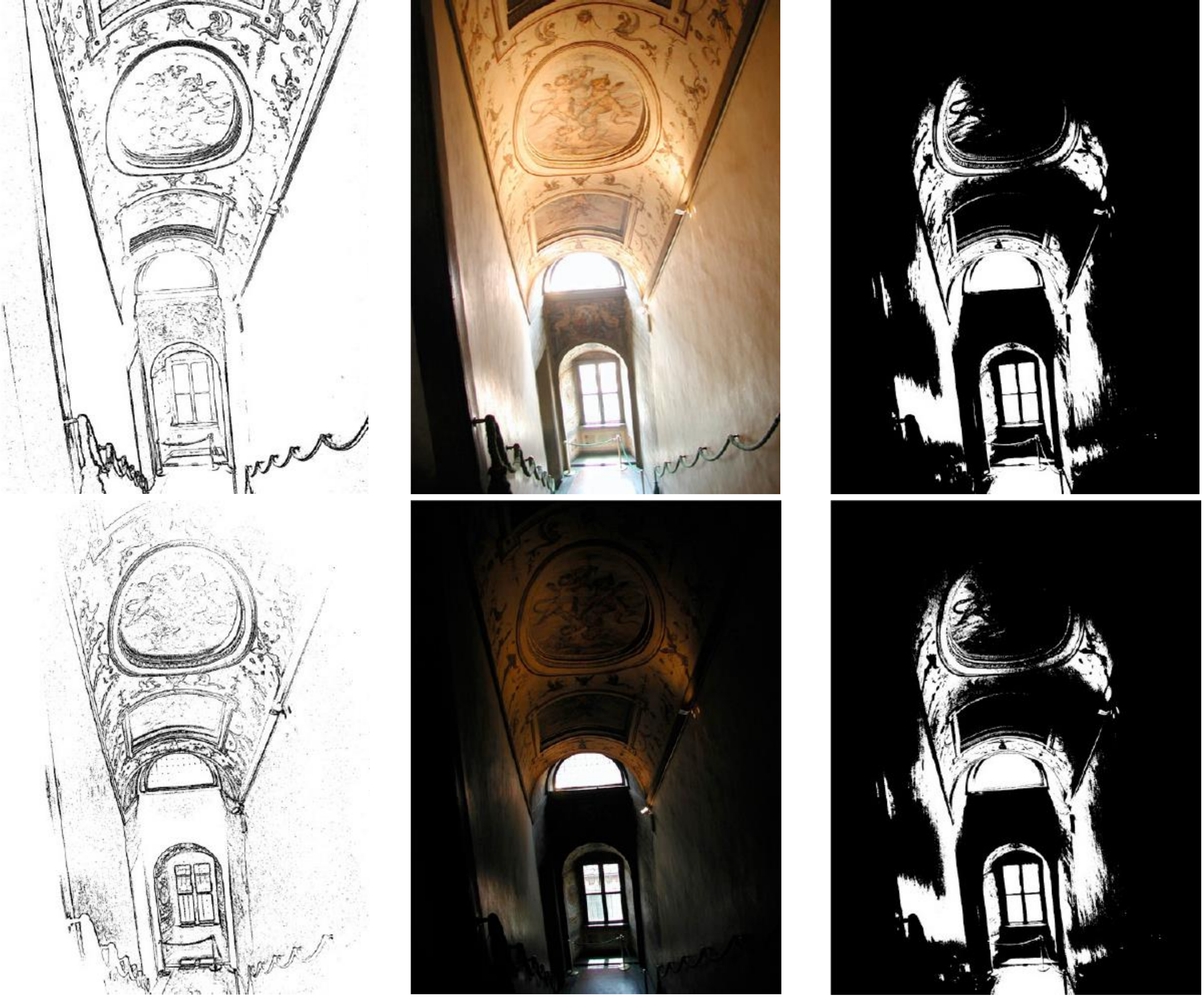
Varying shutter speeds

HDRI capturing from multiple exposures

- Capture images with multiple exposures
- Image alignment (even if you use tripod, it is suggested to run alignment)
- Response curve recovery
- Ghost/flare removal

Image alignment

- We will introduce a fast and easy-to-implement method for this task, called Median Threshold Bitmap (MTB) alignment technique.
- Consider only integral translations. It is enough empirically.
- The inputs are N grayscale images. (You can either use the green channel or convert into grayscale by Y=(54R+183G+19B)/256)
- MTB is a binary image formed by thresholding the input image using the median of intensities.



Why is MTB better than gradient?

- Edge-detection filters are dependent on image exposures
- Taking the difference of two edge bitmaps would not give a good indication of where the edges are misaligned.

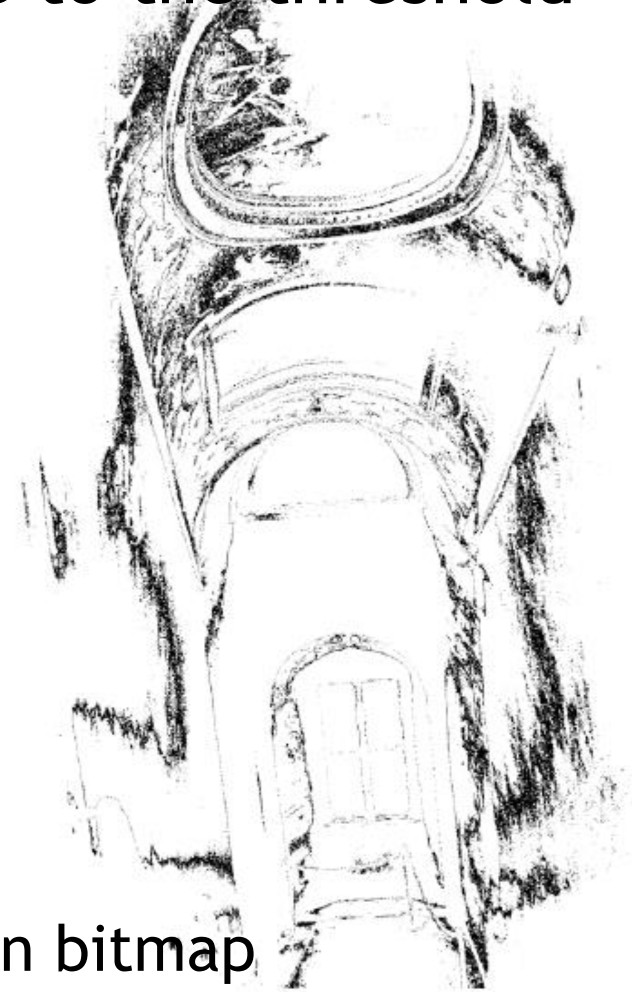
Search for the optimal offset

- Try all possible offsets.
- Gradient descent
- Multiscale technique

- log(max_offset) levels
- Try 9 possibilities for the top level
- Scale by 2 when passing down; try its 9 neighbors

Threshold noise

ignore pixels that are close to the threshold



exclusion bitmap

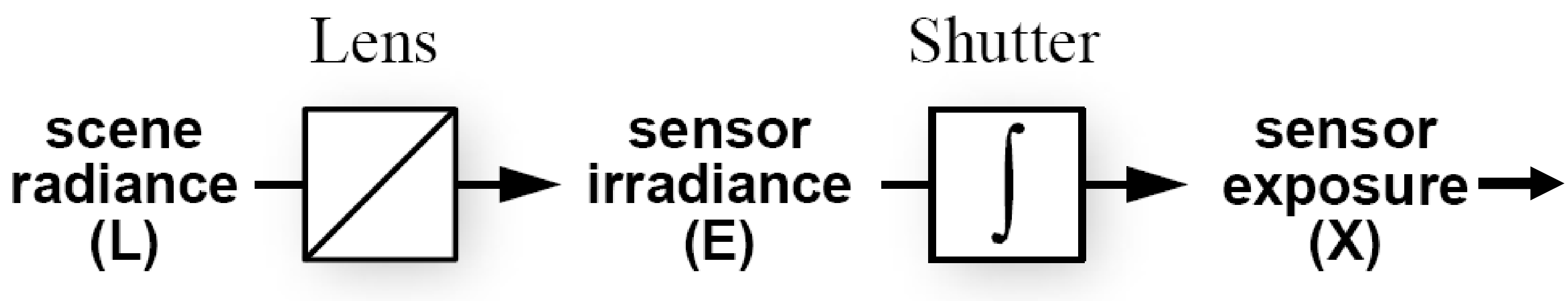
Efficiency considerations

- XOR for taking difference
- AND with exclusion maps
- Bit counting by table lookup

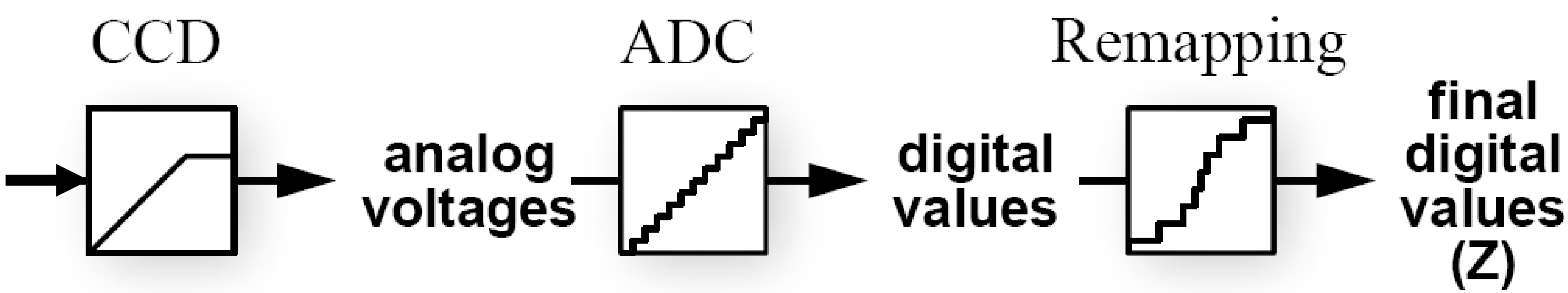
Results

Success rate = 84%. 10% failure due to rotation. 3% for excessive motion and 3% for too much high-frequency content.

Recovering response curve



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



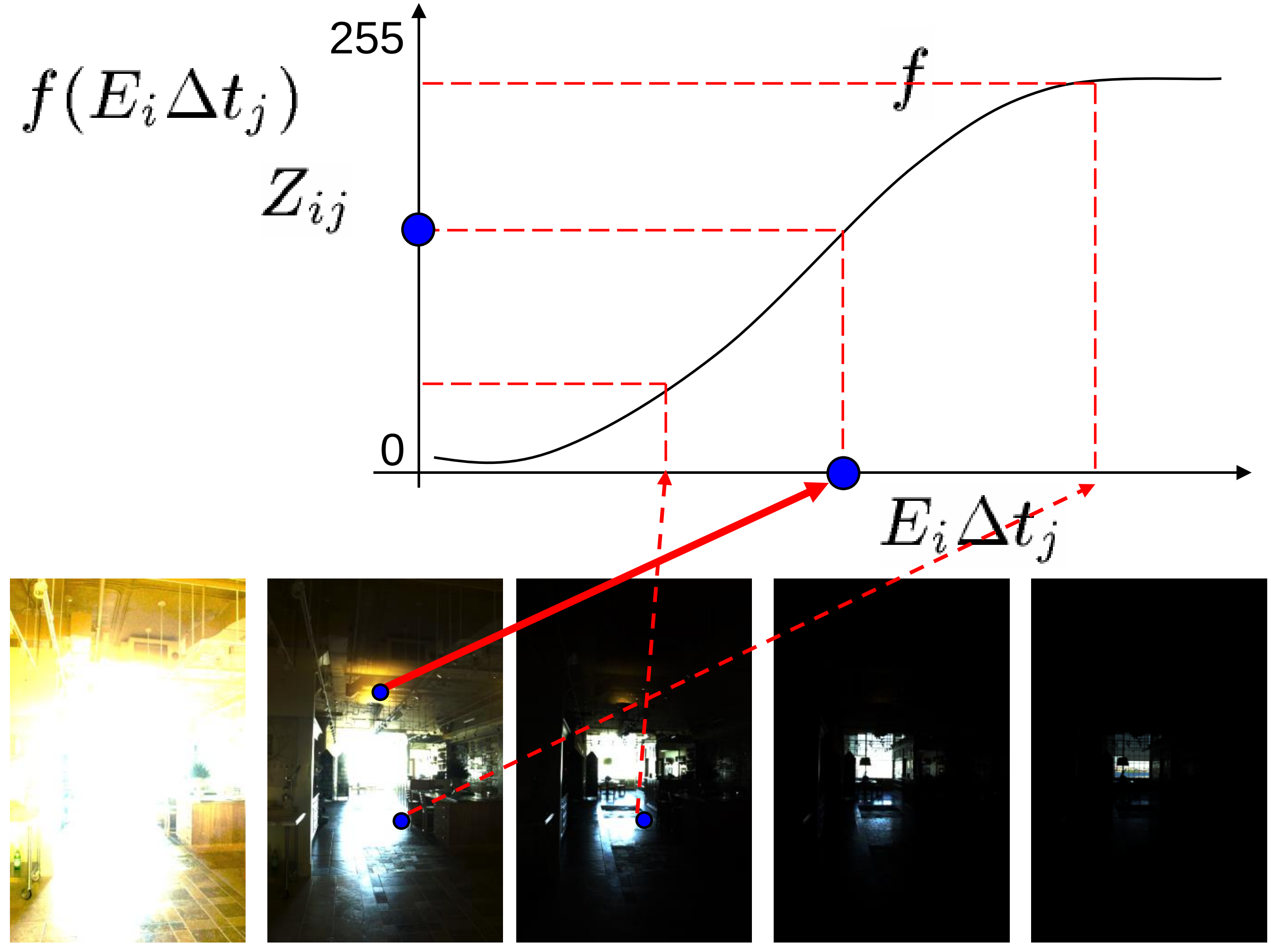
12 bits

8 bits

Recovering response curve

- We want to obtain the inverse of the response curve

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

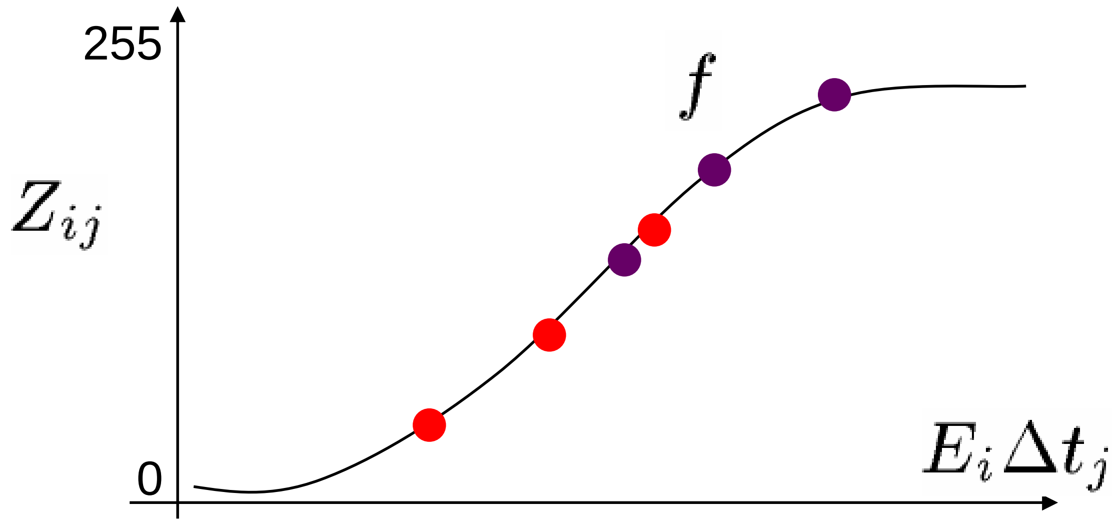


Recovering response curve

Image series

Δt = 2 sec | Δt = 1 sec | Δt = 1/2 sec | Δt = 1/4 sec | Δt = 1/8 sec

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



Recovering response curve

Image series

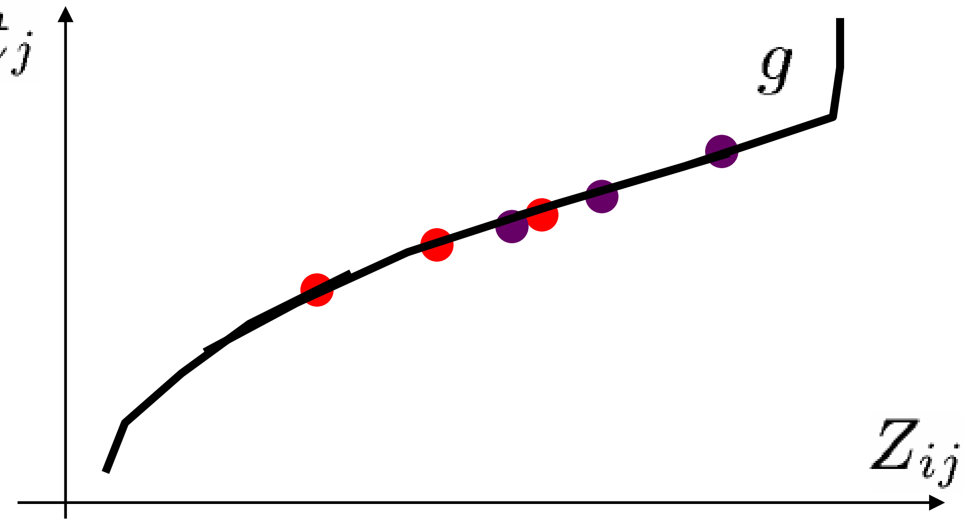
Δt = 2 sec	Δt = 1 sec	Δt = 1/2 sec	Δt = 1/4 sec	Δt = 1/8 sec

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

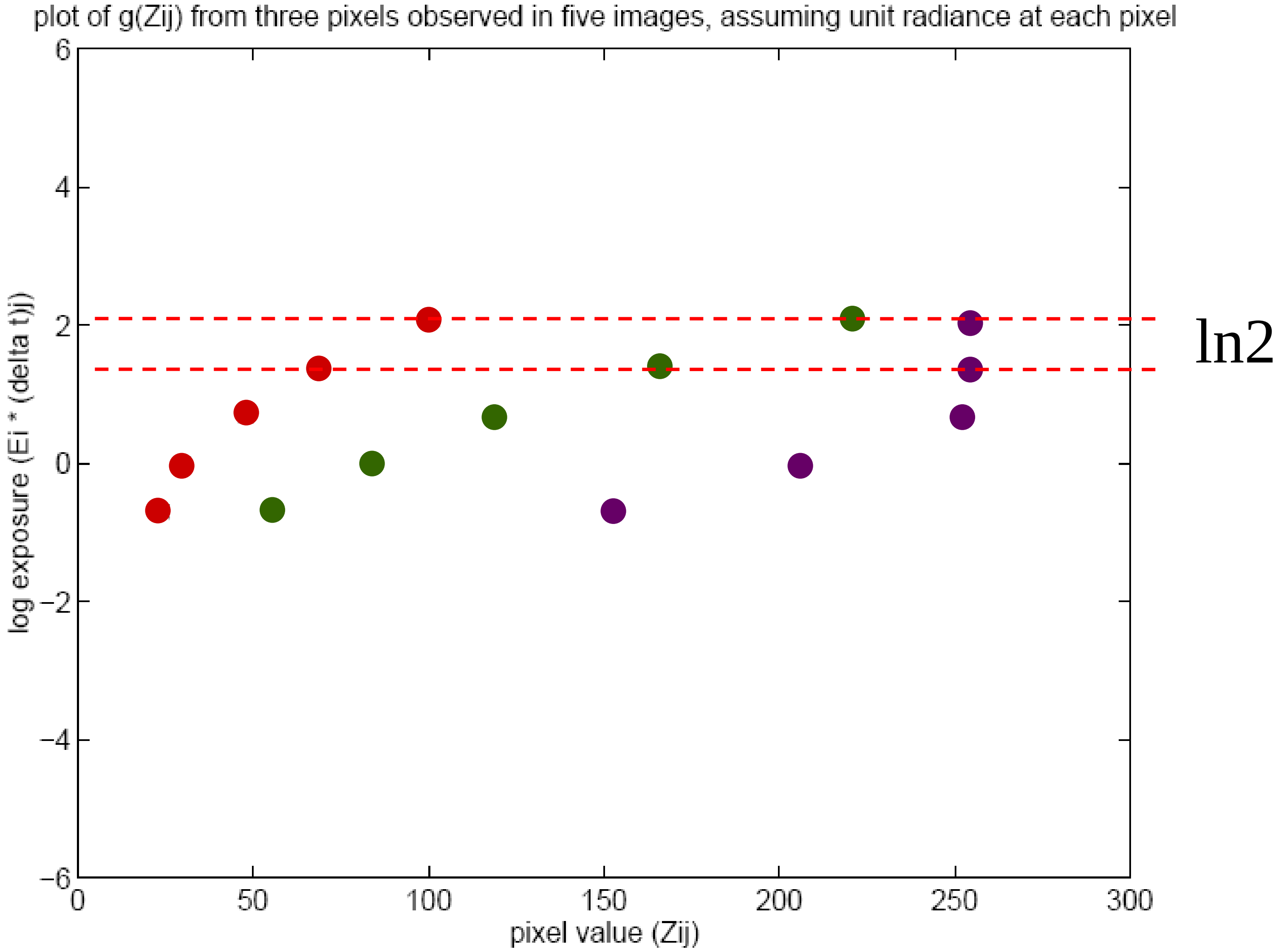
$$X_{ij} = E_i \Delta t_j$$

$$\ln X_{ij} = \ln E_i + \ln \Delta t_j$$

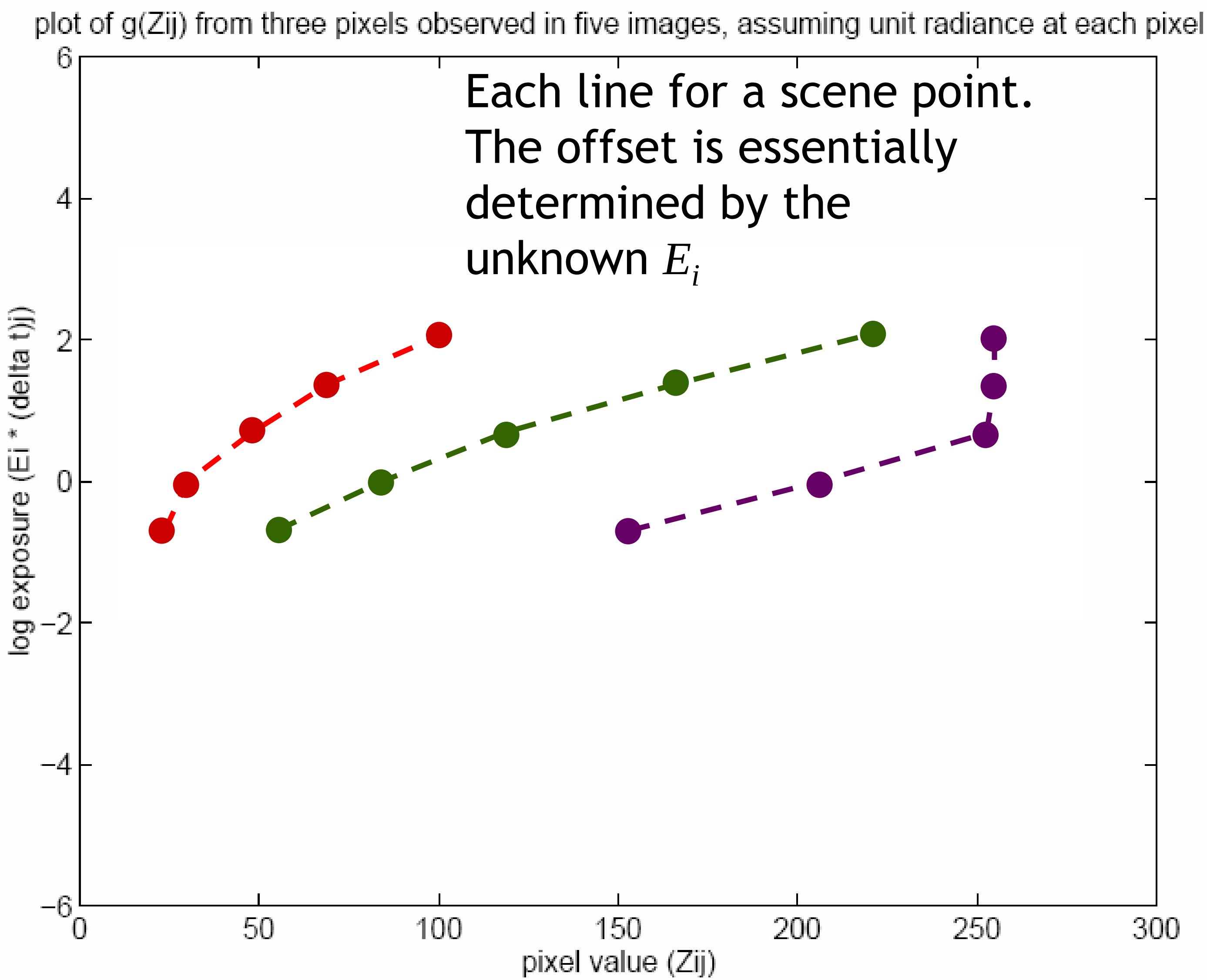
$$g(Z_{ij}) = \ln E_i + \ln \Delta t_j$$



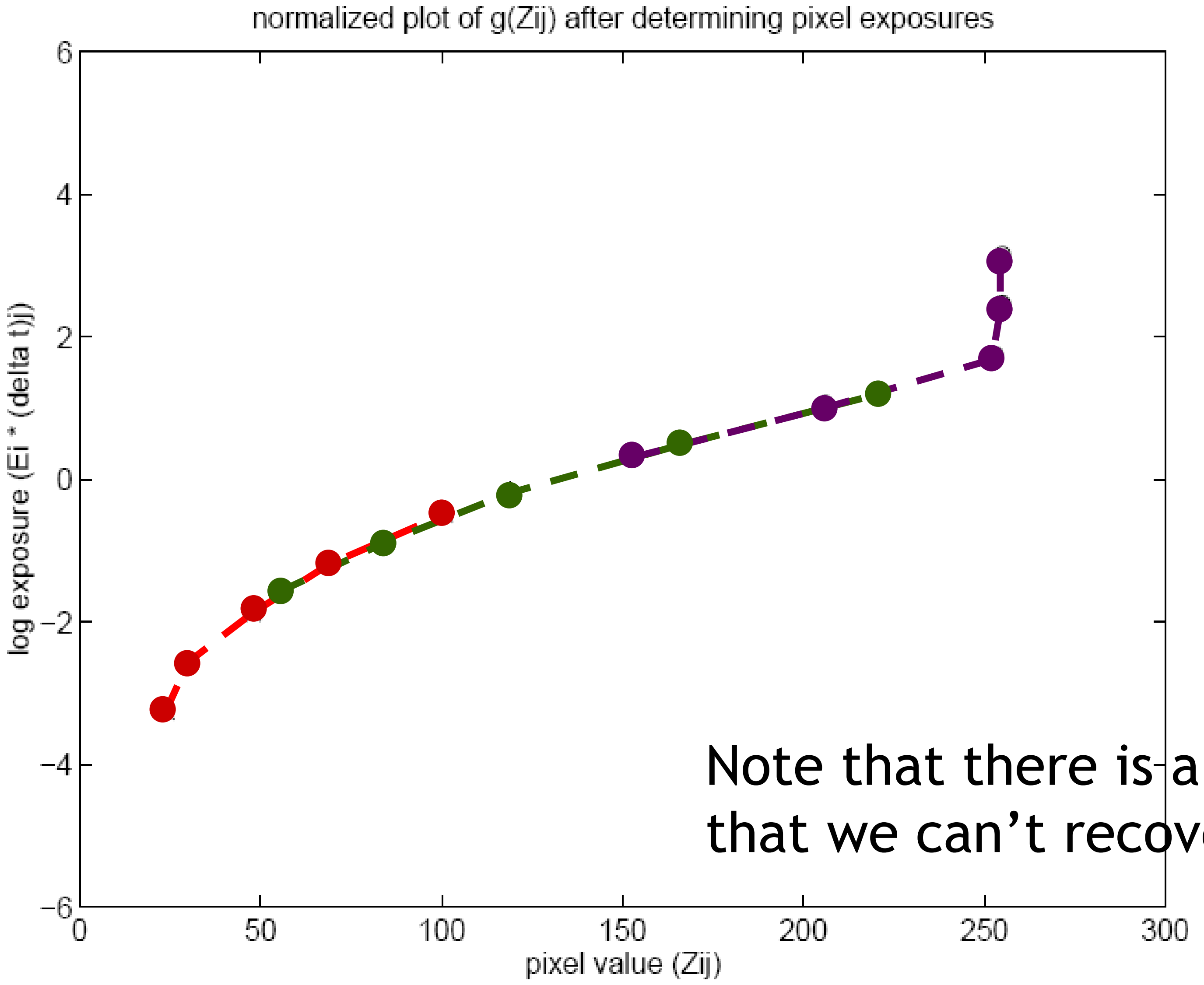
Idea behind the math



Idea behind the math



Idea behind the math



Note that there is a shift that we can't recover

Basic idea

- Design an objective function
- Optimize it

Math for recovering response curve

DigiVFX

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

f is monotonic, it is invertible

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

let us define function $g = \ln f^{-1}$

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

minimize the following

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

$$g''(z) = g(z-1) - 2g(z) + g(z+1)$$

Recovering response curve

- The solution can be only up to a scale, add a constraint

$$g(Z_{mid}) = 0, \text{ where } Z_{mid} = \frac{1}{2}(Z_{min} + Z_{max})$$

- Add a hat weighting function

$$w(z) = \begin{cases} z - Z_{min} & \text{for } z \le \frac{1}{2}(Z_{min} + Z_{max}) \\ Z_{max} - z & \text{for } z > \frac{1}{2}(Z_{min} + Z_{max}) \end{cases}$$

$$\mathcal{O} = \sum_{i=1}^{N} \sum_{j=1}^{P} \{w(Z_{ij}) [g(Z_{ij}) - \ln E_i - \ln \Delta t_j]\}^2 +$$

$$\lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} [w(z) g''(z)]^2$$

Recovering response curve

- We want $N(P-1) > (Z_{max} - Z_{min})$

  If P=11, N~25 (typically 50 is used)
- We prefer that selected pixels are well distributed and sampled from constant regions. They picked points by hand.
- It is an overdetermined system of linear equations and can be solved using SVD

How to optimize?

$$\mathcal{O} = \sum_{i=1}^{N} \sum_{j=1}^{P} \left\{ w(Z_{ij}) \left[g(Z_{ij}) - \ln E_i - \ln \Delta t_j \right] \right\}^2 +$$

$$\lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} \left[w(z) g''(z) \right]^2$$

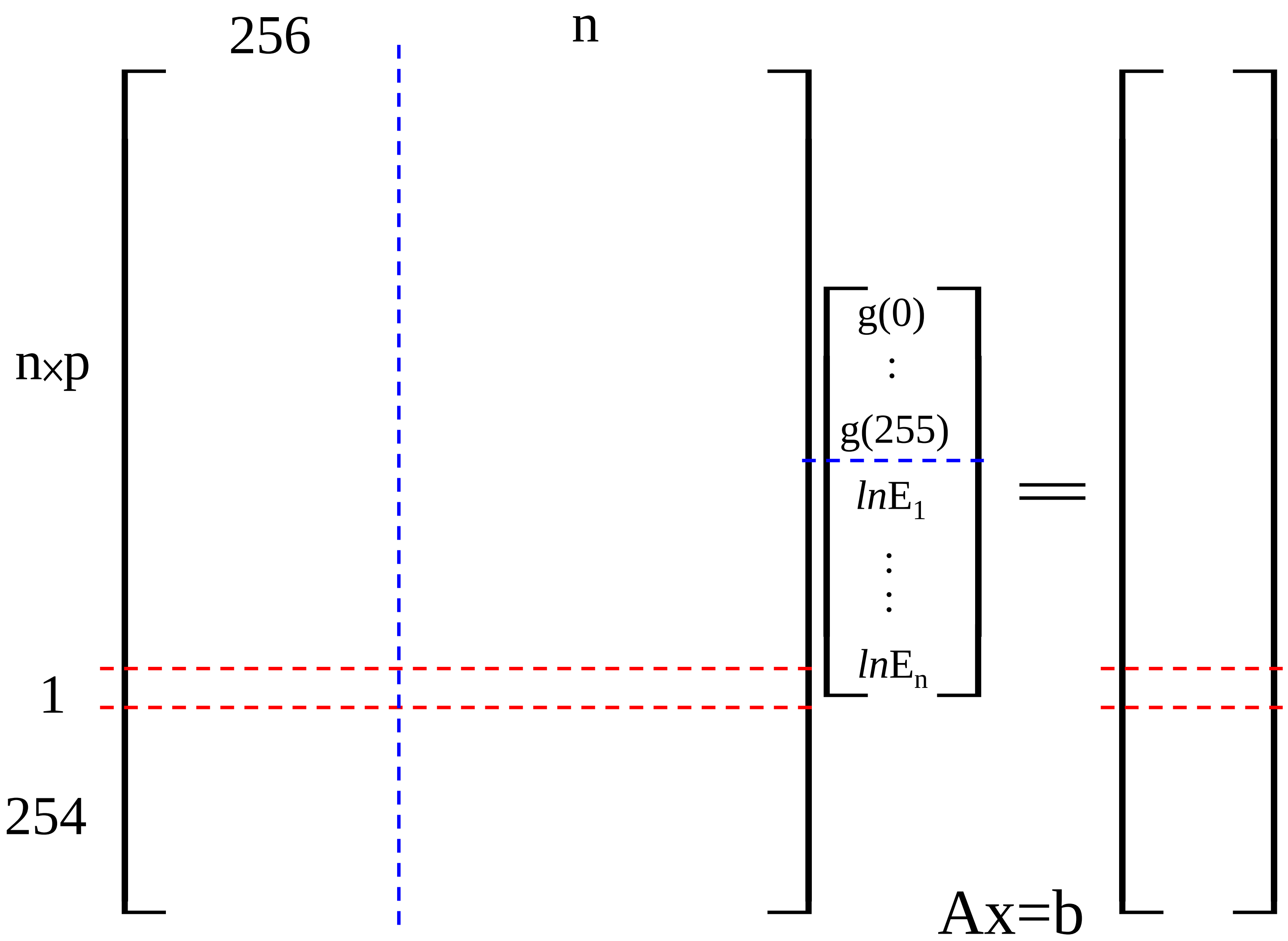
How to optimize?

$$\mathcal{O} = \sum_{i=1}^{N} \sum_{j=1}^{P} \{w(Z_{ij}) [g(Z_{ij}) - \ln E_i - \ln \Delta t_j]\}^2 +$$

$$\lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} [w(z) g''(z)]^2$$

1. Set partial derivatives to zero
2. $$\min \sum_{i=1}^{N} (\mathbf{a_i x} - \mathbf{b_i})^2 \rightarrow \text{least-square solution of } \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \vdots \\ \mathbf{a_N} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \vdots \\ \mathbf{b_N} \end{bmatrix}$$

Sparse linear system



Questions

- Will g(127)=0 always be satisfied? Why or why not?
- How to find the least-square solution for an over-determined system?

Least-square solution for a linear system

DigiVFX

$$\underset{m\times n}{\mathbf{A}}\ \underset{n}{\mathbf{x}} = \underset{m}{\mathbf{b}}$$

$$m > n$$

They are often mutually incompatible. We instead find $\mathbf{x}$ to minimize the norm $\|\mathbf{Ax}-\mathbf{b}\|$ of the residual vector $\mathbf{Ax}-\mathbf{b}$. If there are multiple solutions, we prefer the one with the minimal length $\|\mathbf{x}\|$.

Least-square solution for a linear system

If we perform SVD on $\mathbf{A}$ and rewrite it as

$$\mathbf{A} = \mathbf{U\Sigma}V^{\mathbf{T}}$$

then $\hat{\mathbf{x}} = \mathbf{V\Sigma^{+}U^{T}b}$ is the least-square solution.

pseudo inverse

$$\mathbf{\Sigma}^{+} = \begin{bmatrix} 1/\sigma_1 & & & & & & 0 & \cdots & 0 \\ & \ddots & & & & & & & \\ & & 1/\sigma_r & & & & \vdots & & \vdots \\ & & & 0 & & & & & \\ & & & & \ddots & & & & \\ & & & & & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Proof

Proof

Libraries for SVD

- Matlab
- GSL
- Boost
- LAPACK
- ATLAS

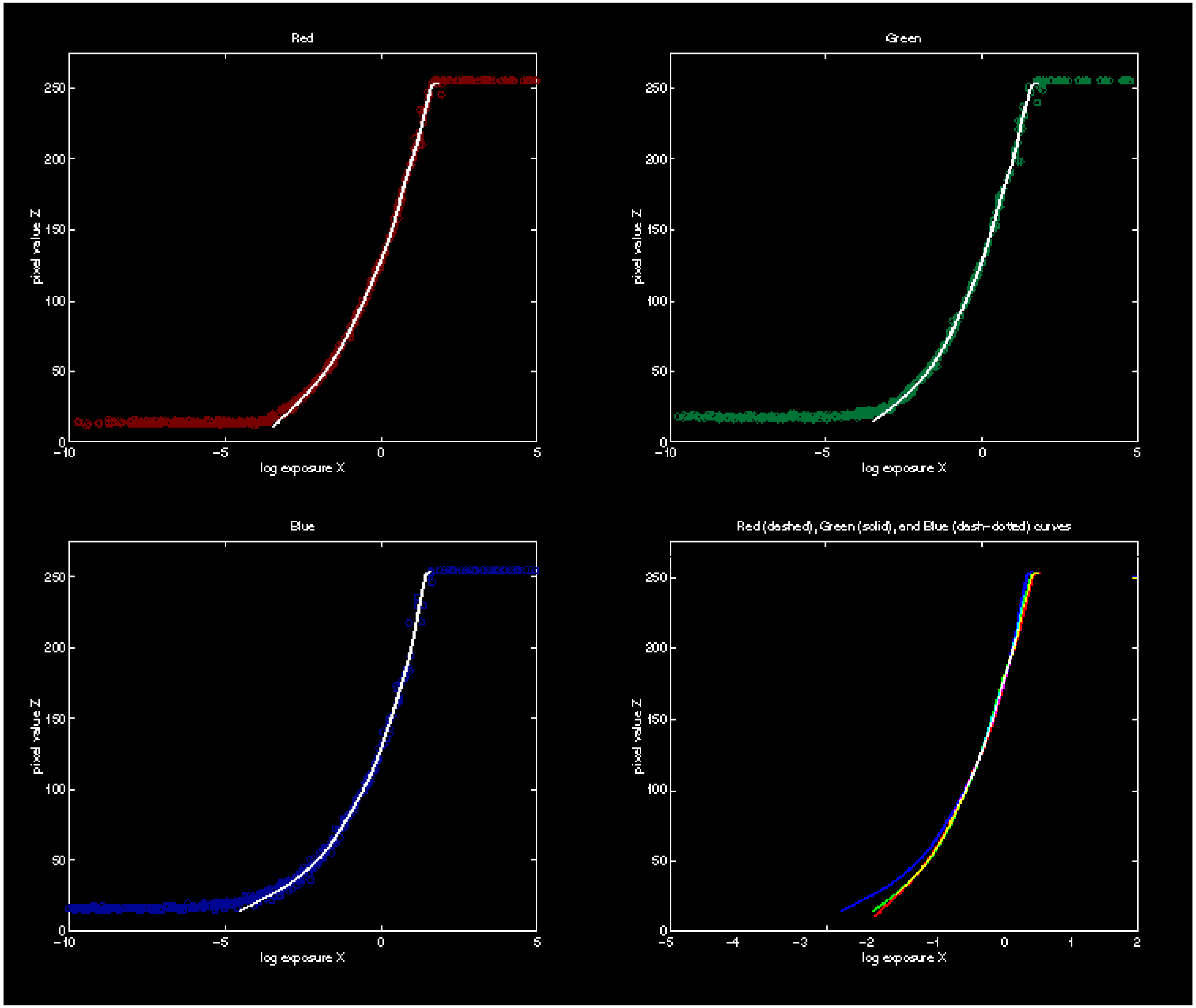
Matlab code

```
%
% gsolve.m - Solve for imaging system response function
%
% Given a set of pixel values observed for several pixels in several
% images with different exposure times, this function returns the
% imaging system's response function g as well as the log film irradiance
% values for the observed pixels.
%
% Assumes:
%
%  Zmin = 0
%  Zmax = 255
%
% Arguments:
%
%  Z(i,j) is the pixel values of pixel location number i in image j
%  B(j)   is the log delta t, or log shutter speed, for image j
%  l      is lamdba, the constant that determines the amount of smoothness
%  w(z)   is the weighting function value for pixel value z
%
% Returns:
%
%  g(z)   is the log exposure corresponding to pixel value z
%  lE(i)  is the log film irradiance at pixel location i
%
```

Matlab code

```
function [g,lE]=gsolve(Z,B,l,w)

n = 256;
A = zeros(size(Z,1)*size(Z,2)+n+1,n+size(Z,1));
b = zeros(size(A,1),1);

k = 1;                  %% Include the data-fitting equations
for i=1:size(Z,1)
  for j=1:size(Z,2)
    wij = w(Z(i,j)+1);
    A(k,Z(i,j)+1) = wij; A(k,n+i) = -wij; b(k,1) = wij * B(   j);
    k=k+1;
  end
end

A(k,129) = 1;           %% Fix the curve by setting its middle value to 0
k=k+1;

for i=1:n-2             %% Include the smoothness equations
  A(k,i)=l*w(i+1); A(k,i+1)=-2*l*w(i+1); A(k,i+2)=l*w(i+1);
  k=k+1;
end

x = A\b;                %% Solve the system using SVD

g = x(1:n);
lE = x(n+1:size(x,1));
```

Recovered response function



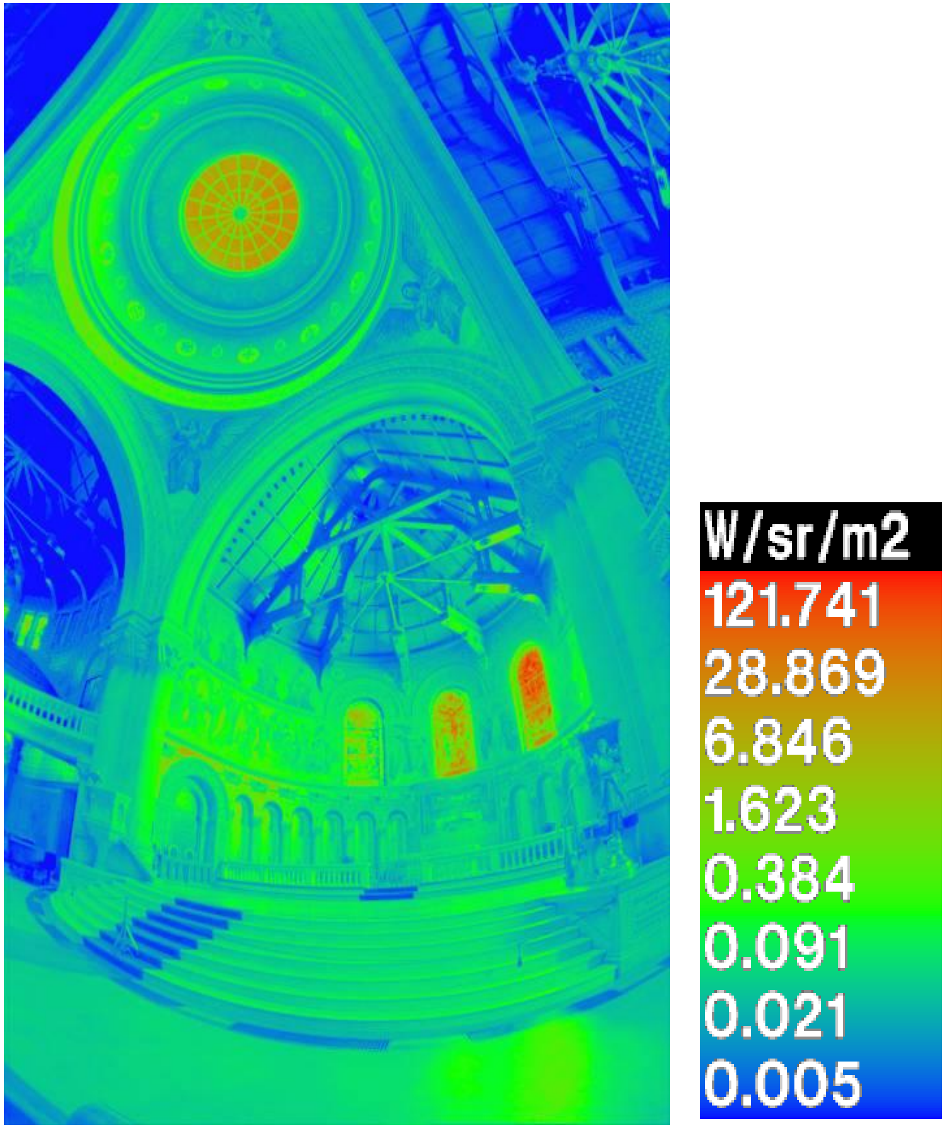
Constructing HDR radiance map

$$\ln E_i = g(Z_{ij}) - \ln \Delta t_j$$

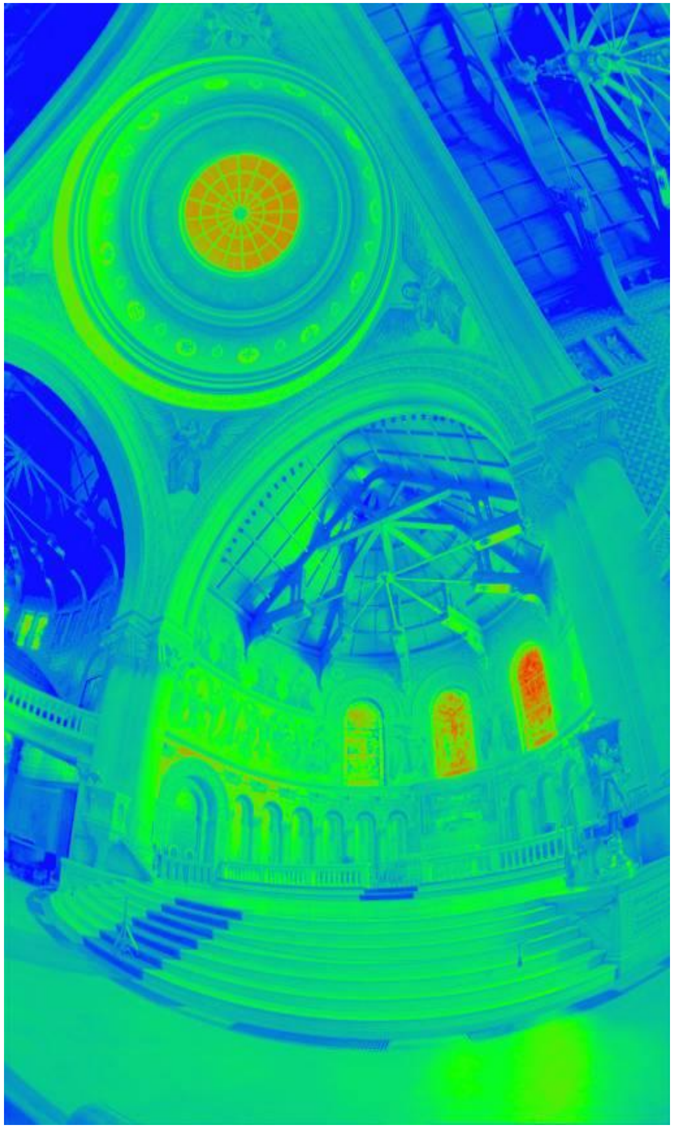
combine pixels to reduce noise and obtain a more reliable estimation

$$\ln E_i = \frac{\sum_{j=1}^{P} w(Z_{ij})(g(Z_{ij}) - \ln \Delta t_j)}{\sum_{j=1}^{P} w(Z_{ij})}$$

Reconstructed radiance map



What is this for?



- Human perception
- Vision/graphics applications

Automatic ghost removal

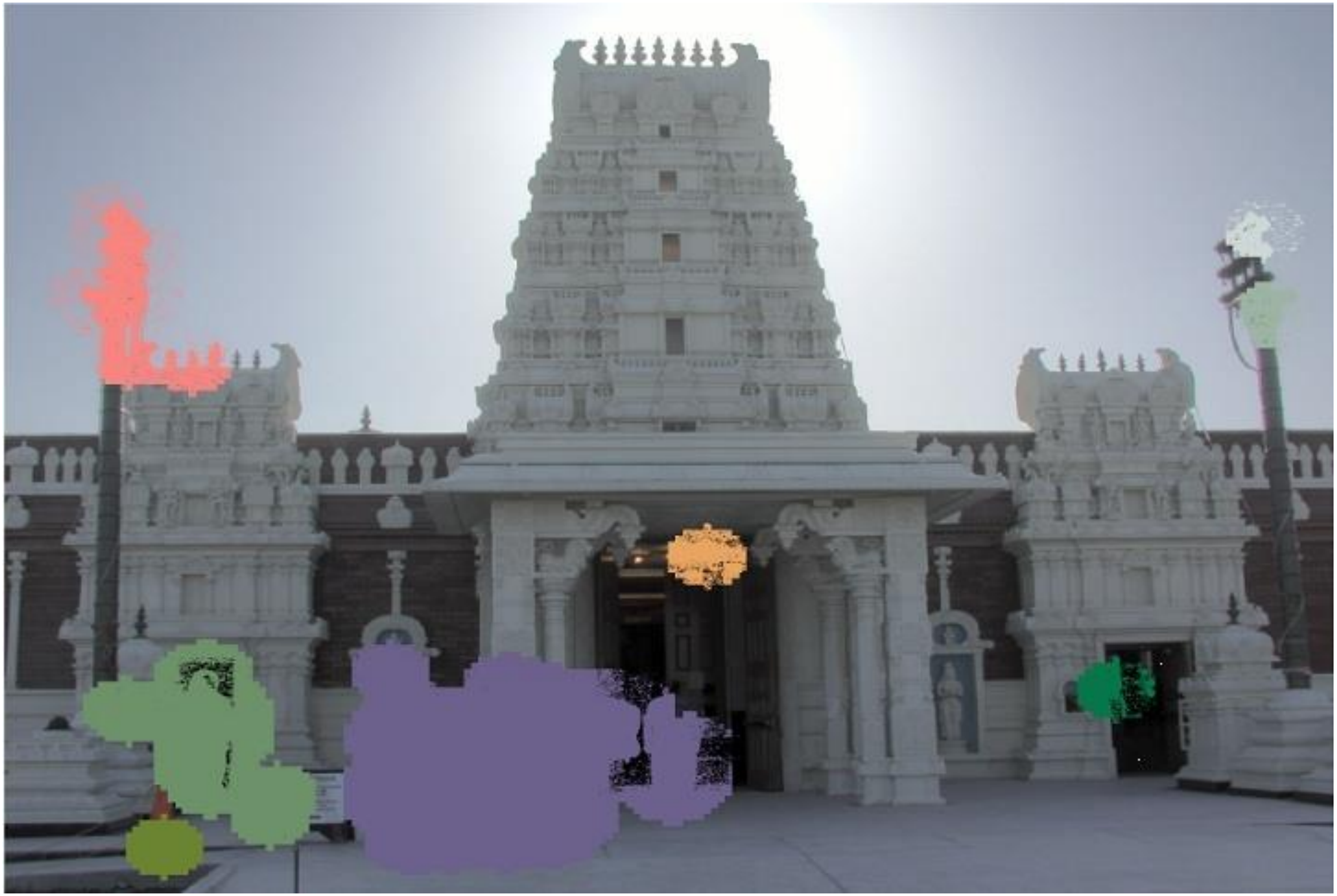
before

after

Weighted variance

Moving objects and high-contrast edges render high variance.

Region masking



Thresholding; dilation; identify regions;

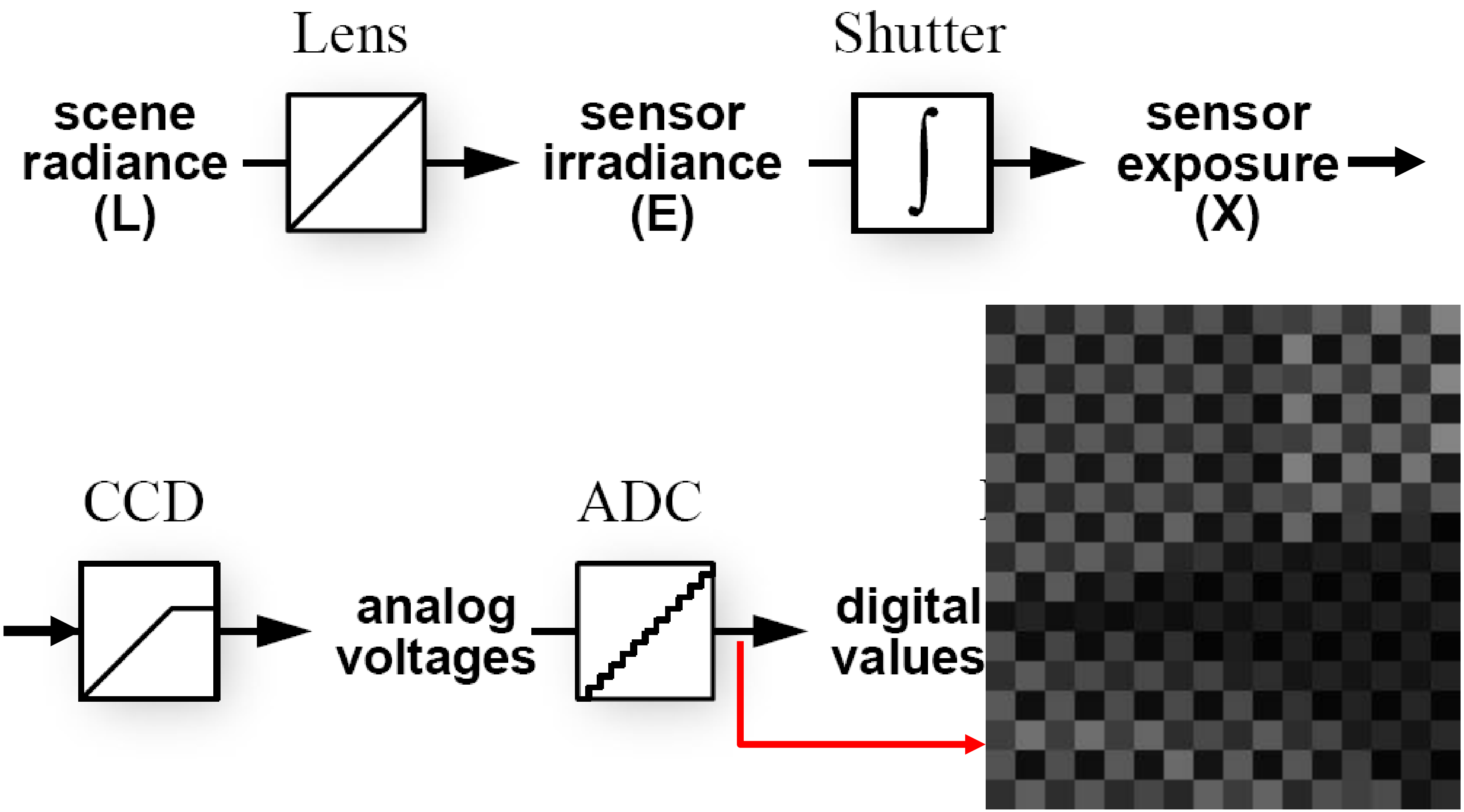
Best exposure in each region

Lens flare removal

before

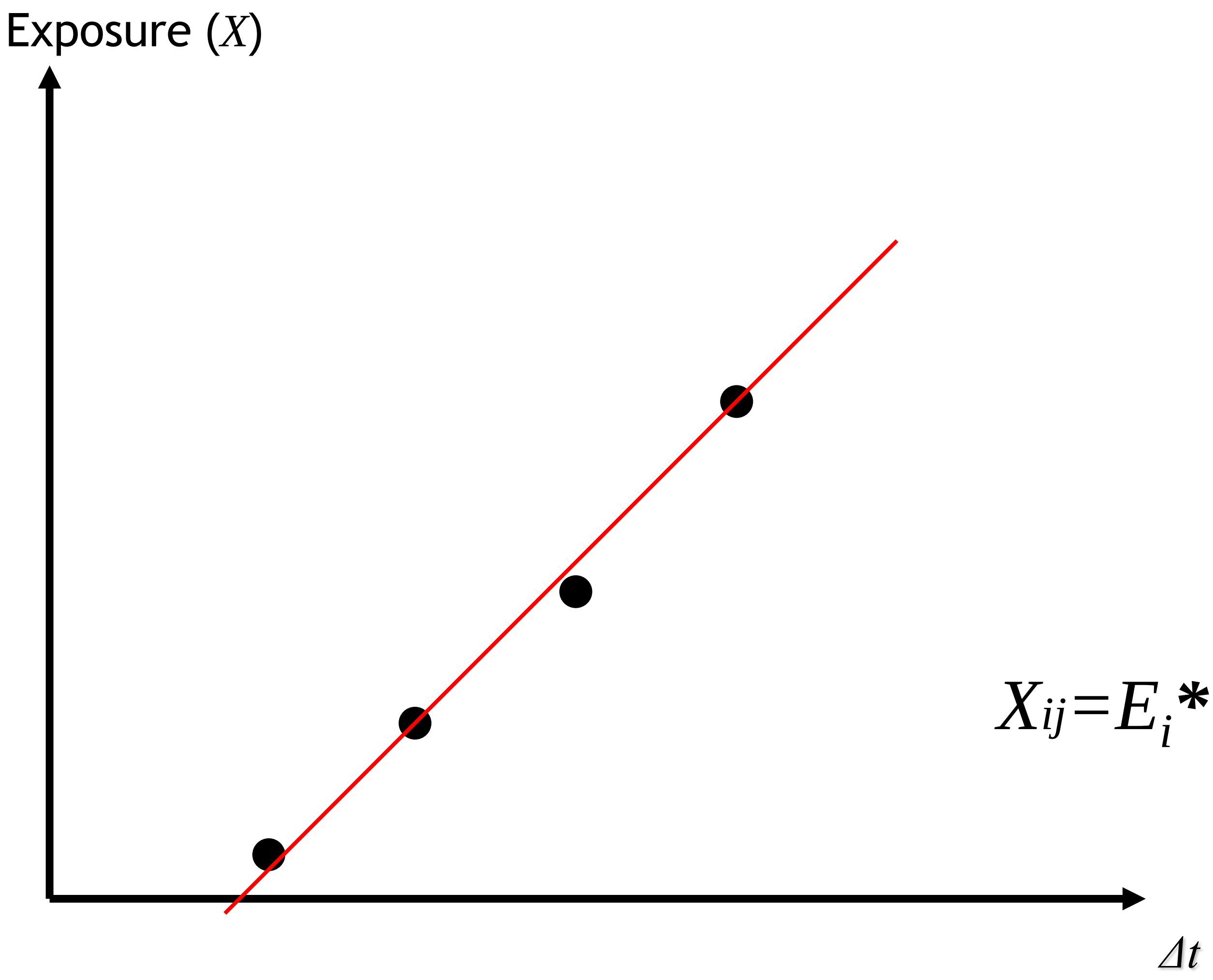
after

Easier HDR reconstruction



raw image =
12-bit CCD snapshot

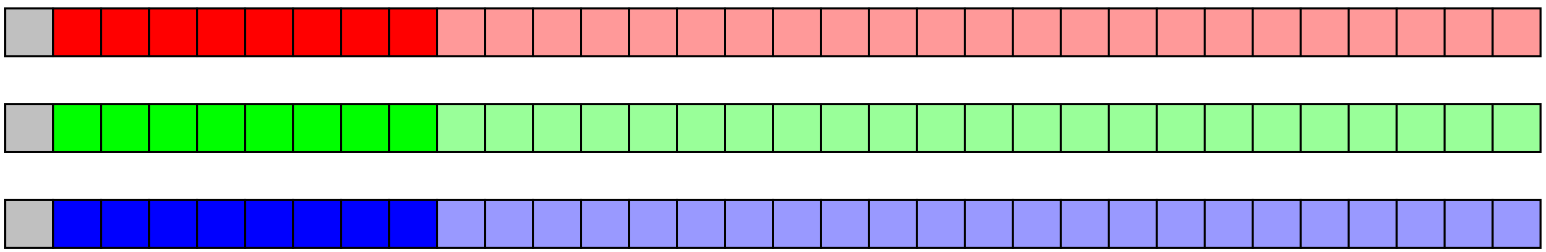
Easier HDR reconstruction



$$X_{ij}=E_i * \Delta t_j$$

Portable floatMap (.pfm)

- 12 bytes per pixel, 4 for each channel



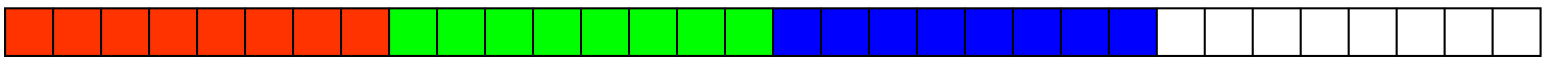
Text header similar to Jeff Poskanzer's .ppm image format:

```
PF
768 512
1
<binary image data>
```

Floating Point TIFF similar

Radiance format (.pic, .hdr, .rad)

32 bits/pixel



Red Green Blue Exponent

(145, 215, 87, 149) =

(145, 215, 87) * 2^(149-128) =

1190000 1760000 713000

(145, 215, 87, 103) =

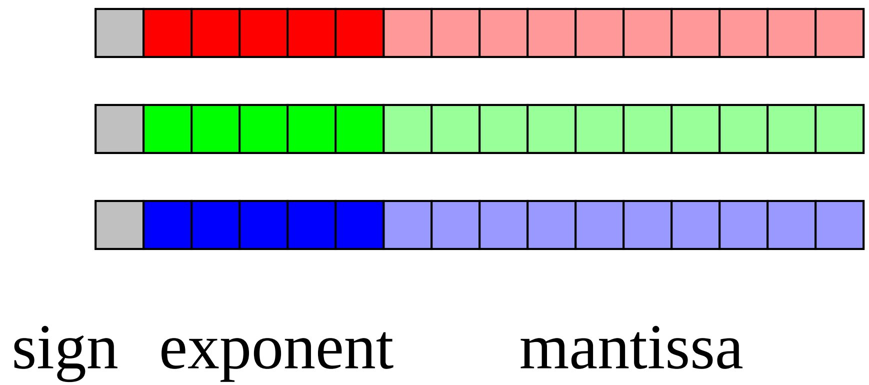
(145, 215, 87) * 2^(103-128) =

0.00000432 0.00000641 0.00000259

Ward, Greg. "Real Pixels," in Graphics Gems IV, edited by James Arvo, Academic Press, 1994

ILM's OpenEXR (.exr)

- 6 bytes per pixel, 2 for each channel, compressed



- Several lossless compression options, 2:1 typical
- Compatible with the "half" datatype in NVidia's Cg
- Supported natively on GeForce FX and Quadro FX

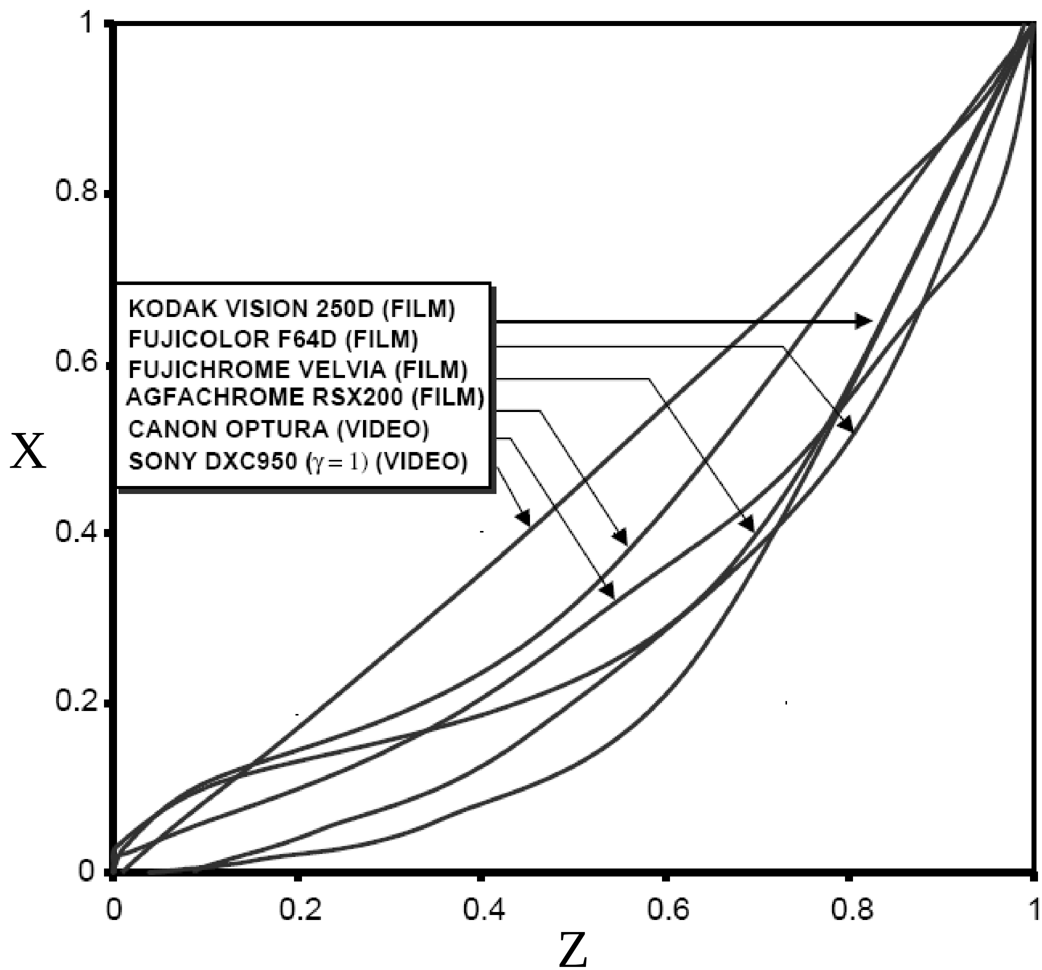
- Available at http://www.openexr.net/

Radiometric self calibration

- Assume that any response function can be modeled as a high-order polynomial

$$X = g(Z) = \sum_{m=0}^{M} c_m Z^m$$

- No need to know exposure time in advance. Useful for cheap cameras



Mitsunaga and Nayar

- To find the coefficients c_m to minimize the following

$$\varepsilon = \sum_{i=1}^{N}\sum_{j=1}^{P}\left[\sum_{m=0}^{M} c_m Z_{ij}^{m} - R_{j,j+1}\sum_{m=0}^{M} c_m Z_{i,j+1}^{m}\right]^2$$

A guess for the ratio of

$$\frac{X_{ij}}{X_{i,j+1}} = \frac{E_i \Delta t_j}{E_i \Delta t_{j+1}} = \frac{\Delta t_j}{\Delta t_{j+1}}$$

Mitsunaga and Nayar

- Again, we can only solve up to a scale. Thus, add a constraint $f(1)=1$. It reduces to M variables.
- How to solve it?

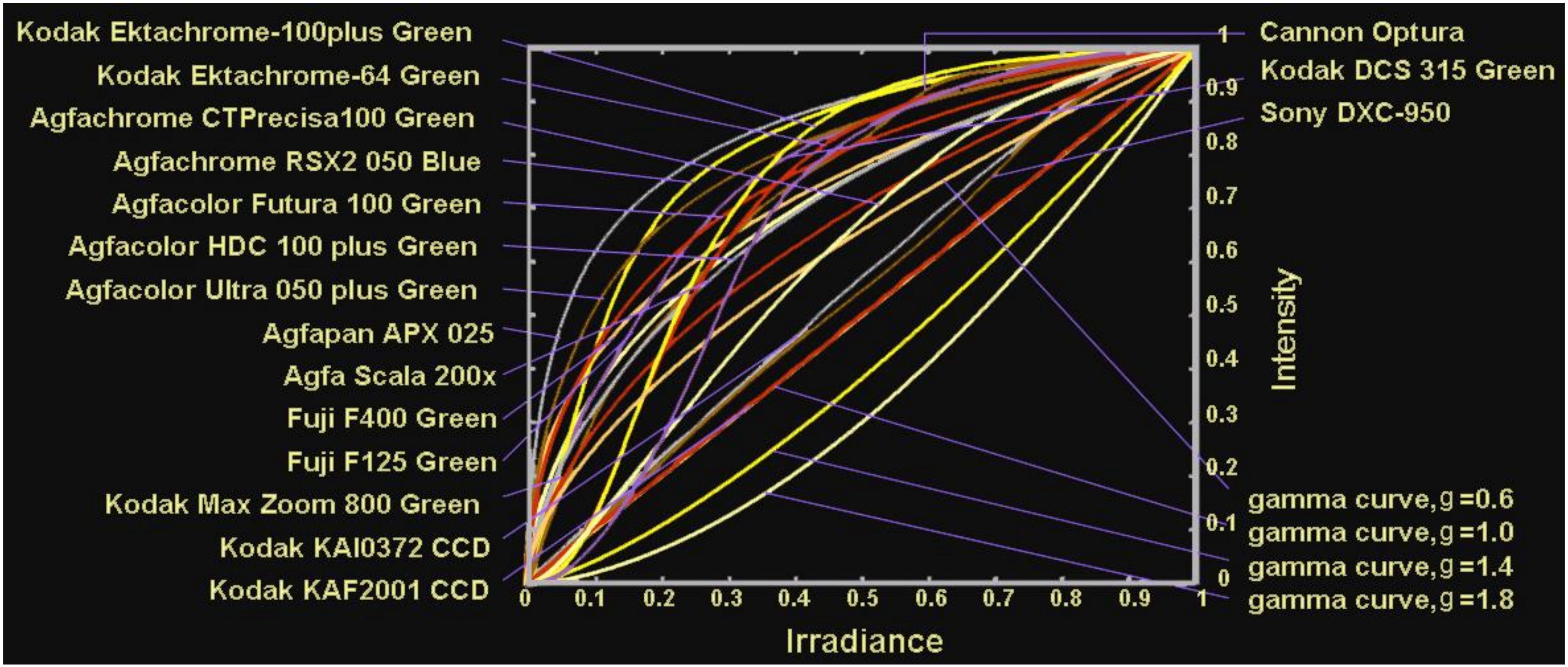
Mitsunaga and Nayar

- We solve the above iteratively and update the exposure ratio accordingly

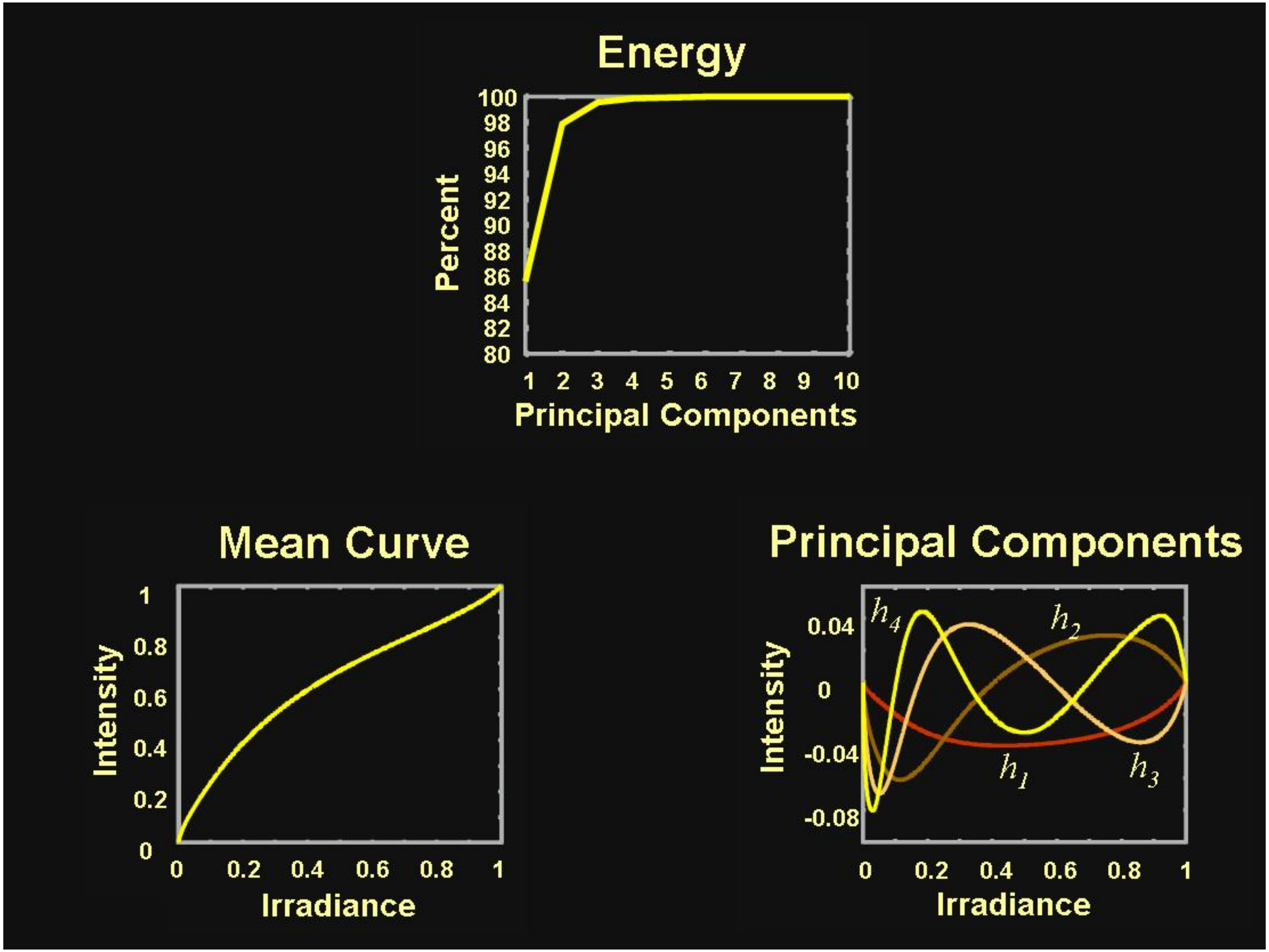
$$R_{j,j+1}^{(k)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_{m=0}^{M} c_m^{(k)}{}_k Z_{ij}^{m}}{\sum_{m=0}^{M} c_m^{(k)} Z_{i,j+1}^{m}}$$

- How to determine M? Solve up to M=10 and pick up the one with the minimal error. Notice that you prefer to have the same order for all channels. Use the combined error.

Space of response curves



Space of response curves



Robertson et. al.

DigiVFX

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

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

Given Z_{ij} and Δt_j, the goal is to find both E_i and $g(Z_{ij})$

Maximum likelihood

$$\Pr(E_i, g \mid Z_{ij}, \Delta t_j) \propto \exp\left(-\frac{1}{2} \sum_{ij} w(Z_{ij}) \left(g(Z_{ij}) - E_i \Delta t_j \right)^2 \right)$$

$$\hat{g}, \hat{E}_i = \arg\min_{g, E_i} \sum_{ij} w(Z_{ij}) \left(g(Z_{ij}) - E_i \Delta t_j \right)^2$$

Robertson et. al.

$$\hat{g}, \hat{E}_i = \arg\min_{g, E_i} \sum_{ij} w(Z_{ij}) \left(g(Z_{ij}) - E_i \Delta t_j \right)^2$$

repeat

assuming $g(Z_{ij})$ is known, optimize for E_i

assuming E_i is known, optimize for $g(Z_{ij})$

until converge

Robertson et. al.

$$\hat{g}, \hat{E}_i = \arg\min_{g, E_i} \sum_{ij} w(Z_{ij}) \left(g(Z_{ij}) - E_i \Delta t_j \right)^2$$

repeat

assuming $g(Z_{ij})$ is known, optimize for E_i

assuming E_i is known, optimize for $g(Z_{ij})$

until converge

Robertson et. al.

$$\hat{g}, \hat{E}_i = \arg\min_{g, E_i} \sum_{ij} w(Z_{ij}) \left(g(Z_{ij}) - E_i \Delta t_j \right)^2$$

repeat

assuming $g(Z_{ij})$ is known, optimize for E_i

assuming E_i is known, optimize for $g(Z_{ij})$

until converge

$$E_i = \frac{\sum_j w(Z_{ij}) g(Z_{ij}) \Delta t_j}{\sum_j w(Z_{ij}) \Delta t_j^2}$$

Robertson et. al.

$$\hat{g}, \hat{E}_i = \arg\min_{g, E_i} \sum_{ij} w(Z_{ij}) \left(g(Z_{ij}) - E_i \Delta t_j \right)^2$$

repeat

assuming $g(Z_{ij})$ is known, optimize for E_i

assuming E_i is known, optimize for $g(Z_{ij})$

until converge

$$g(m) = \frac{1}{|E_m|} \sum_{ij \in E_m} E_i \Delta t_j$$

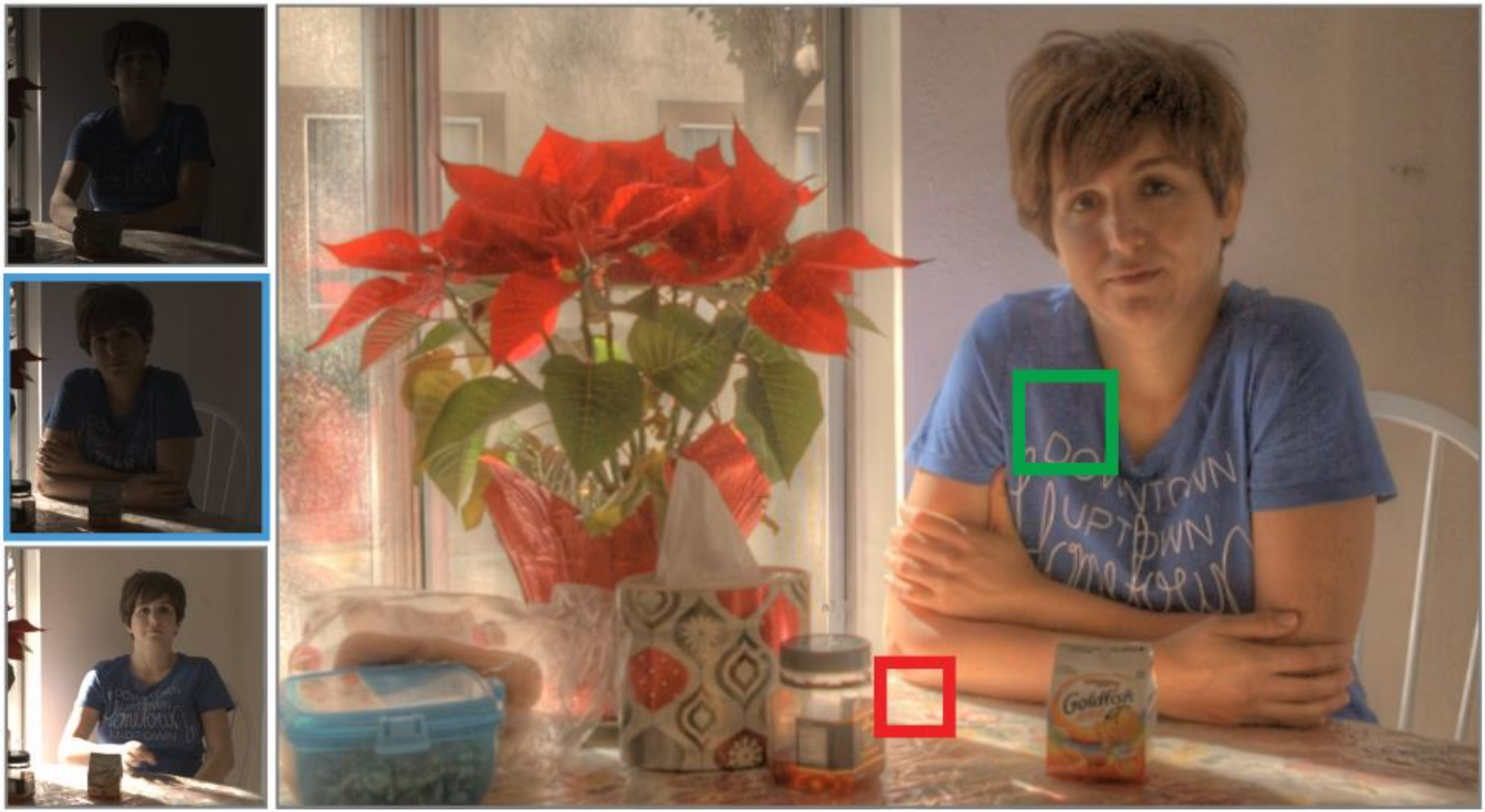
normalize so that

$g(128) = 1$

Patch-Based HDR

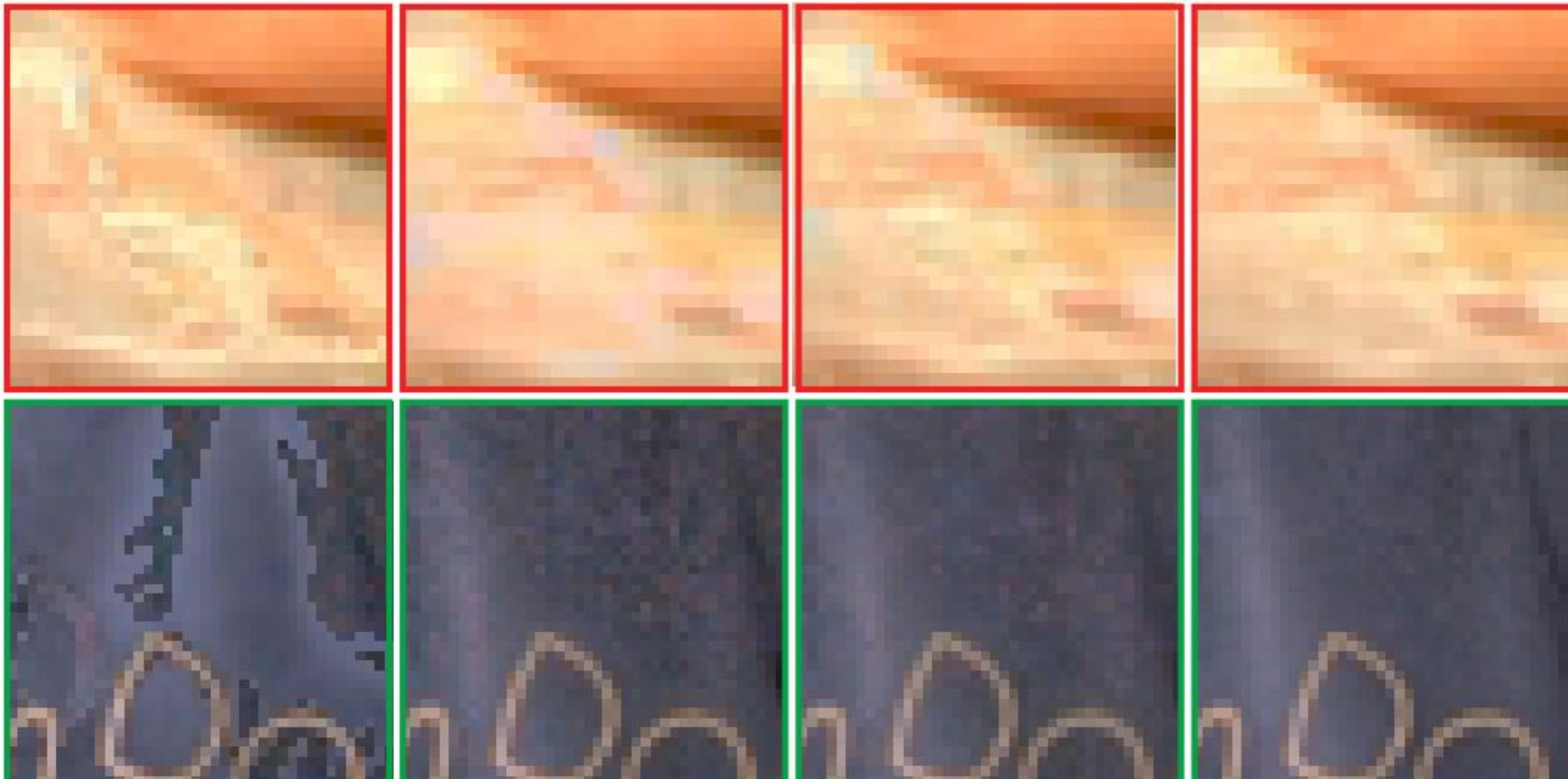
Input LDR sources   Reconstructed LDR images   Final tonemapped HDR result

Deep learning HDR assembly



LDR Images　　　　Our Tonemapped HDR Image

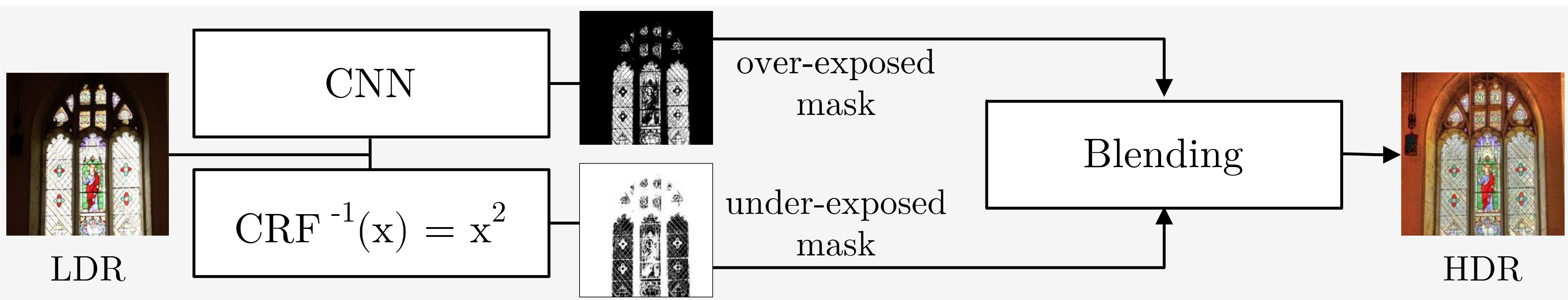
Deep learning HDR assembly



Kang (40.02 dB) | Sen (46.12 dB) | Ours (48.88 dB) | Ground Truth

Deep single-image HDR reconstruction

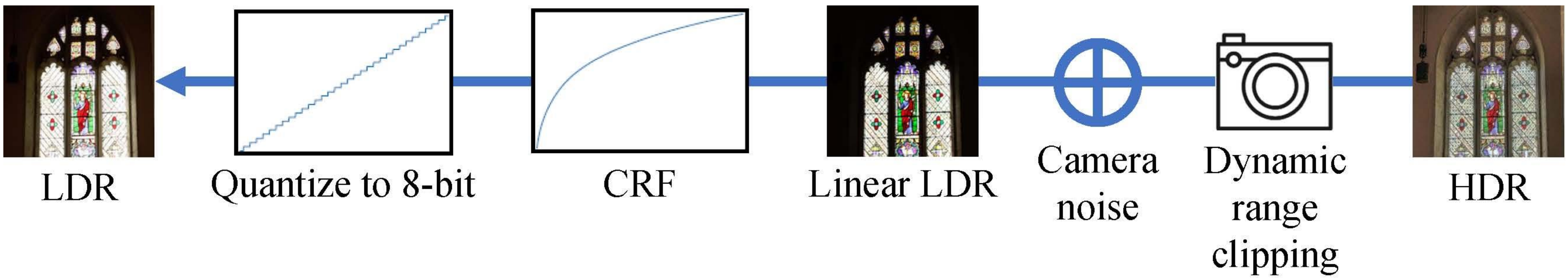
DrTMO



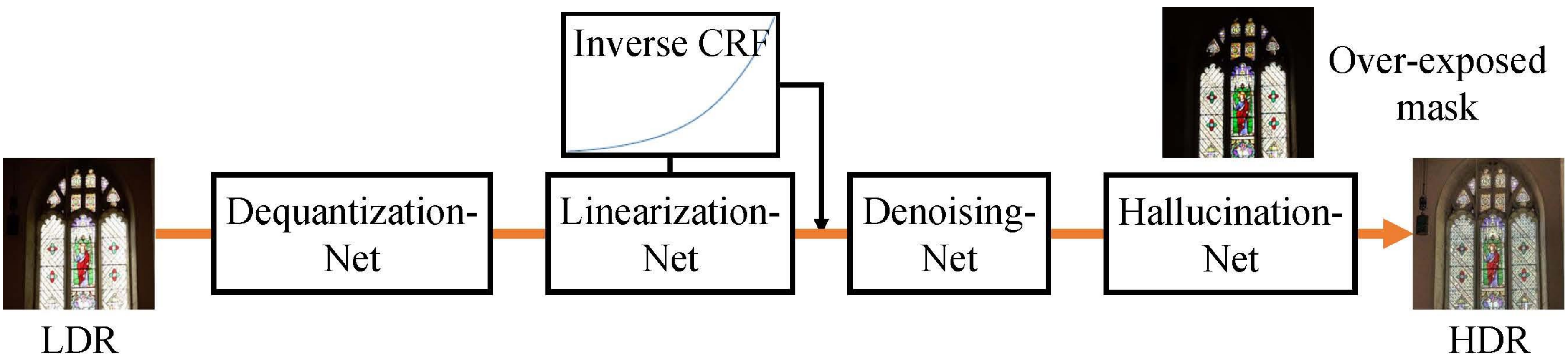
HDRCNN

ExpandNet

Learning to reverse the pipeline

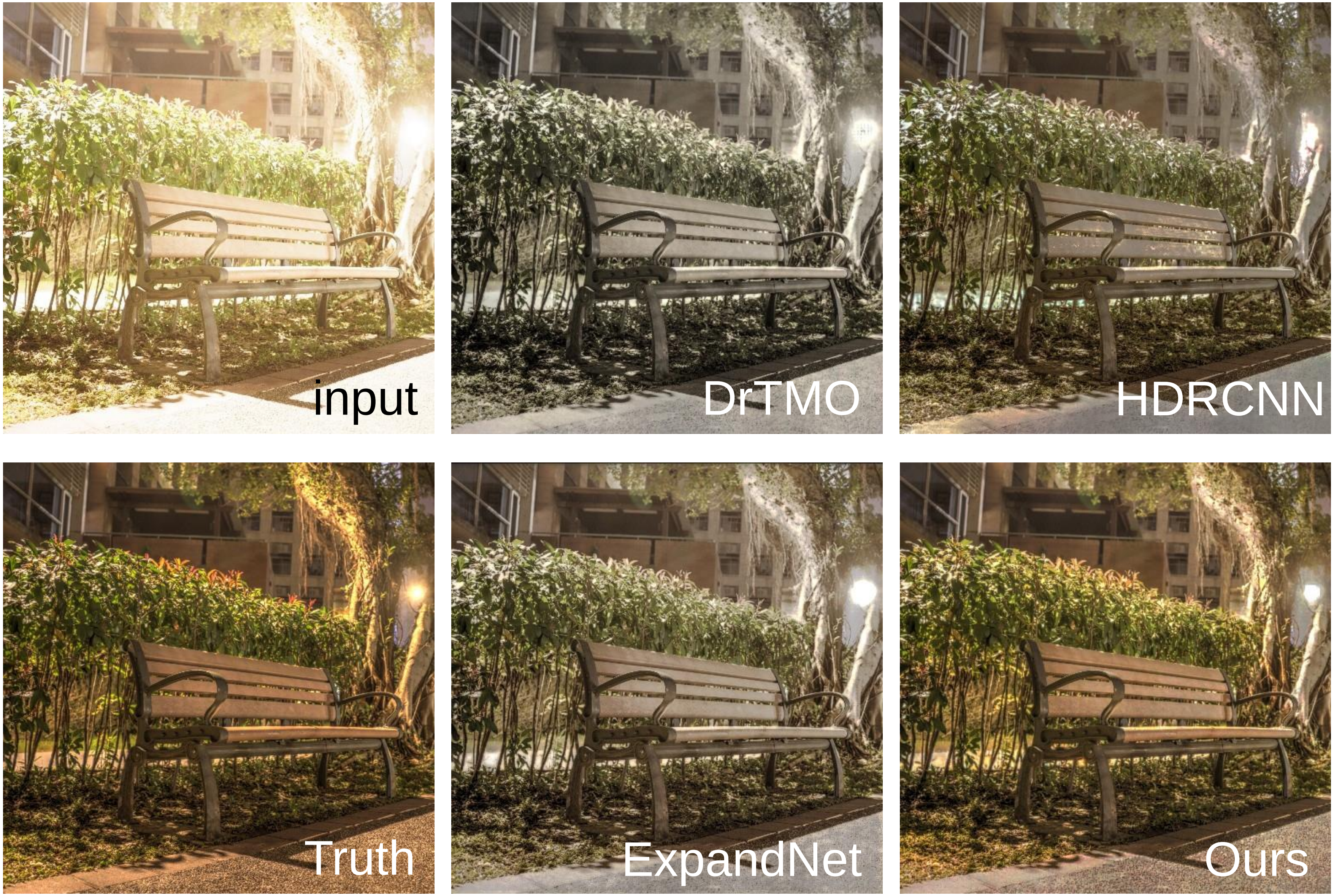


Camera pipeline



Learning to reverse the camera pipeline

Comparison



Input

Result

Input

Result

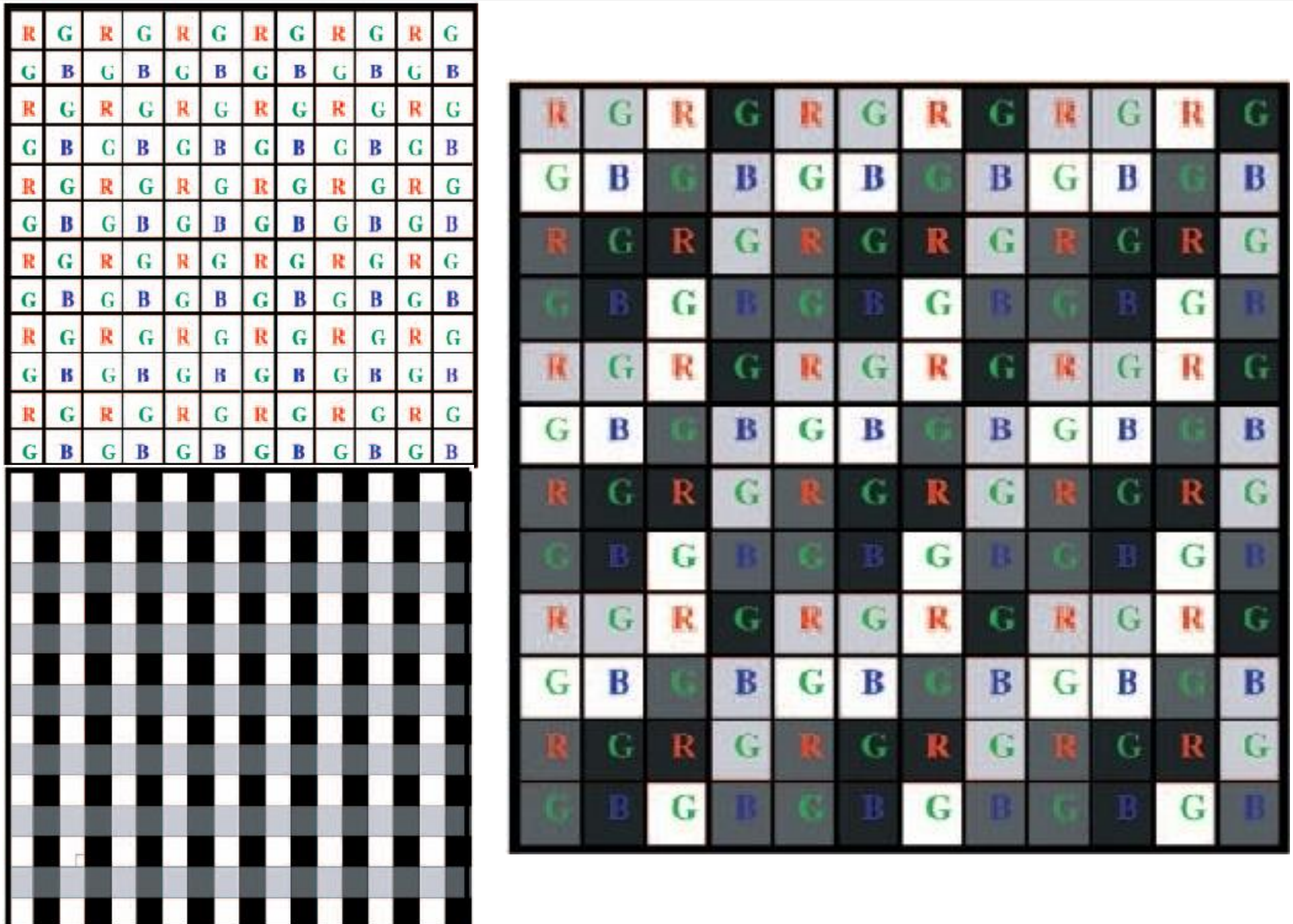
HDR Video

- **High Dynamic Range Video**
  Sing Bing Kang, Matthew Uyttendaele, Simon Winder, Richard Szeliski

  SIGGRAPH 2003

video

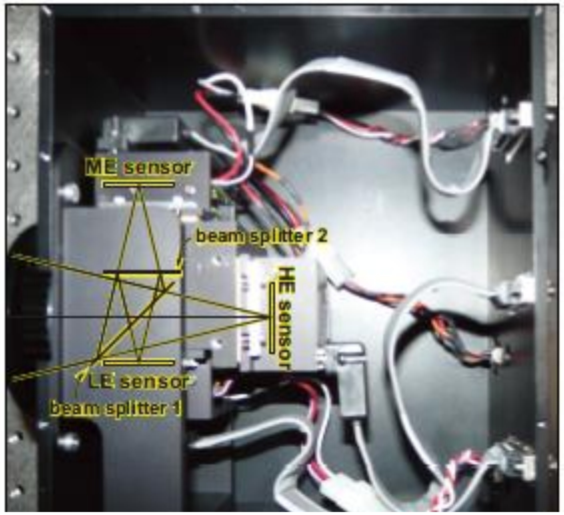
Assorted pixel

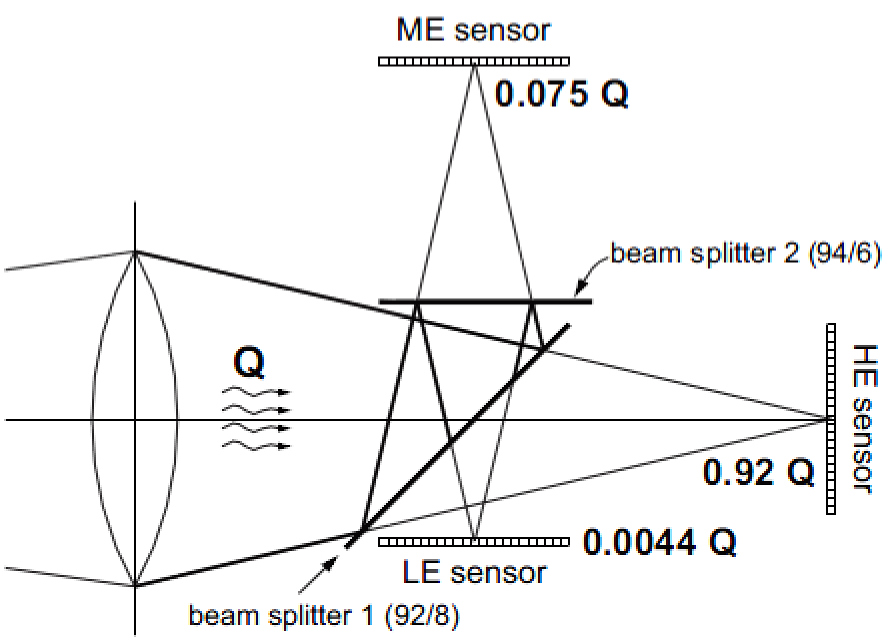


Assorted pixel

Assorted pixel

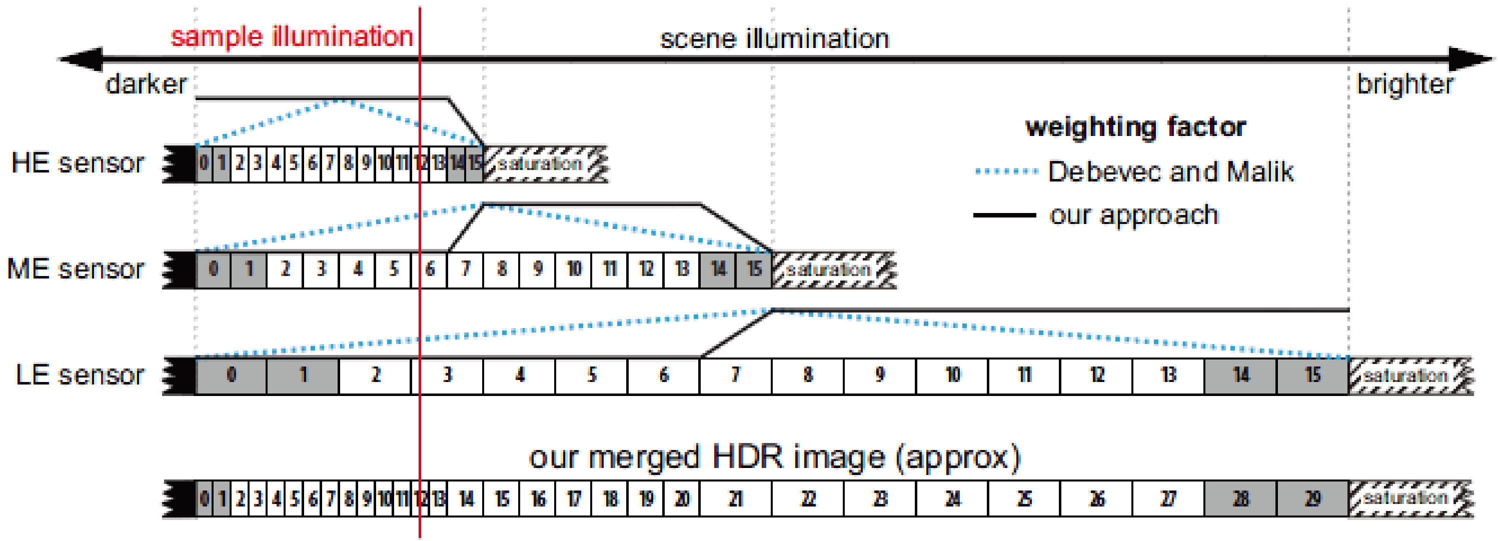
A Versatile HDR Video System

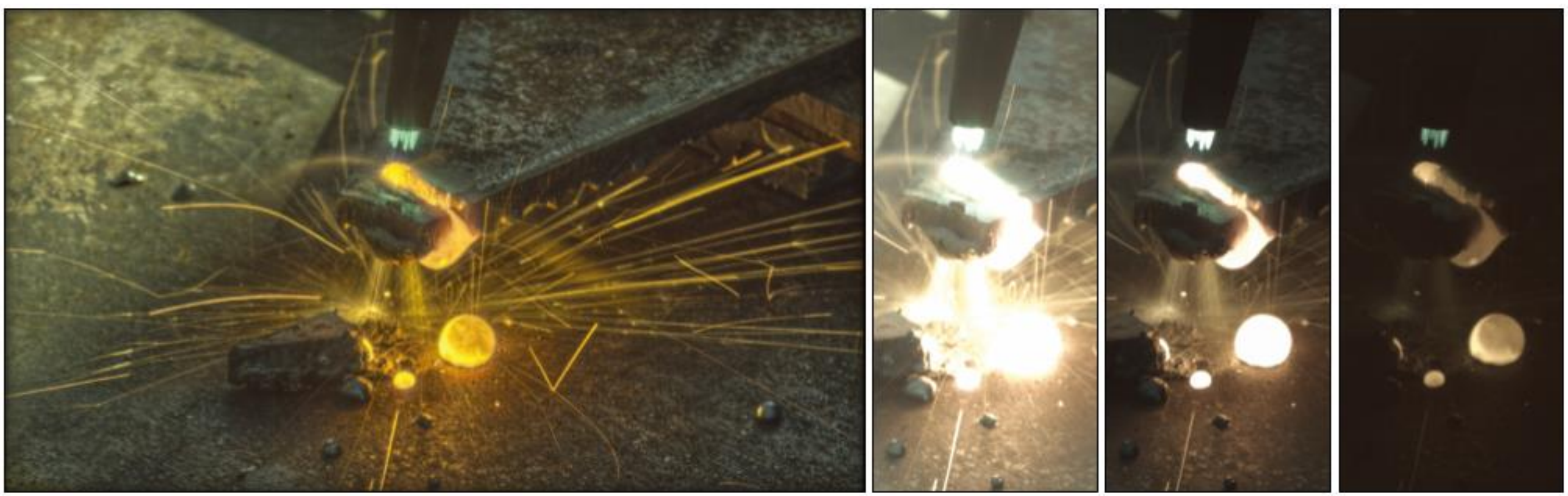




video

A Versatile HDR Video System

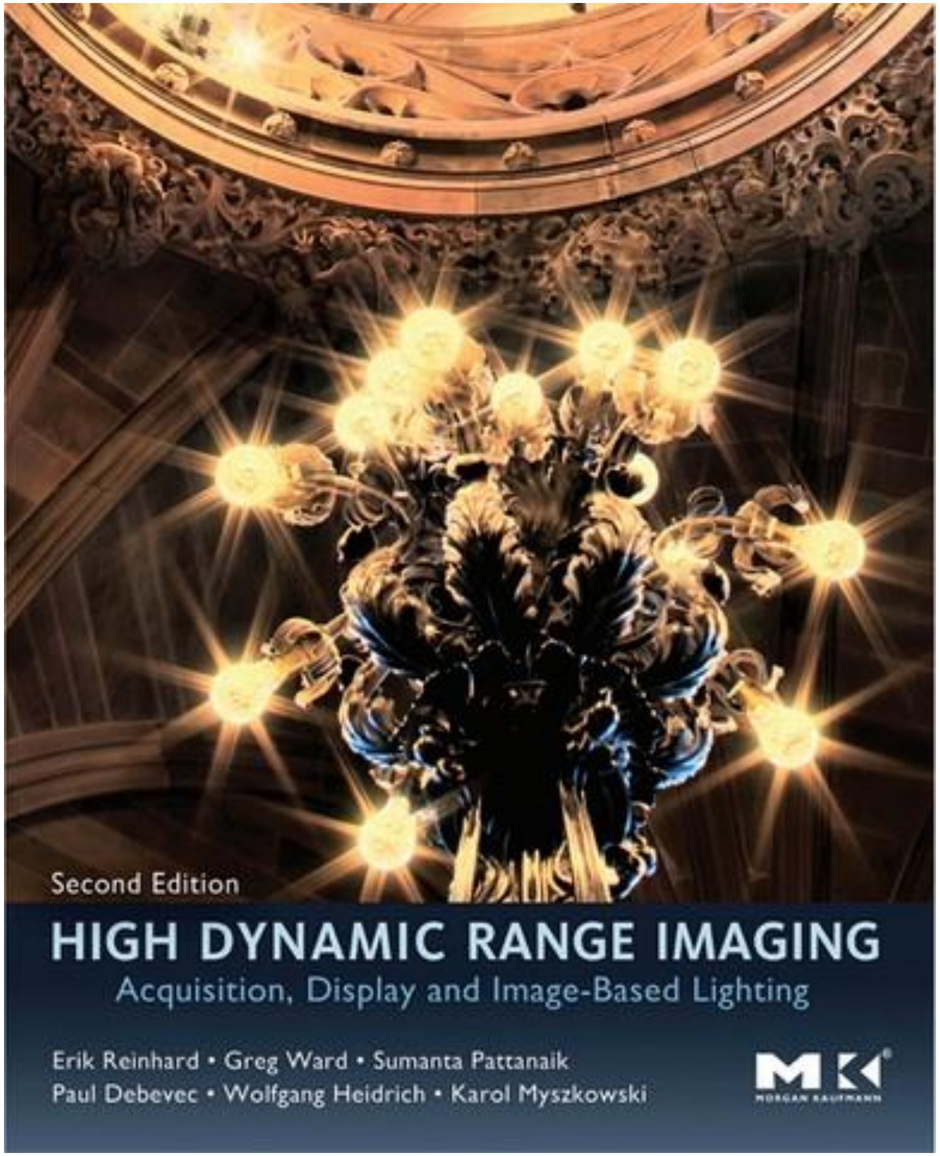




HDR becomes common practice

- Many cameras has bracket exposure modes
- For example, since iPhone 4, iPhone has HDR option. But, it could be more exposure blending rather than true HDR.

References


Second Edition
HIGH DYNAMIC RANGE IMAGING
Acquisition, Display and Image-Based Lighting
Erik Reinhard • Greg Ward • Sumanta Pattanaik
Paul Debevec • Wolfgang Heidrich • Karol Myszkowski
MK

References

- Paul E. Debevec, Jitendra Malik, Recovering High Dynamic Range Radiance Maps from Photographs, SIGGRAPH 1997.
- Tomoo Mitsunaga, Shree Nayar, Radiometric Self Calibration, CVPR 1999.
- Mark Robertson, Sean Borman, Robert Stevenson, Estimation-Theoretic Approach to Dynamic Range Enhancement using Multiple Exposures, Journal of Electronic Imaging 2003.
- Michael Grossberg, Shree Nayar, Determining the Camera Response from Images: What Is Knowable, PAMI 2003.
- Michael Grossberg, Shree Nayar, Modeling the Space of Camera Response Functions, PAMI 2004.
- Srinivasa Narasimhan, Shree Nayar, Enhancing Resolution Along Multiple Imaging Dimensions Using Assorted Pixels, PAMI 2005.
- G. Krawczyk, M. Goesele, H.-P. Seidel, Photometric Calibration of High Dynamic Range Cameras, MPI Research Report 2005.
- G. Ward, Fast Robust Image Registration for Compositing High Dynamic Range Photographs from Hand-held Exposures, jgt 2003.