Introduction to Computer Graphics

2016 Spring

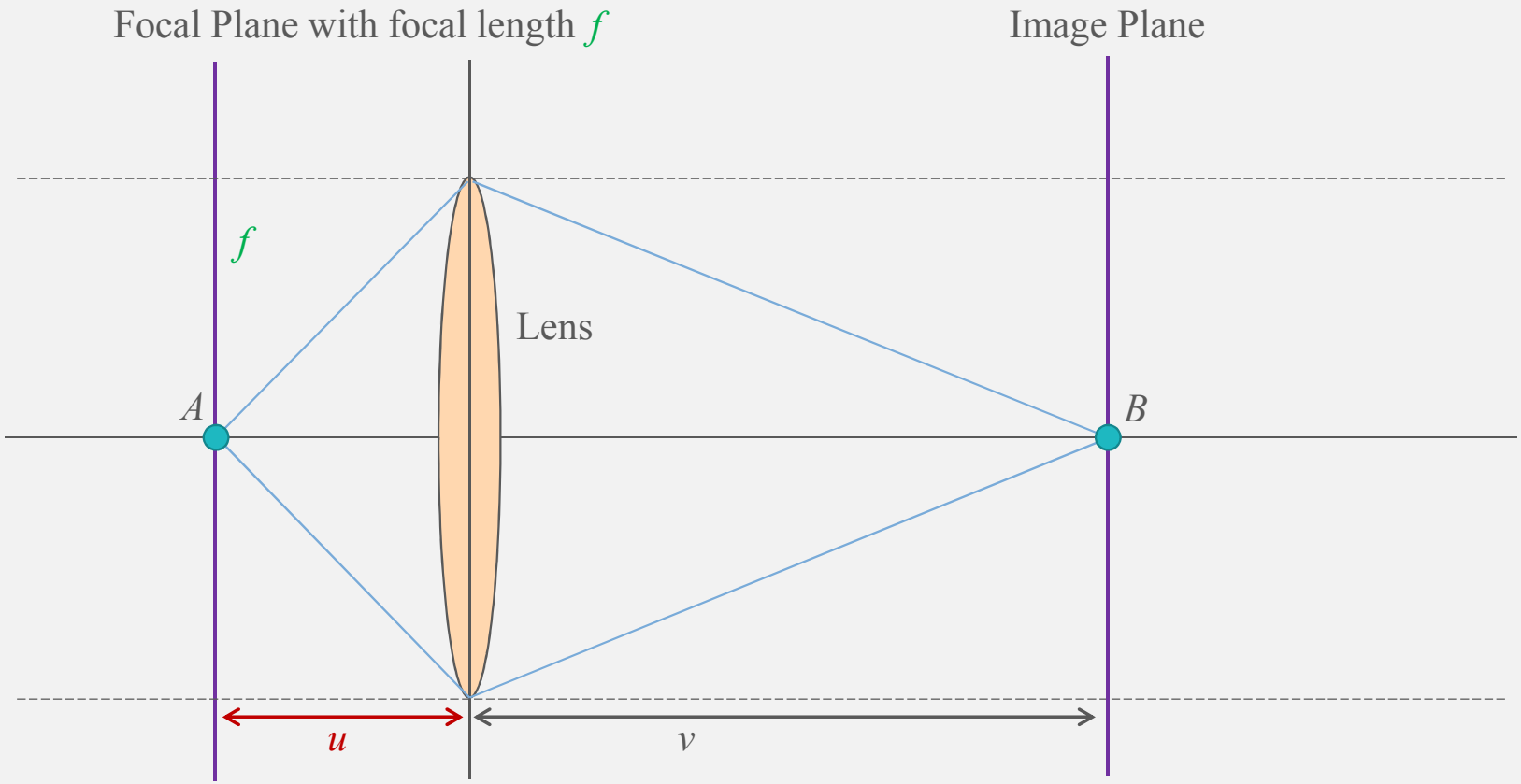
National Cheng Kung University

Instructors: Min-Chun Hu 胡敏君

Shih-Chin Weng 翁士欽 (西基電腦動畫)

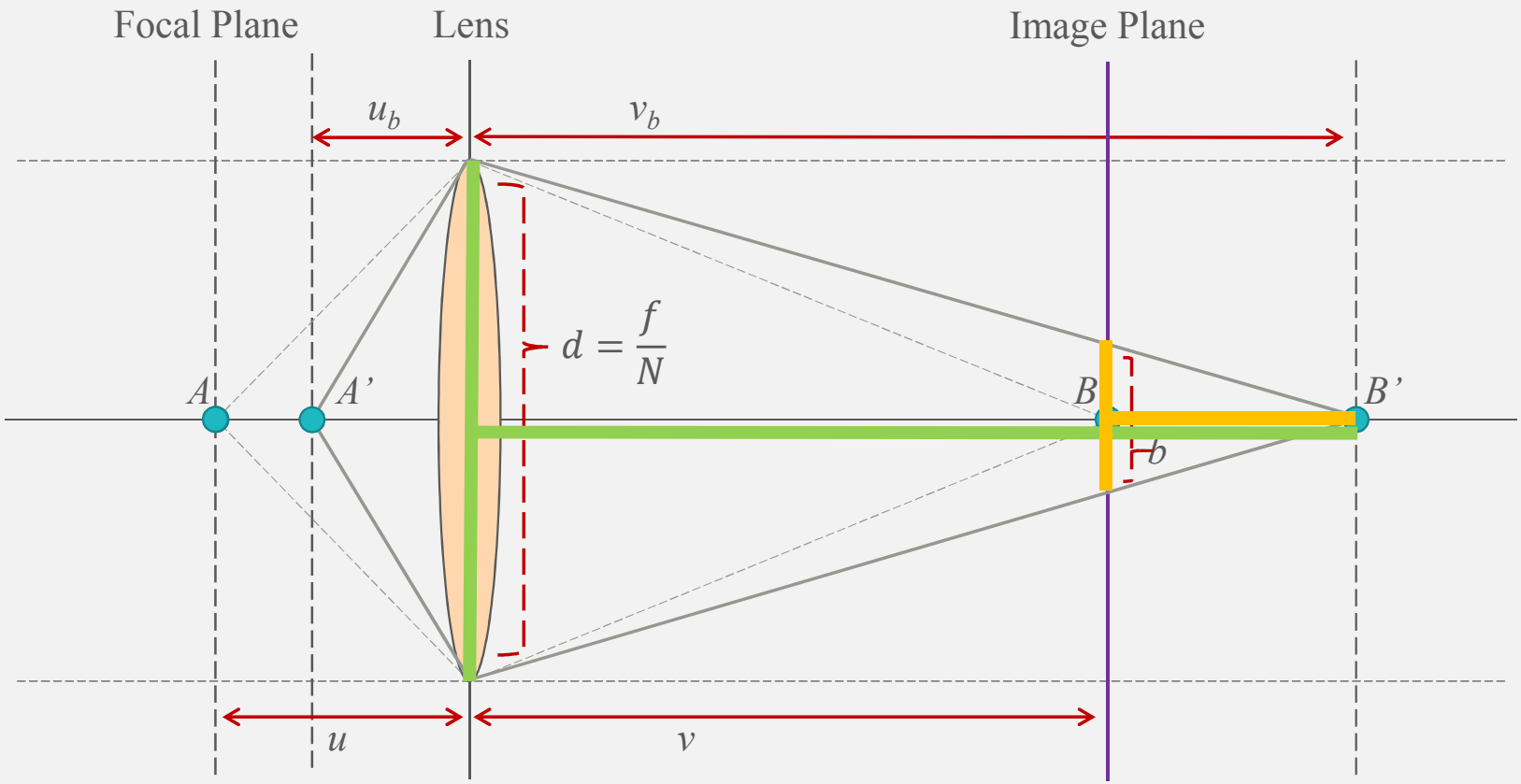
Depth of Field (DOF) Rendering

DOF Effect



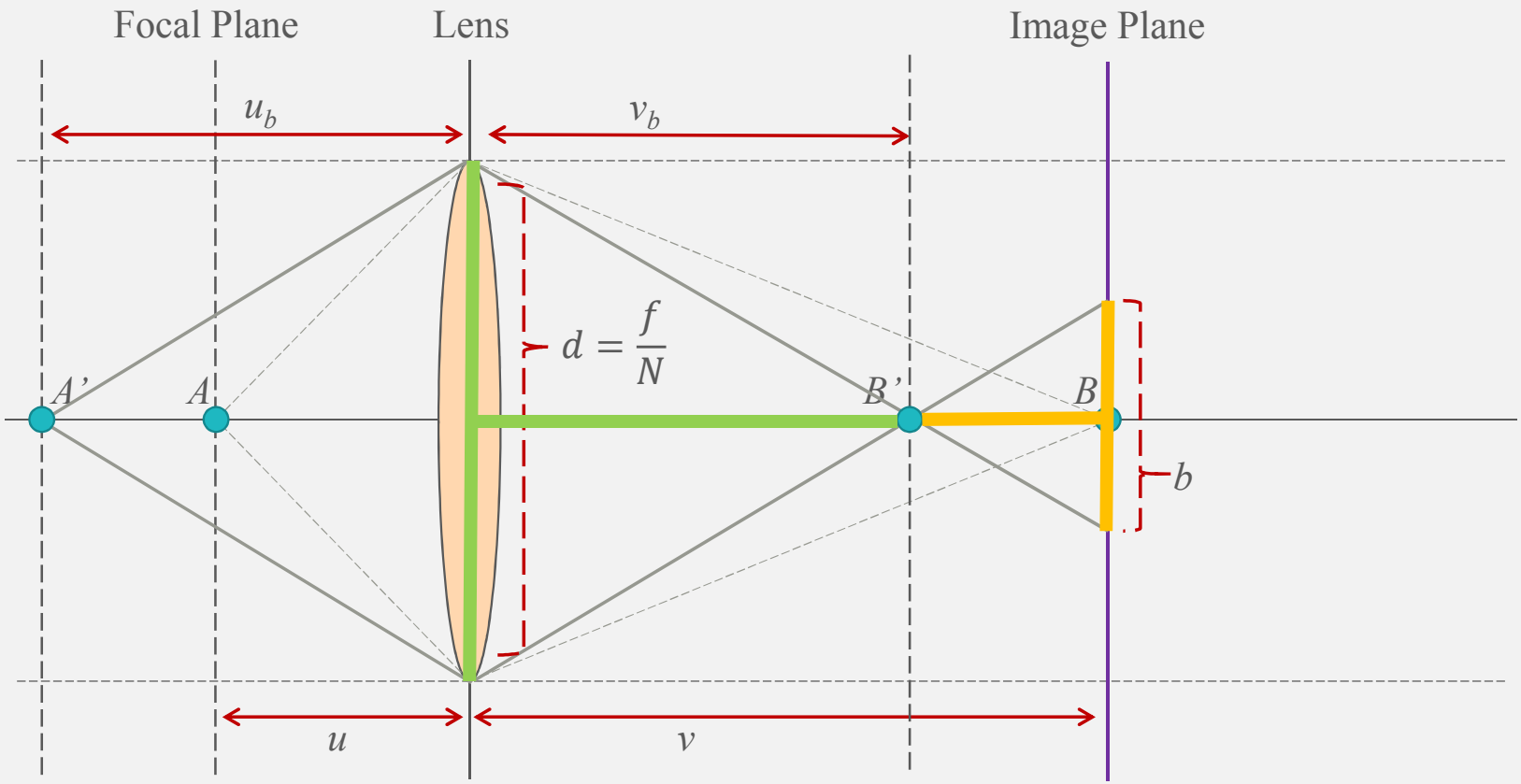
$$\frac{1}{u}+\frac{1}{v}=\frac{1}{f}$$

$$v=\frac{u\cdot f}{u-f}$$

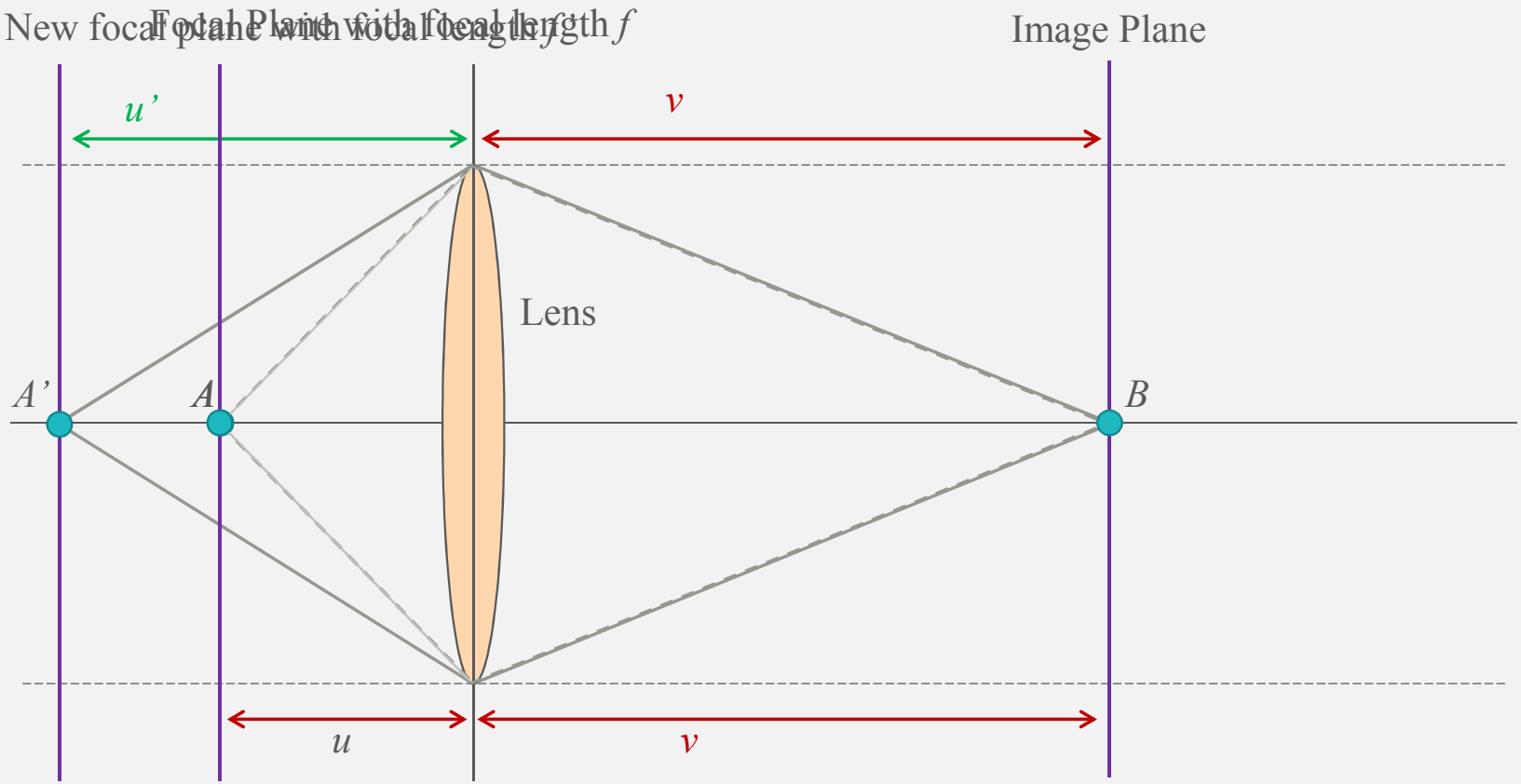


$$b = \frac{d(v_b - v)}{v_b} = \frac{f|(u_b - u)|f}{N(u_b - f)u_b}$$

$$DPI = \frac{1}{b}$$



$$b = DPI \cdot \frac{d(v - v_b)}{v_b} = DPI \cdot \frac{f|(u_b - u)|f}{N(u - f)u_b}$$



$$\frac{1}{u'} + \frac{1}{v} = \frac{1}{f'}, \qquad v_1 = \frac{u \cdot f}{u - f}$$

$$\frac{1}{u_b} + \frac{1}{v_b} = \frac{1}{f'}, \qquad v_1 = \frac{u' * u'}{u' - f'}$$

DOF Rendering Methods

A. Multi-pass approach **Heavy computational cost.**
- Distributed ray tracing
- Accumulation buffer method

B. Post-filtering approach using a single layer
- Scatter-based method **Require heavy sorting of entire depth.**
- Gather -based method **Very fast but involve two severe visual artifacts. (intensity leakage and blurring discontinuity)**

C. Post-filtering approach using multiple layers **Require sufficient layers.**
1. Decomposing a pinhole image into several sub-images
2. Blurring the sub-images are separately
3. blending from farther to nearer depths

Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation

Step 1:

Construction of Mipmap Pyramid.

Step 2:

Computing CoC from Depth and relating CoC to Mipmap Level.

Step 3:

Nonlinear Mipmap Interpolation.

[Ref] S. Lee; J. Kim, G. S. C. 2009. Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. *IEEE Transactions on Visualization and Computer Graphics*.

Step 1: Construction of Mipmap Pyramid

$I_p(0)$

$I_p(1)$

quarter-sized

$I_p(2)$

quarter-sized

.
.
.

$$I_p(l) = w_G \sum_{q \in \Omega} G(q-p) I_q(l-1)$$

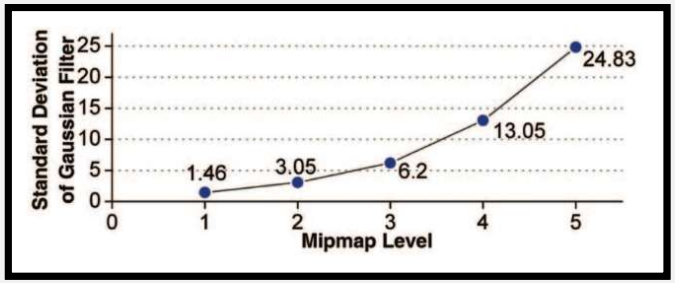
Step 2: Computing CoC from Depth and Relating CoC to Mipmap Level.

Depth Map

$$b = DPI \cdot \frac{f|(u_b - u)|f}{N(u - f)u_b}$$

$$\frac{b}{2} = \alpha, \qquad \alpha = \frac{3}{2} \cdot 2^{m-1}$$

(α: Degree of blur)

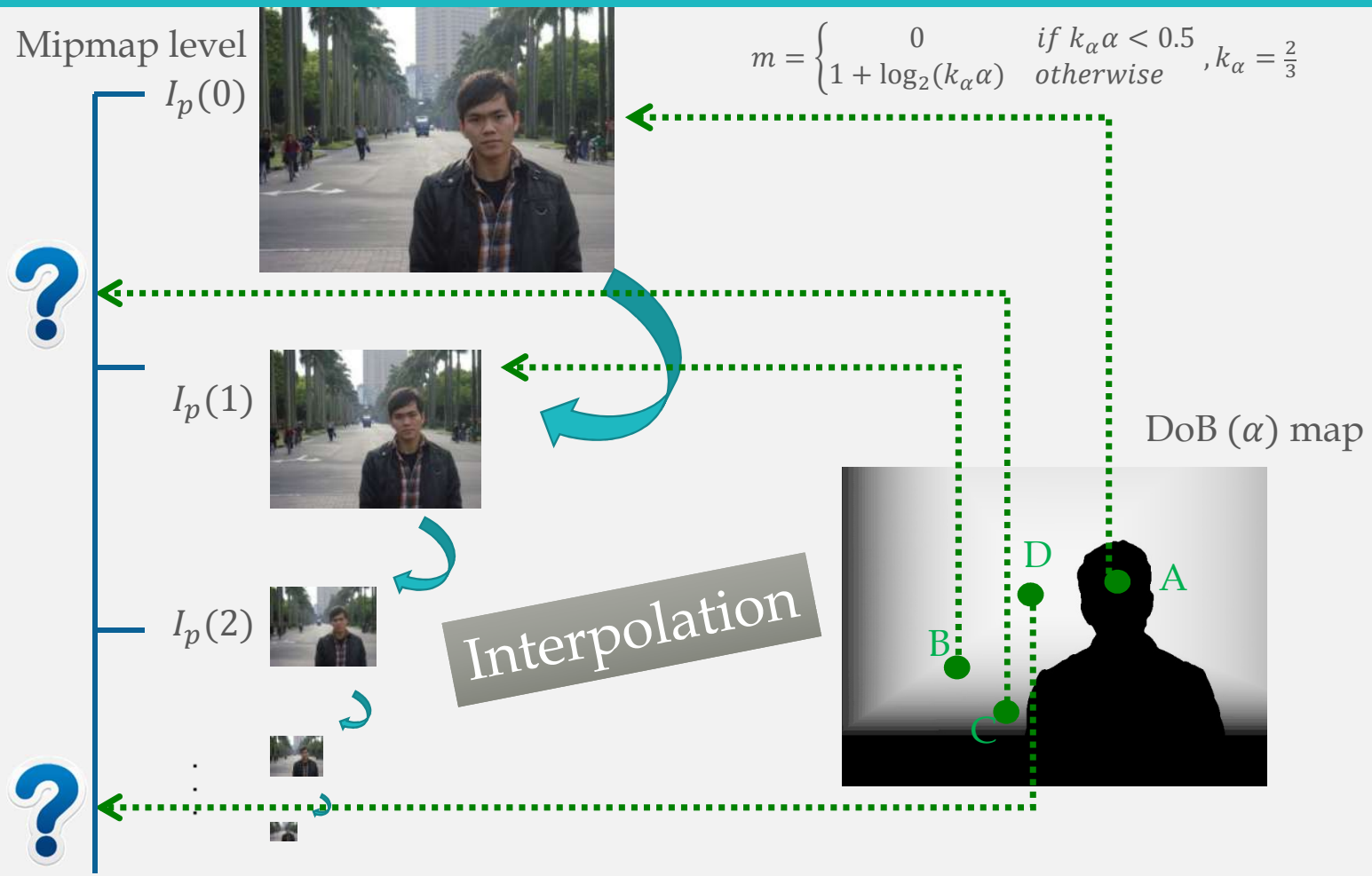


$$m = \begin{cases} 0 & if\ k_\alpha \alpha < 0.5 \\ 1 + log_2(k_\alpha \alpha) & otherwise \end{cases}, k_\alpha = \frac{2}{3} \quad (1)$$

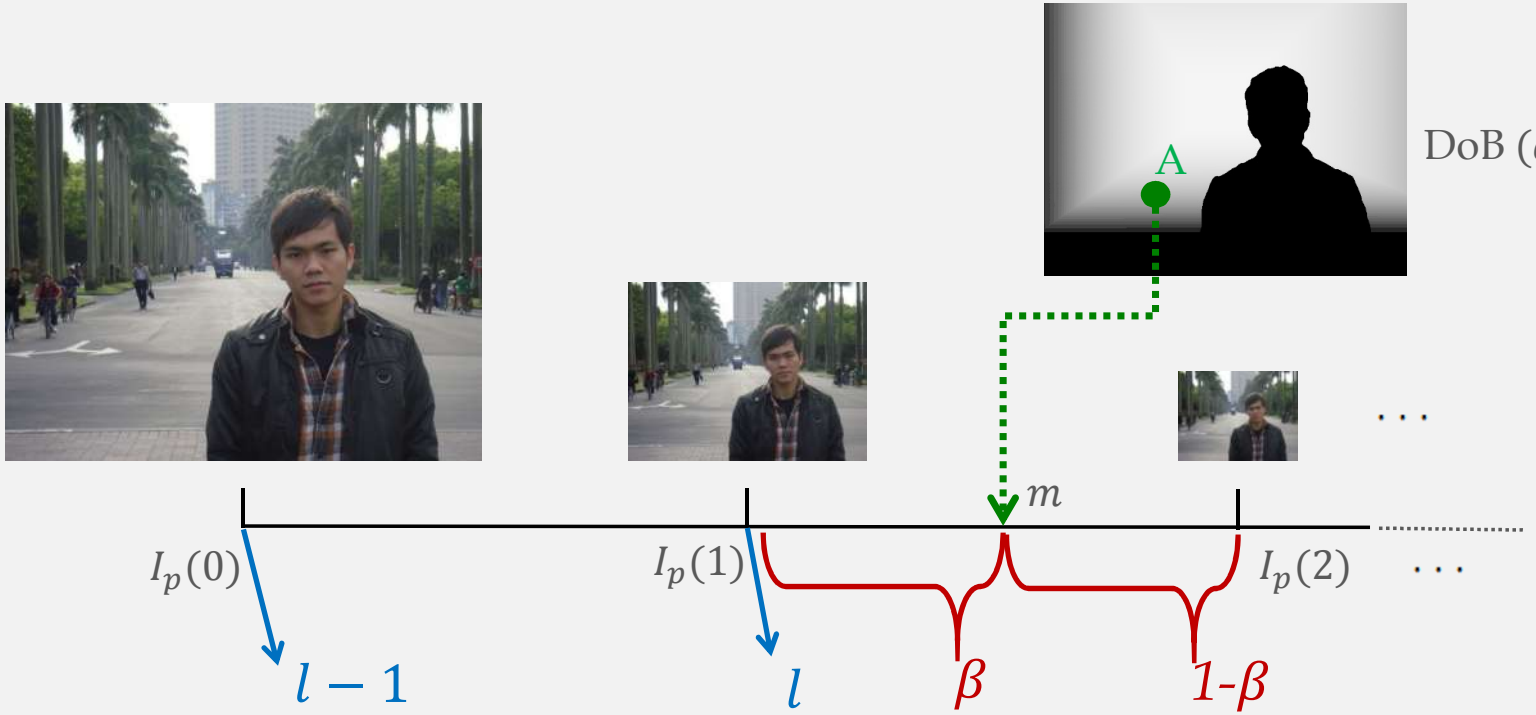
Focal length
Aperture size
DPI

DoB (α) Map

Step 3: Nonlinear Mipmap Interpolation.

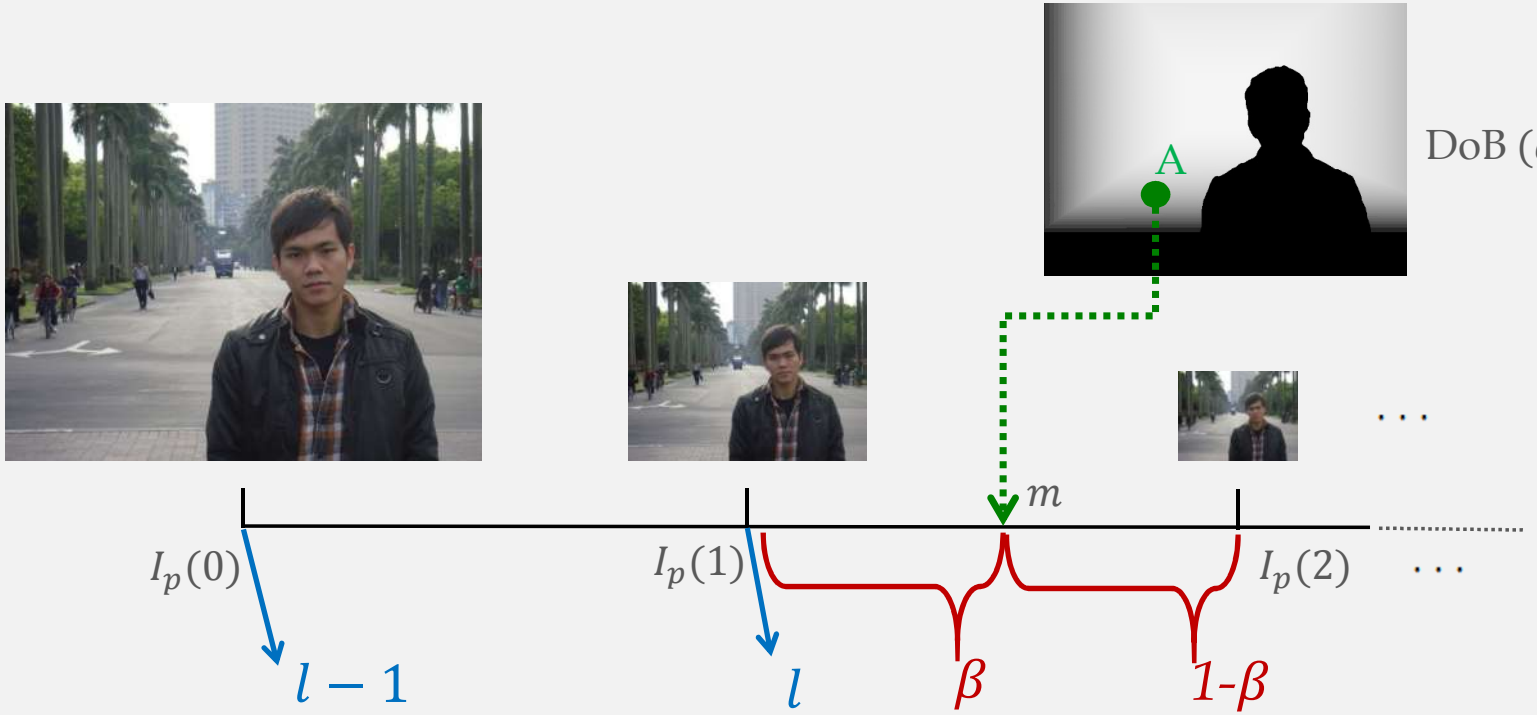


Step 3: Nonlinear Mipmap Interpolation.



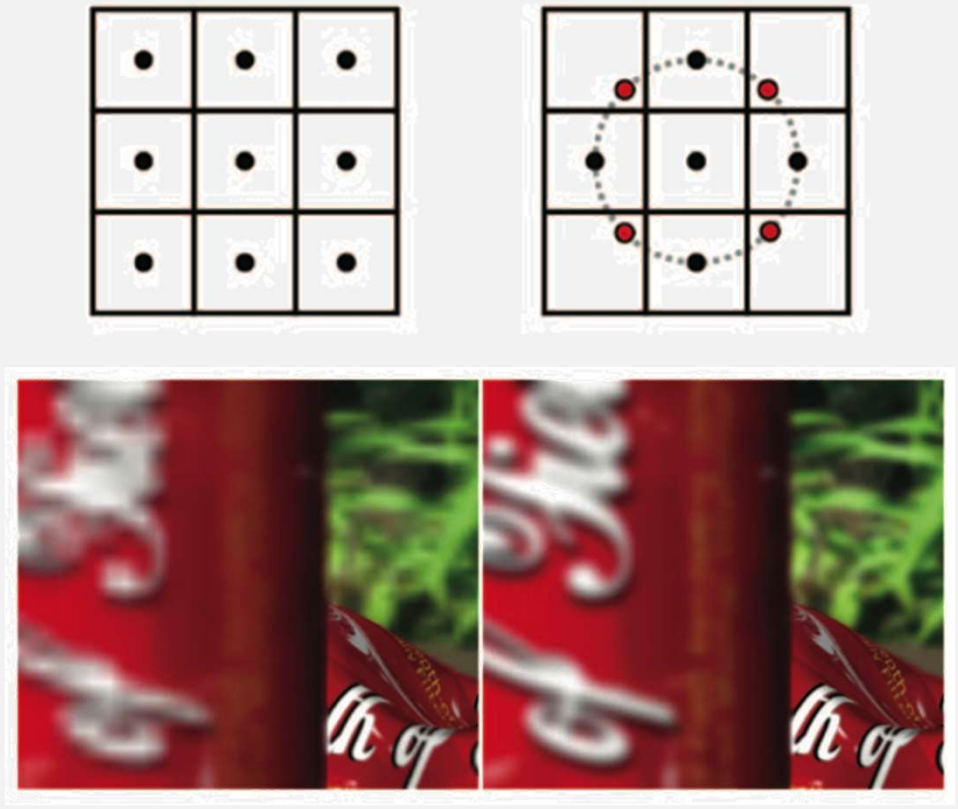
$$l = \lfloor m \rfloor, \quad \beta = m - l$$

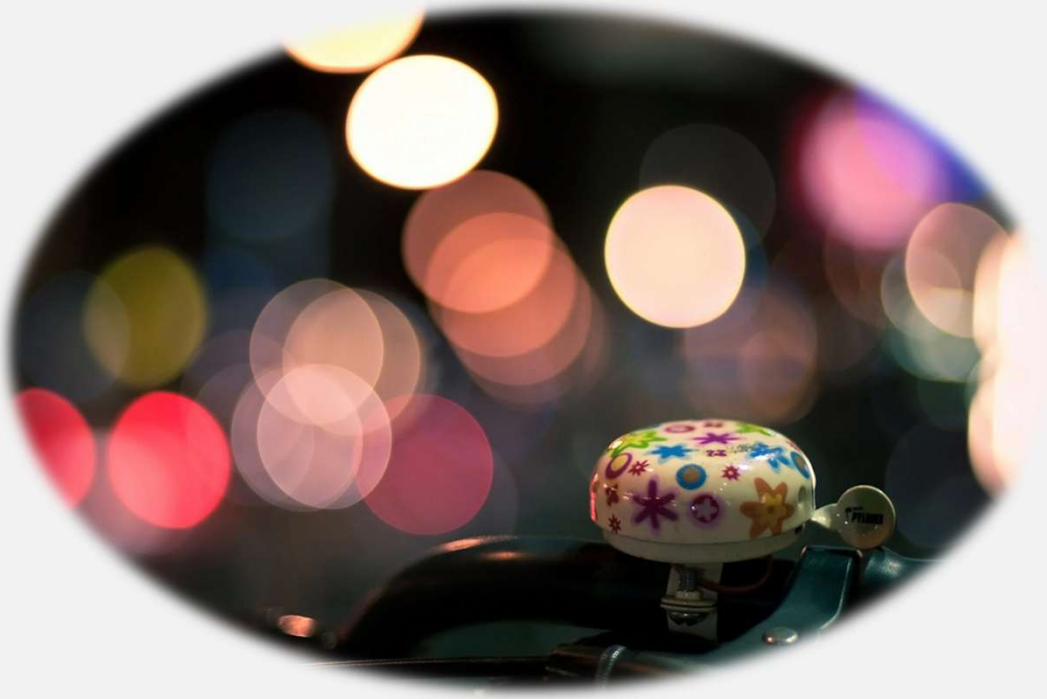
Step 3: Nonlinear Mipmap Interpolation.



$$I_p(m) = w_G \sum_{q \in \Omega} G(q-p) \cdot I_q(l-1)^{1-\beta} \cdot I_q\,(l)^{\beta}$$

Finer Mipmap Images





Apply “Bokeh” effects

Extension method for Bokeh Effect

Step 1:

Intensity-preserved image construction.

Step 2:

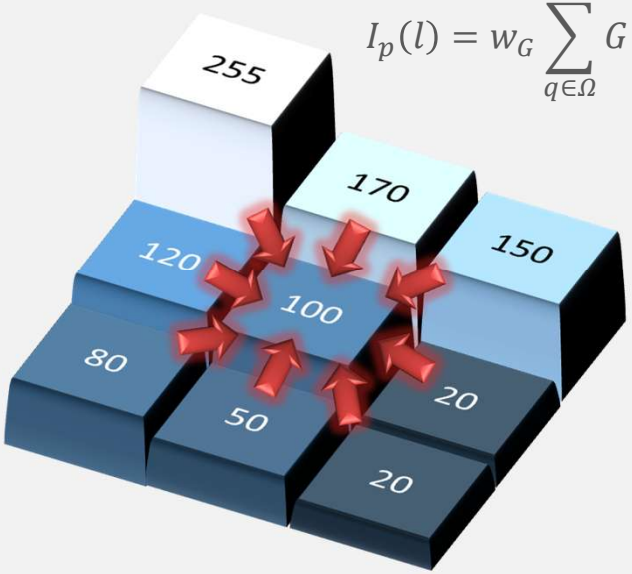
Bokeh probability calculation.

Step 3:

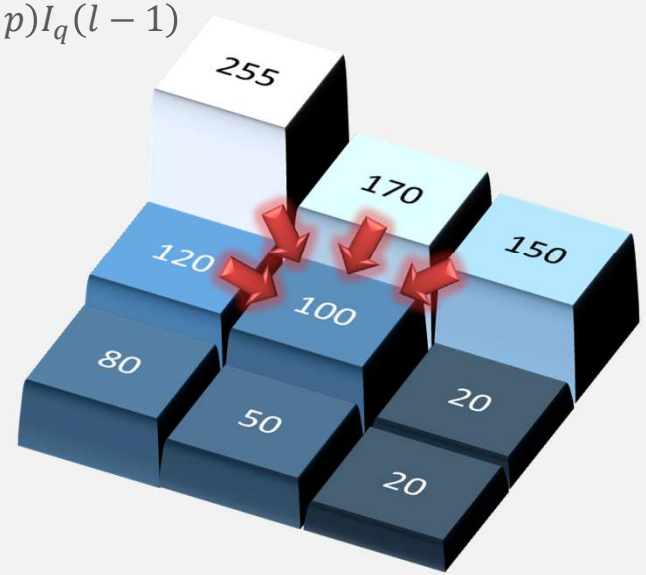
DOF rendering with Bokeh effect.

Step 1: Intensity-preserved image construction

$$I_p(l) = w_G \sum_{q \in \Omega} G(q-p) I_q(l-1)$$



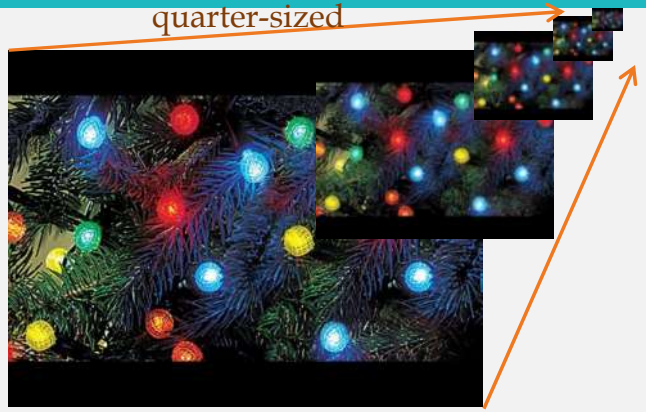
The isotropic Gaussian filter.



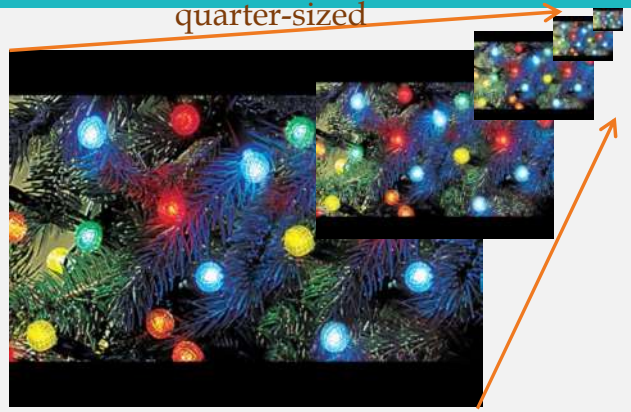
The anisotropic Gaussian filter.

$$G'(p,q,l) = \begin{cases} 0 & \text{if } I_{\boldsymbol{p}}(l) > I_{\boldsymbol{q}}(l) + \delta_{\mathbf{2}} \\ G(q-p) & otherwise \end{cases}$$

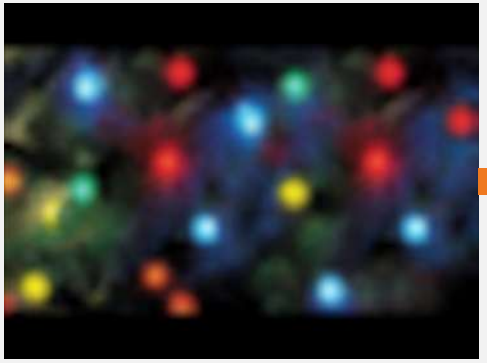
Step 1: Intensity-preserved image construction



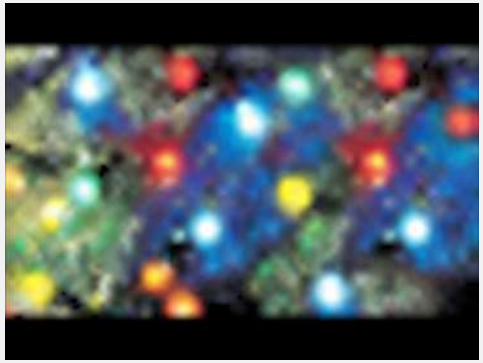
The isotropic Gaussian filter.



The anisotropic Gaussian filter.

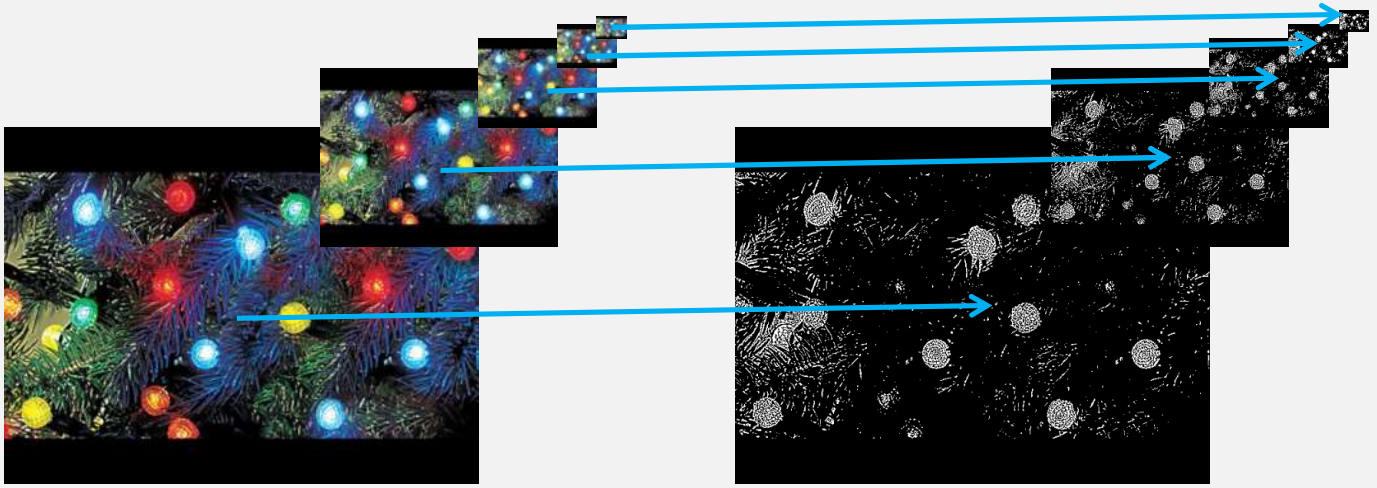


Rendering without intensity preserving.



The intensity preserved image.

Step 2: Bokeh Probability Calculation

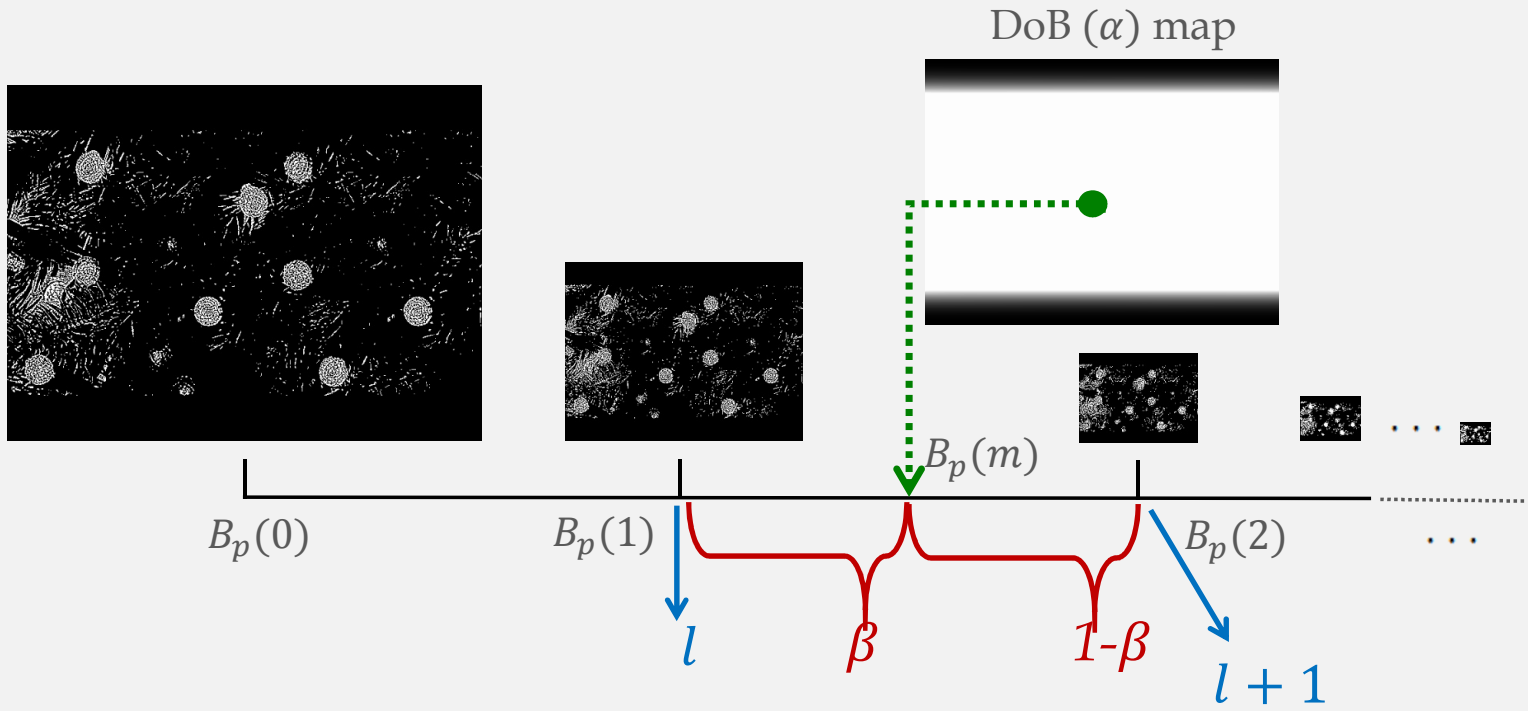


The anisotropic Gaussian filter.

Positions of Bokeh occurrence

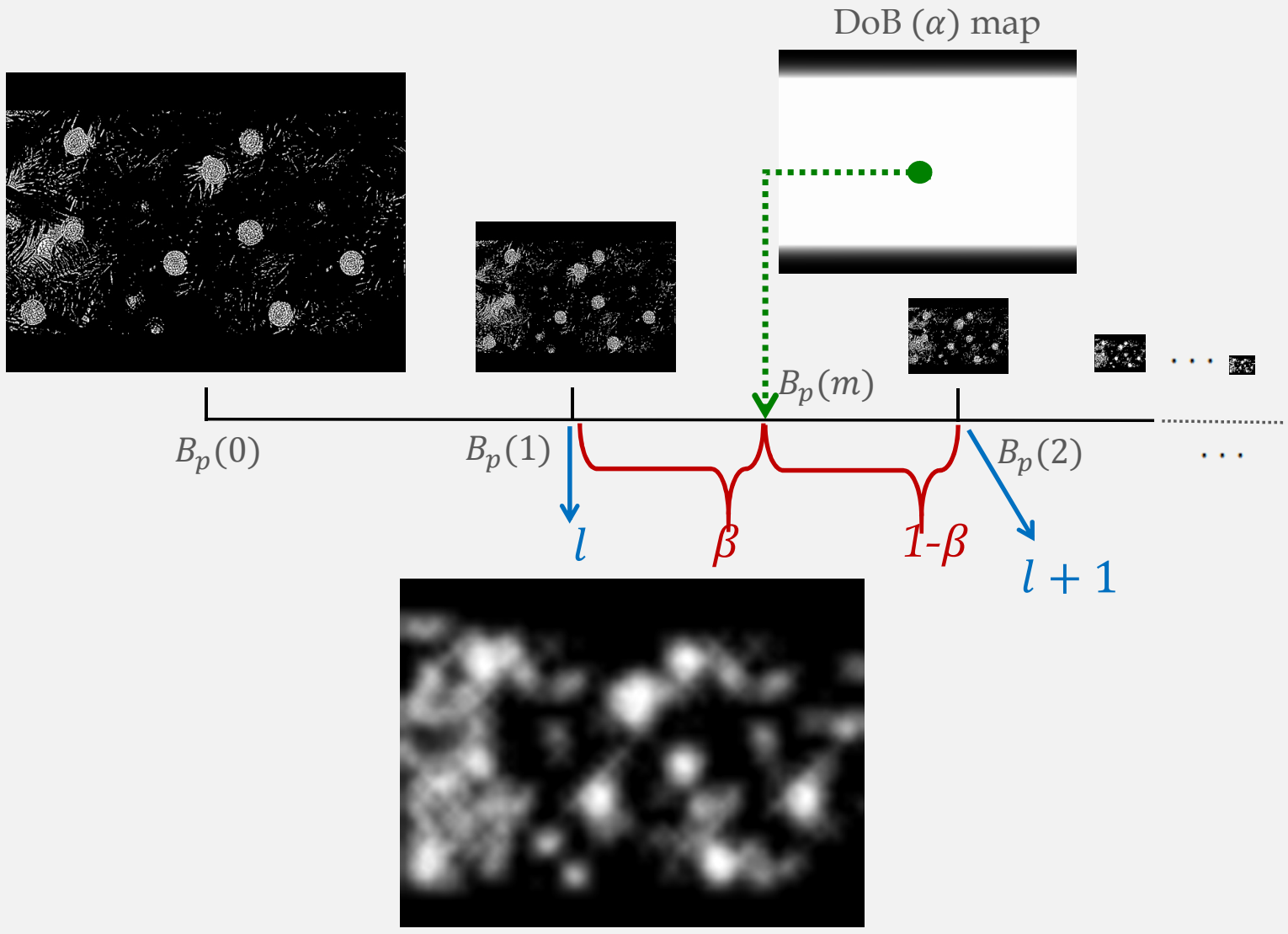
$$B_p(l) = \begin{cases} 1 & \text{if } I_p(l) > \omega_1 \text{ and } \sum_{q \in \Omega} \lambda\left(I_p(l), I_q(l)\right) \geq \omega_2 \\ 0 & \text{otherwise} \end{cases}$$

$$\lambda\left(I_p(l), I_q(l)\right) = \begin{cases} 1 & \text{if } I_p(l) > I_q(l) \\ 0 & \text{otherwise} \end{cases}, \qquad \omega_1 = 128,\ \omega_2 = 4$$



$$l = \lfloor m \rfloor, \quad \beta = m - l$$

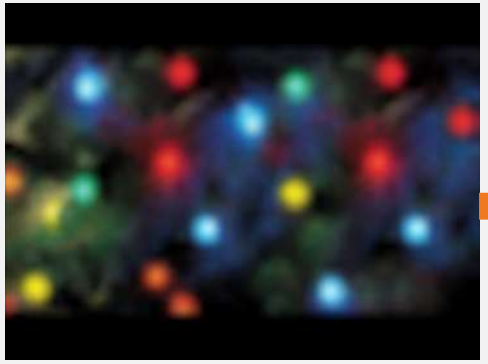
$$B_p(m) = B_p(l) \cdot (1 - \beta) + B_p(l + 1) \cdot \beta$$



$B_p(m)$ Positions of Bokeh map

Step 3: DOF Rendering with Bokeh Effect

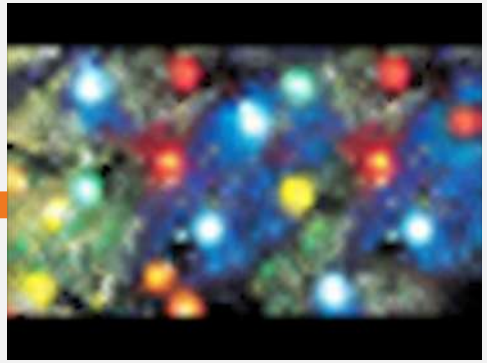
$NB(p)$



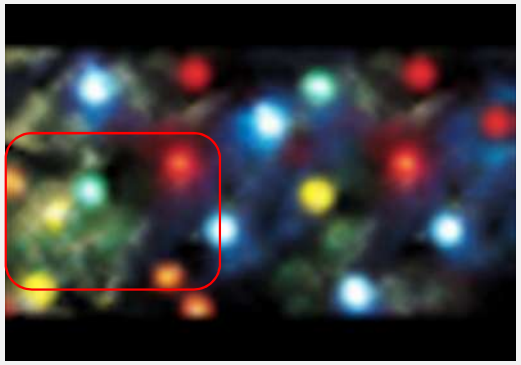
Rendering without intensity preserving.

$B_p(m)$

$IP(p)$



The intensity preserved image.



Final result.

$$I^{bokeh}(p) = NB(p)^{(1-\alpha)} \times IP(p)^{\alpha}$$

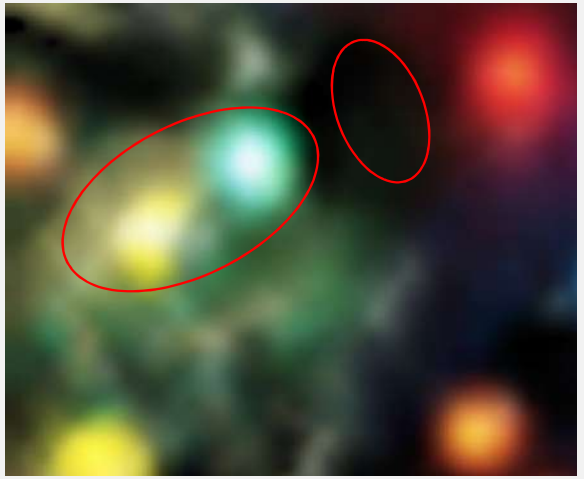
$$\alpha = B_p(m),\ 0 \leq \alpha \leq 1,$$

Result

Original image

Lee's method

Intensity preserved map



Our method

High Dynamic Range Imaging (HDRI)

Parts of the slides are from the class Digital Visual Effects, National Taiwan University, Prof. Y. Y. Chuang

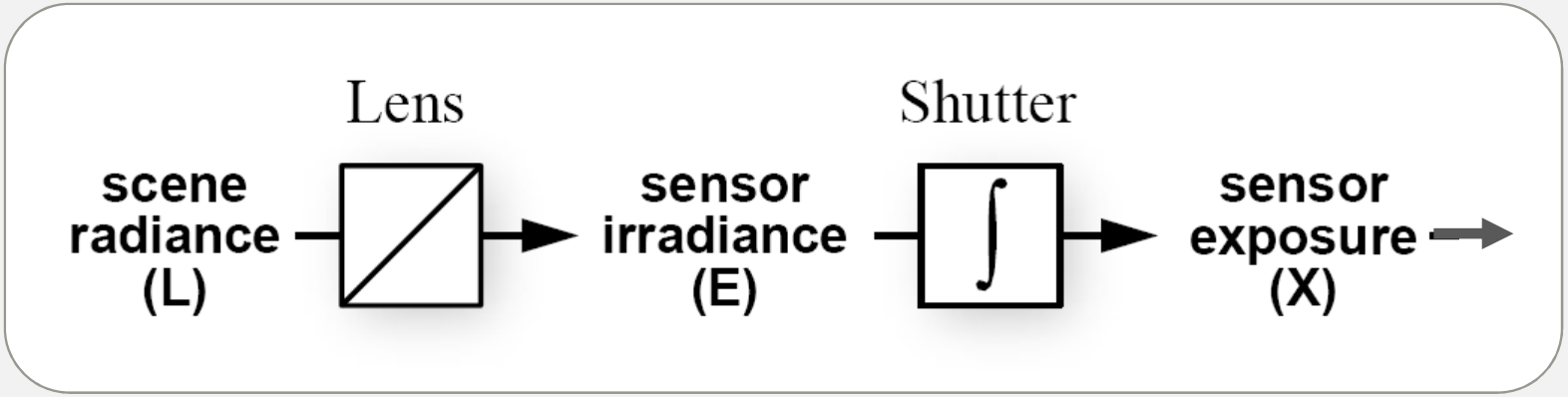
http://www.csie.ntu.edu.tw/~cyy/courses/vfx/15spring/lectures/index.html#hdr

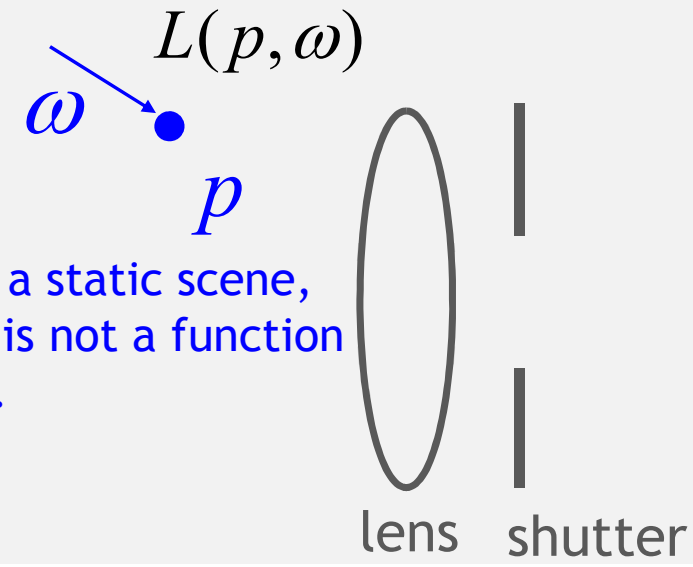
The world is high dynamic range

Camera Sensing

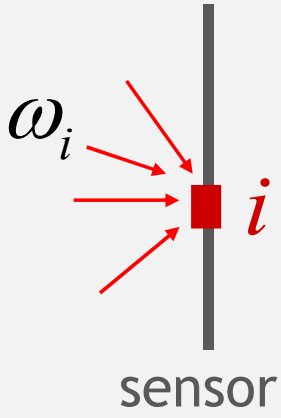
- Camera is an imperfect device for measuring the radiance distribution of a scene because it cannot capture the full spectral content and dynamic range.
- Limitations in sensor design prevent cameras from capturing all information passed by lens.

Camera Pipeline





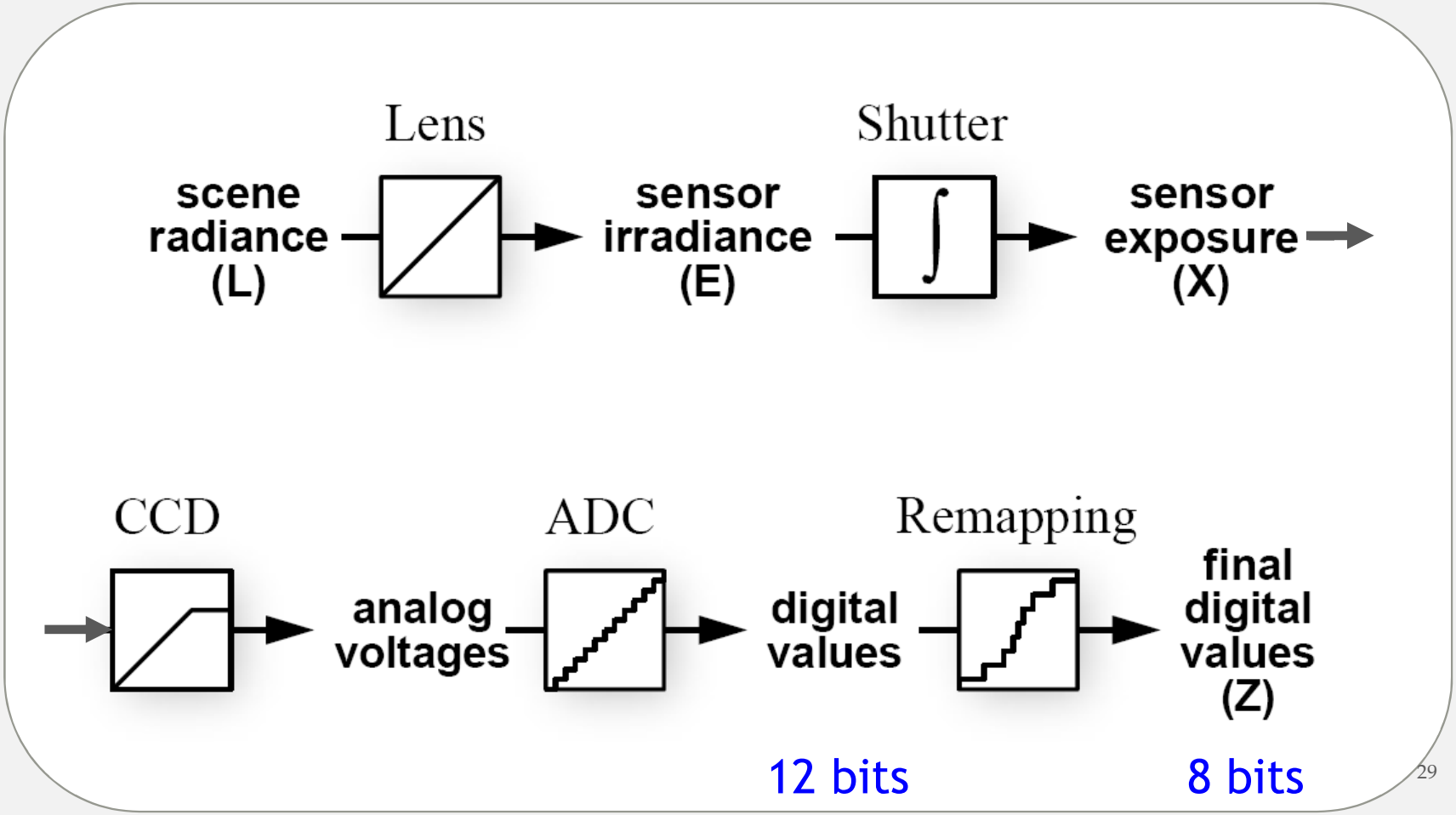
Assume a static scene, Thus, L is not a function of time.



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

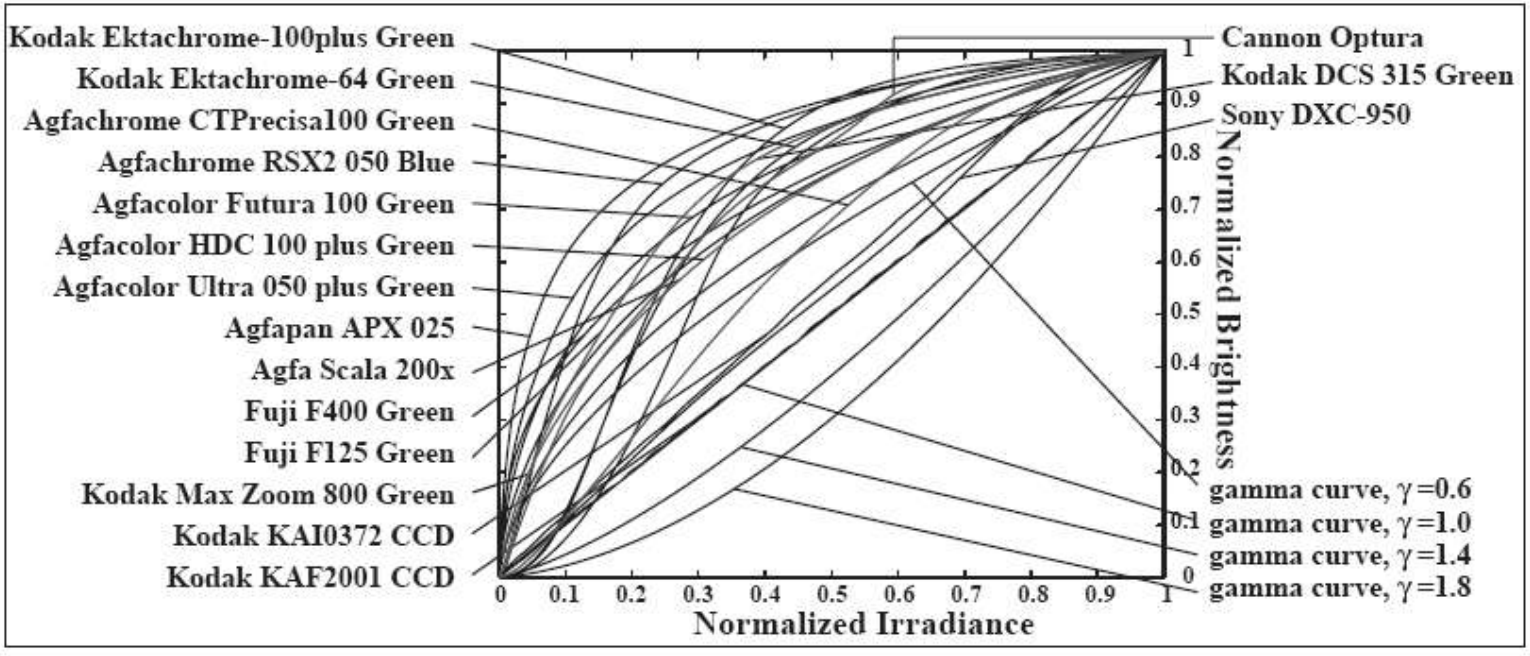
$$X_i = \int_{t=0}^{\Delta t} E_i\,dt = E_i \Delta t$$

Camera Pipeline (Cont.)



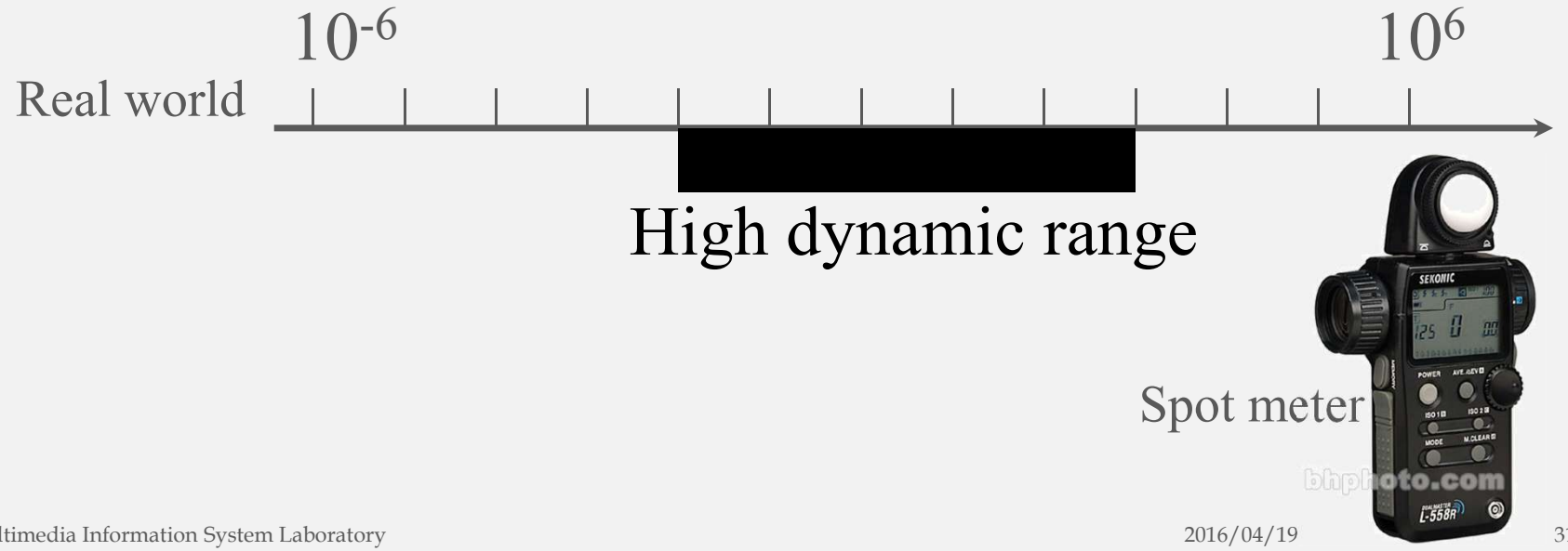
Real-world Response Functions

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

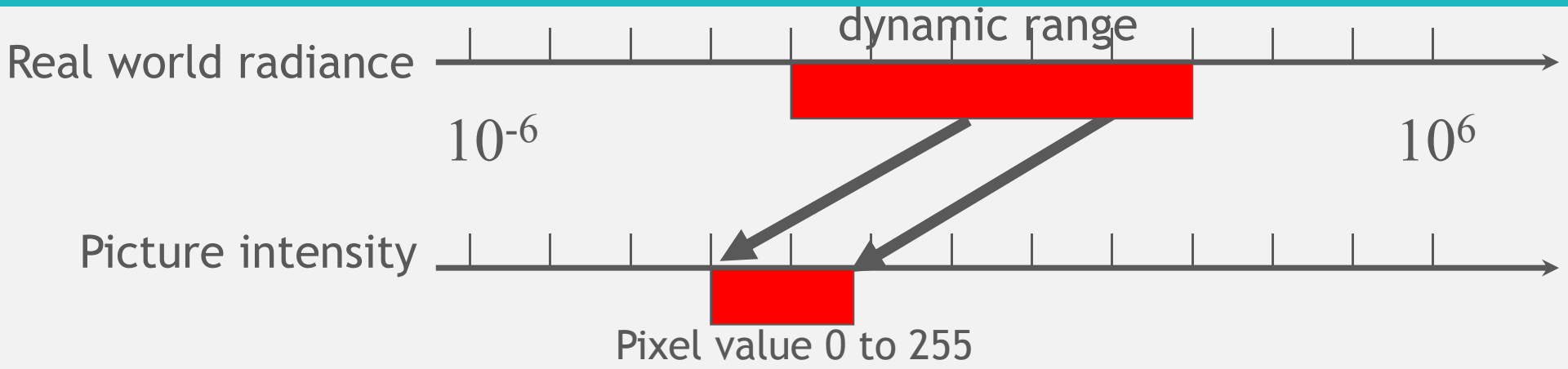


Real World Dynamic Range

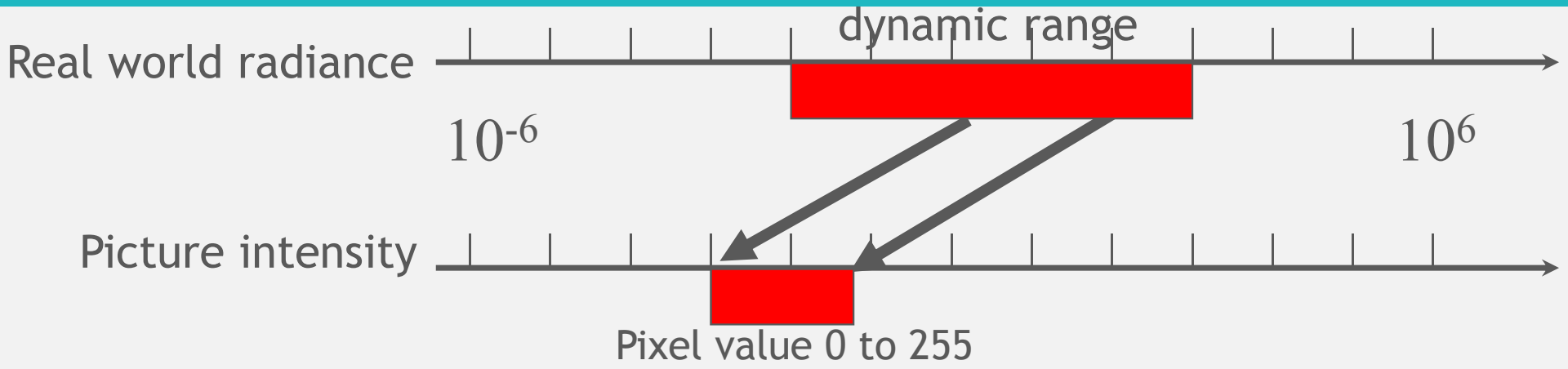
- Eye can adapt from ~ 10^{-6} to 10^{6} cd/m^2
- Often 1 ： 100,000 in a scene
- Typical 1:50, max 1:500 for pictures



Short Exposure

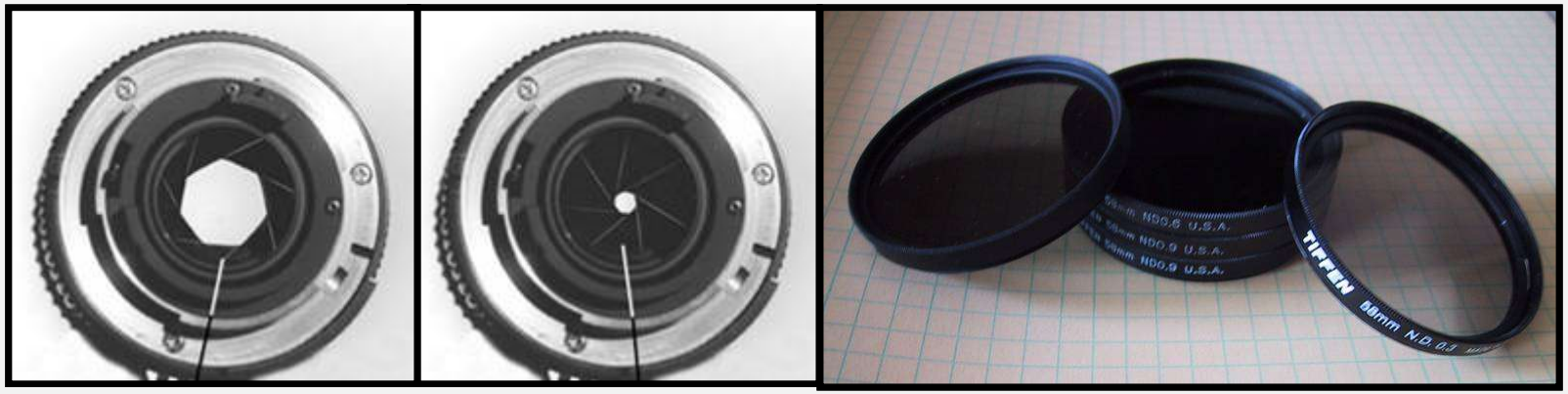


Long Exposure



Solution

- Varying exposure:
  - Recover response curve from multiple exposures, then reconstruct the ***radiance map***
- Ways to change exposure
  - Shutter speed
  - Aperture
  - Neutral density filters



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

Varying Shutter speed

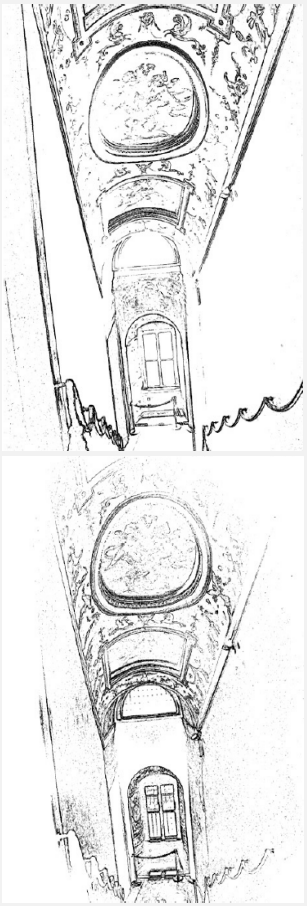
- Shutter speed usually obey a power series – each "stop" is a factor of 2
  - Theoretically is: ¼, 1/8, 1/15, 1/30, 1/60, 1/125, 1/250, 1/500, 1/1000 sec
  - Practically is: ¼, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512, 1/1024 sec

Image Alignment

- The inputs are N grayscale images
  - You can either use the green channel or convert into grayscale by Y=(54R+183G+19B)/256)
- Median Threshold Bitmap (MTB) alignment technique
  - a binary image formed by thresholding the input image using the median of intensities
  - consider only integer pixel offset (empirically enough)
  - fast and easy-to-implement

Original Image

MTB



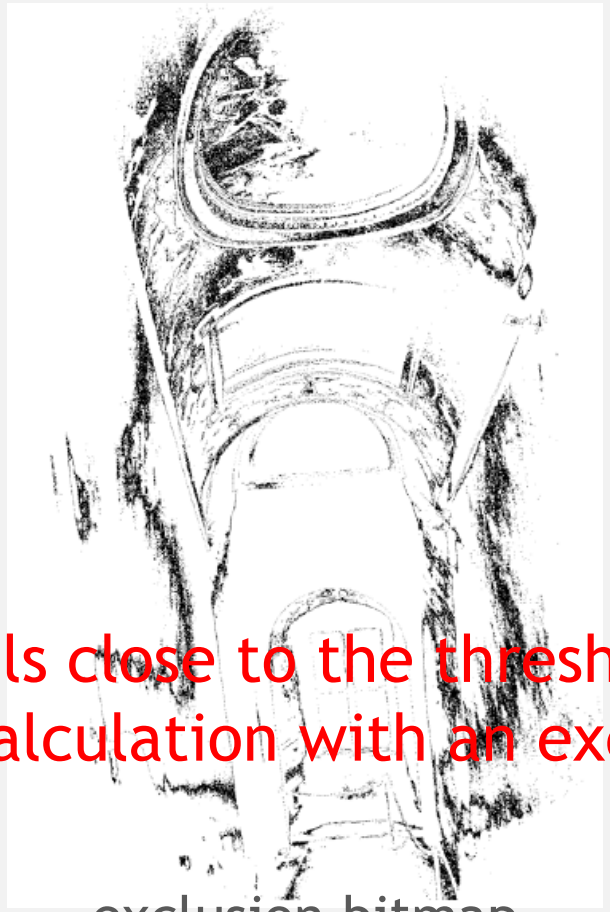
Gradient Image

Search for the Optimal Offset (minimum difference)

- Try all possible offsets (x, y)
  - High complexity
- Gradient descent
  - Compute the local bitmaps differences between the starting offset (0,0) and the nearest minimum
- Multiscale technique
  - Construct the image pyramids with $\log_2$(max_offset) levels
  - Start from the top level. Try 9 nighbors and pass the opt_offset*2 to the next level
  - As fast as gradient descent in most cases, but is more likely to find the global minimum within the allowed offset range

image pyramids

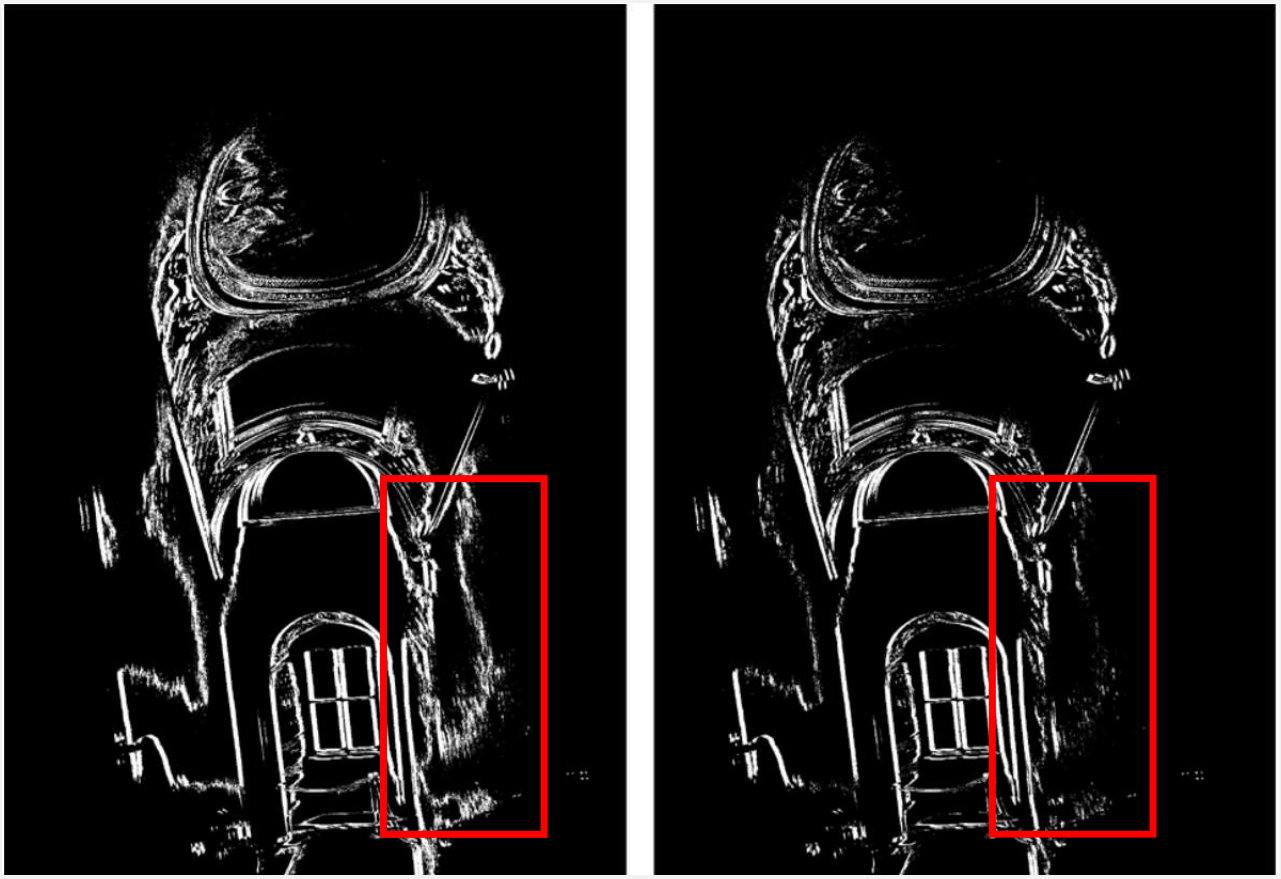
Threshold Noise



exclusion bitmap

Exclude pixels close to the threshold from our difference calculation with an exclusion bitmap.

Result of Threshold Noise

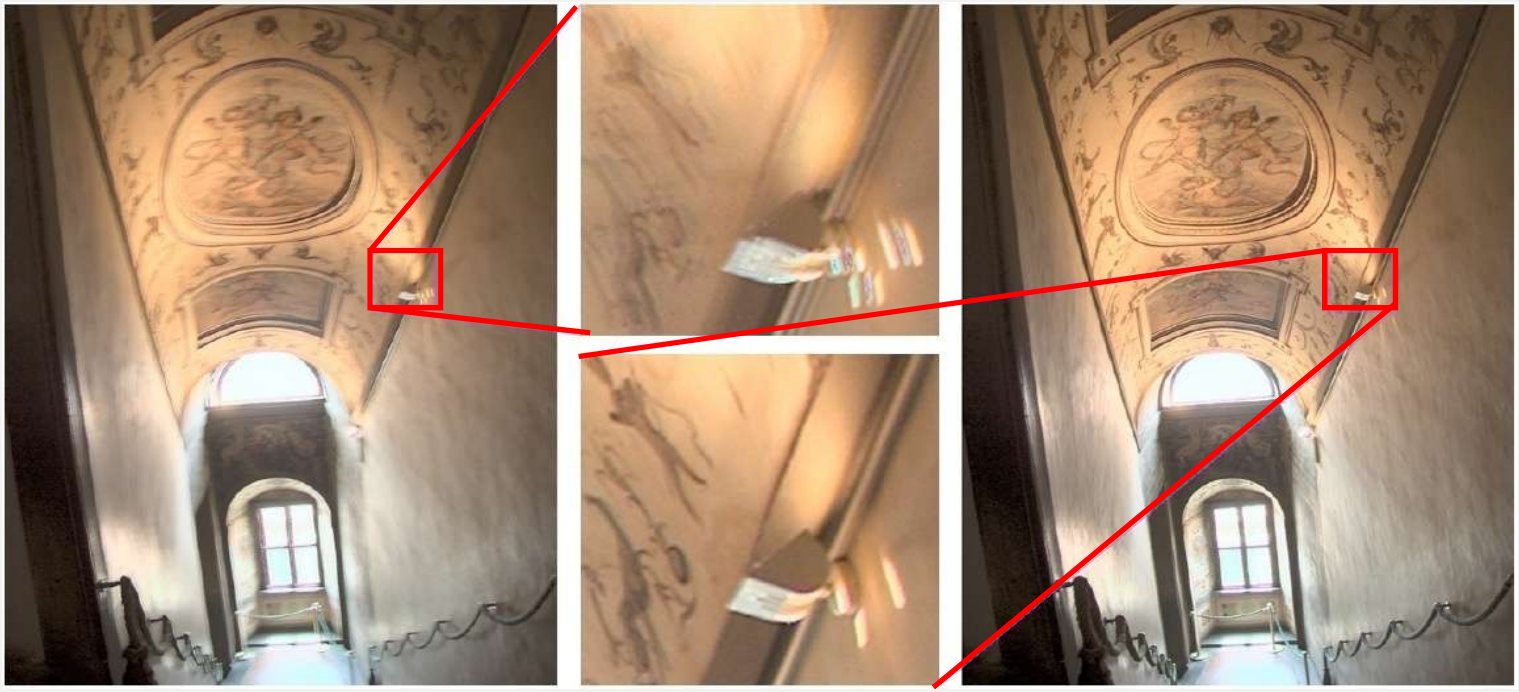


Efficiency considerations

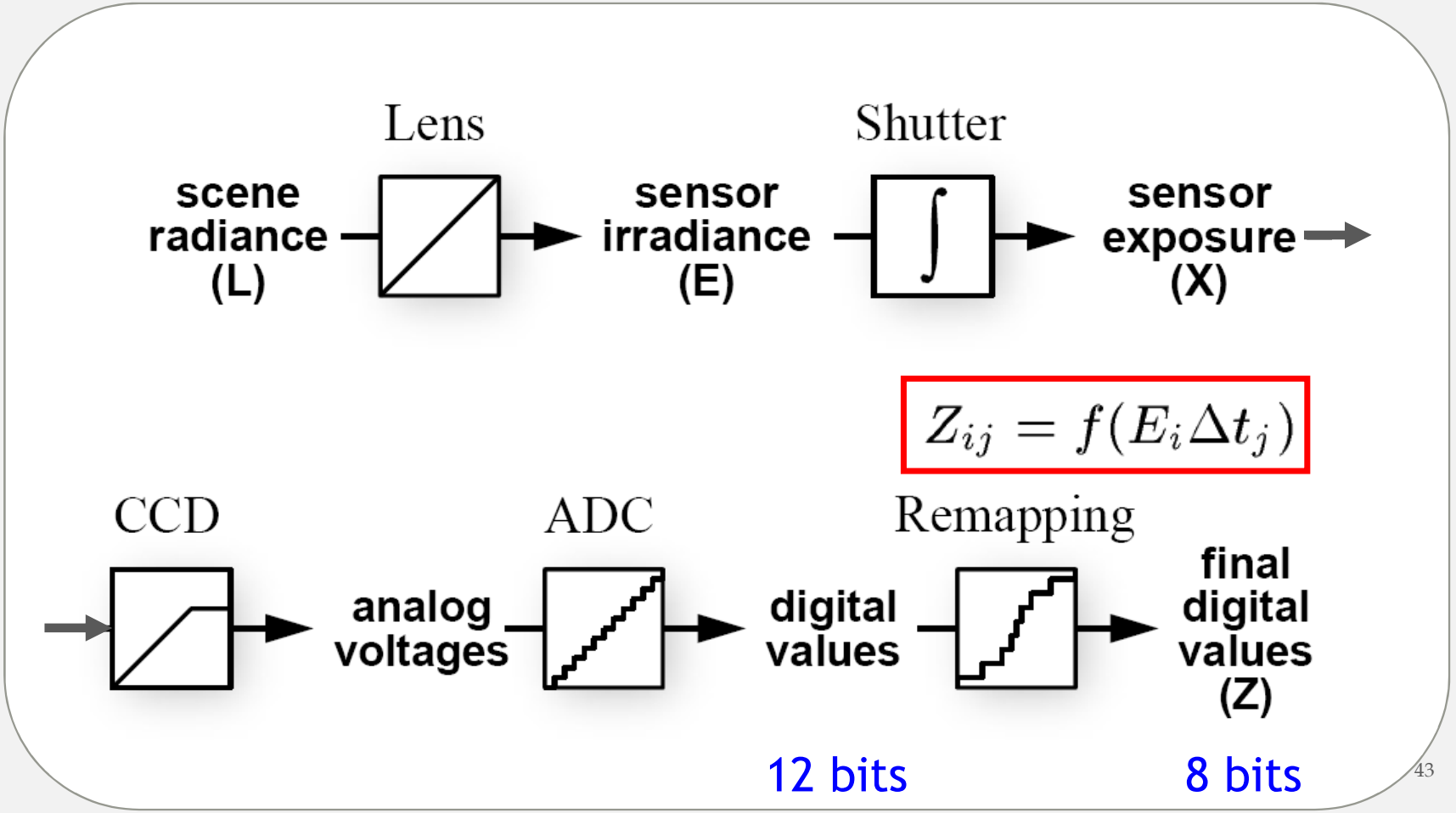
- XOR for taking difference
- AND with exclusion maps
- Bit counting by table lookup

Alignment Results

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



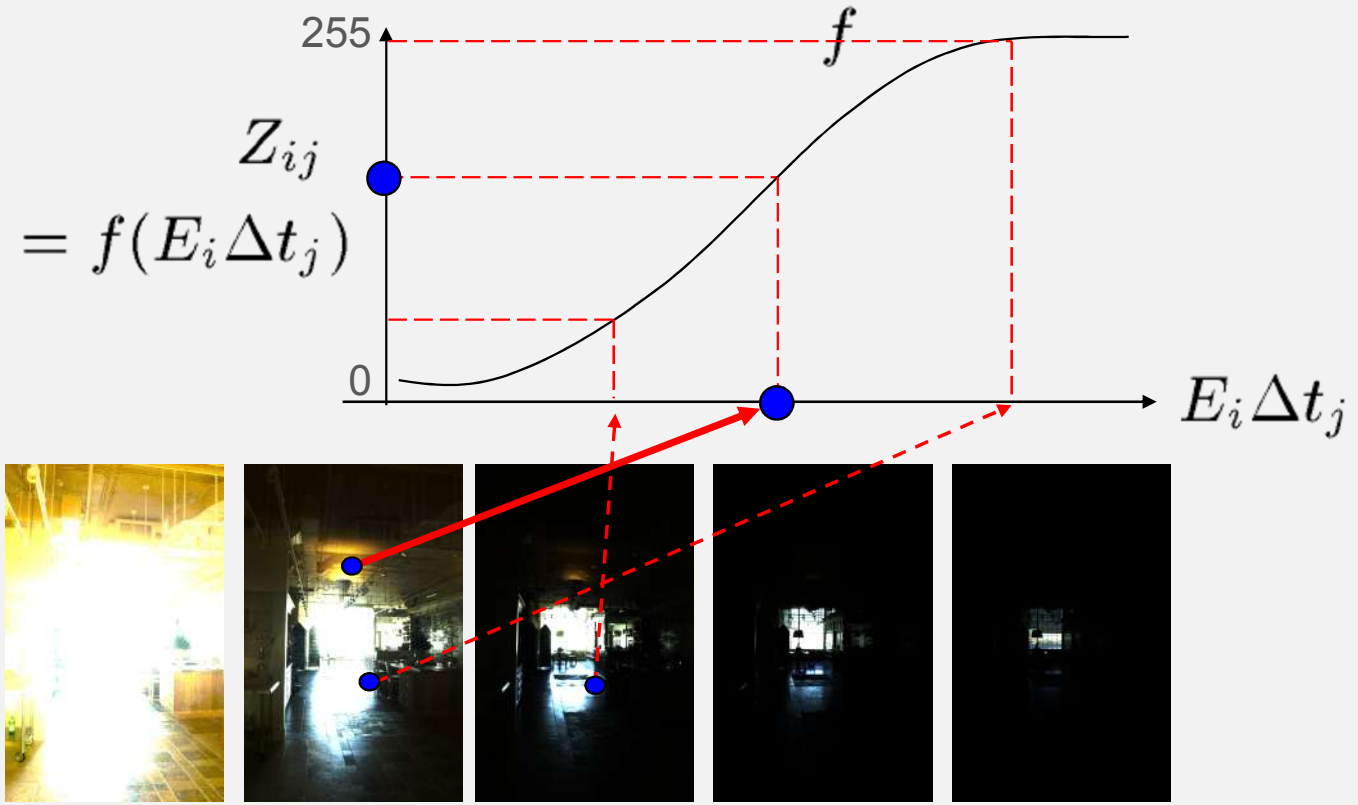
Camera Pipeline



Response Curve Recovering

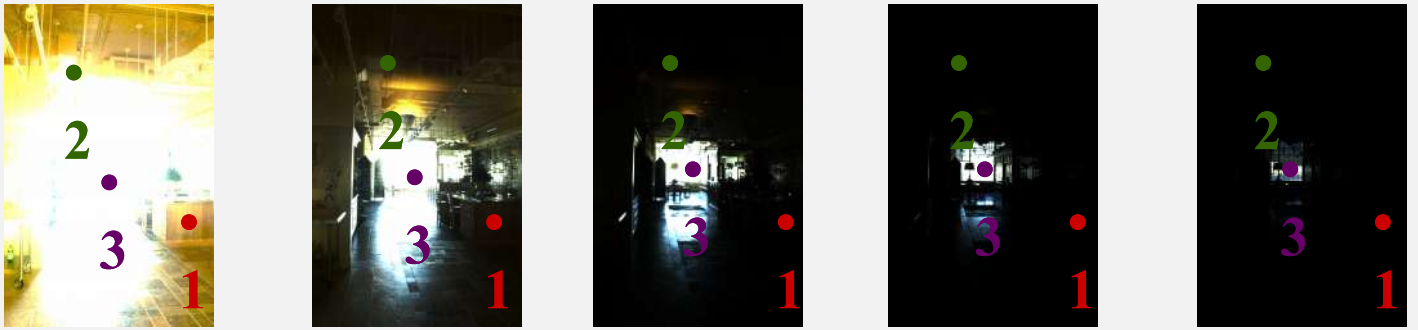
- We want to obtain the inverse of the response curve

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



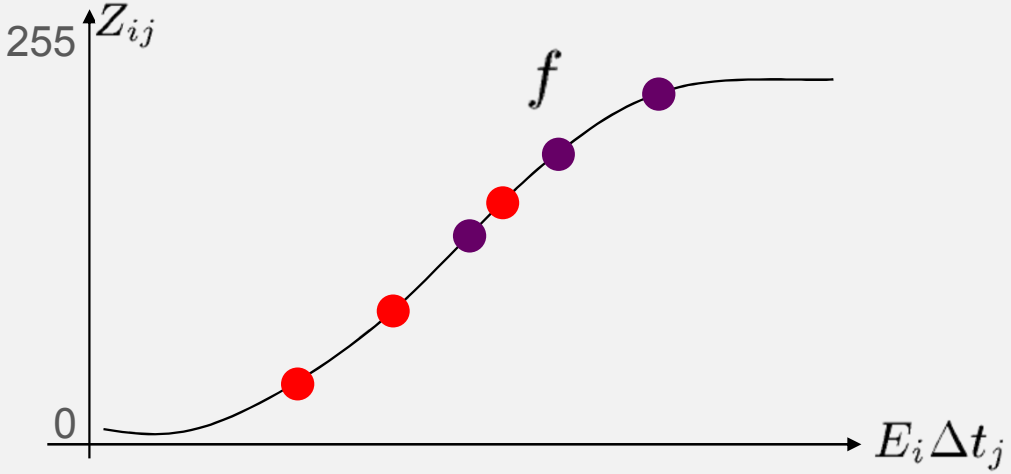
Response Curve Recovering (Cont.)

Image series



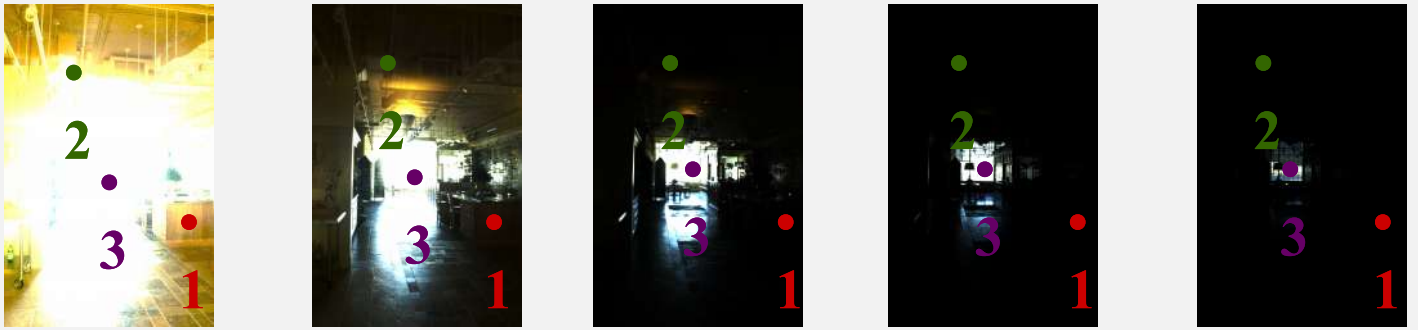
Δt =2 sec Δt =1 sec Δt =1/2 sec Δt =1/4 sec Δt =1/8 sec

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



Response Curve Recovering (Cont.)

Image series



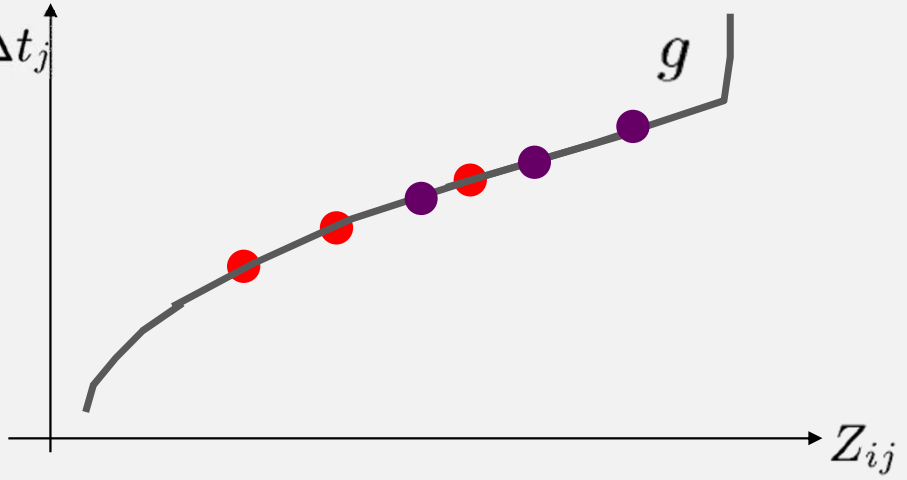
Δt =2 sec Δt =1 sec Δt =1/2 sec Δt =1/4 sec Δt =1/8 sec

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

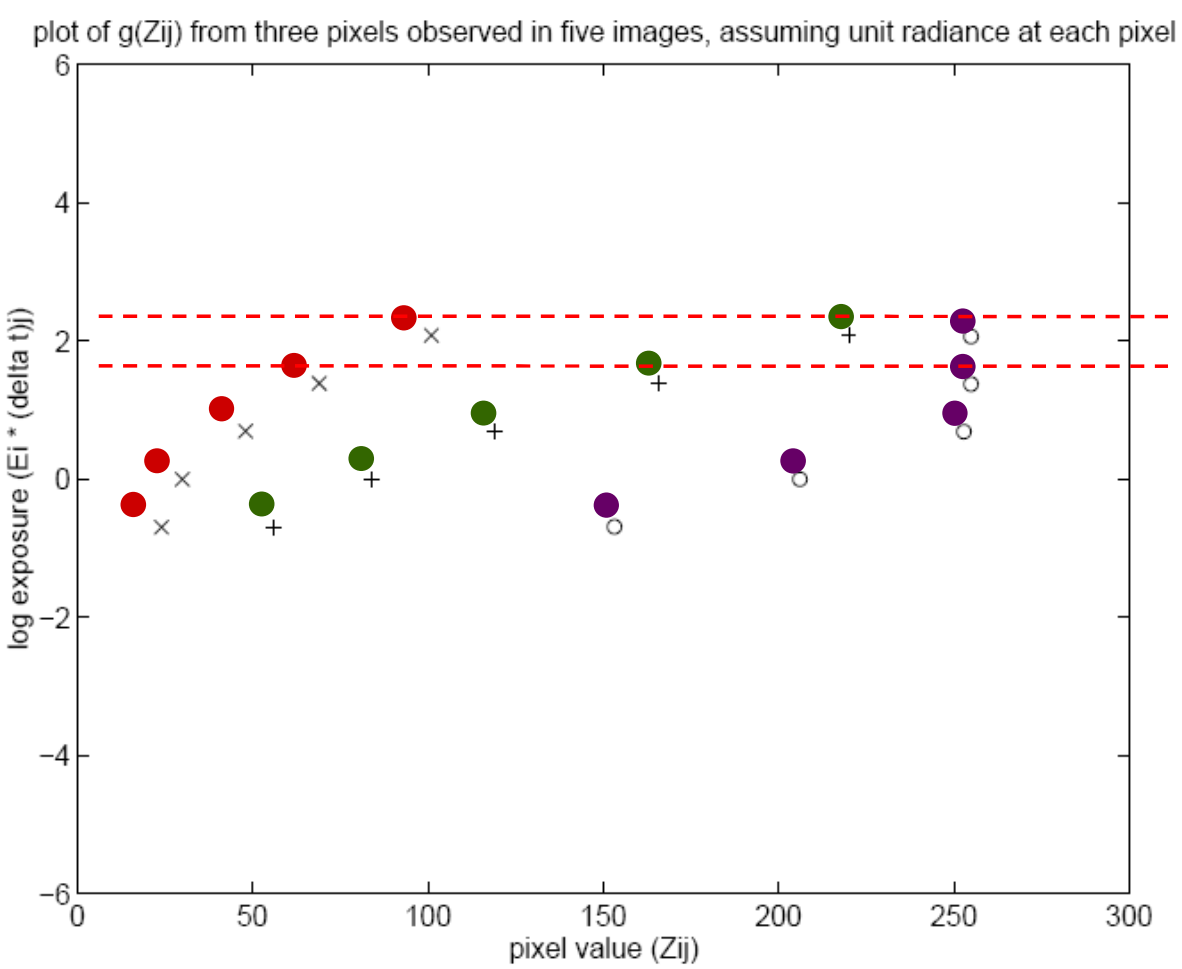
$$X_{ij} = E_i \Delta t_j$$

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

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

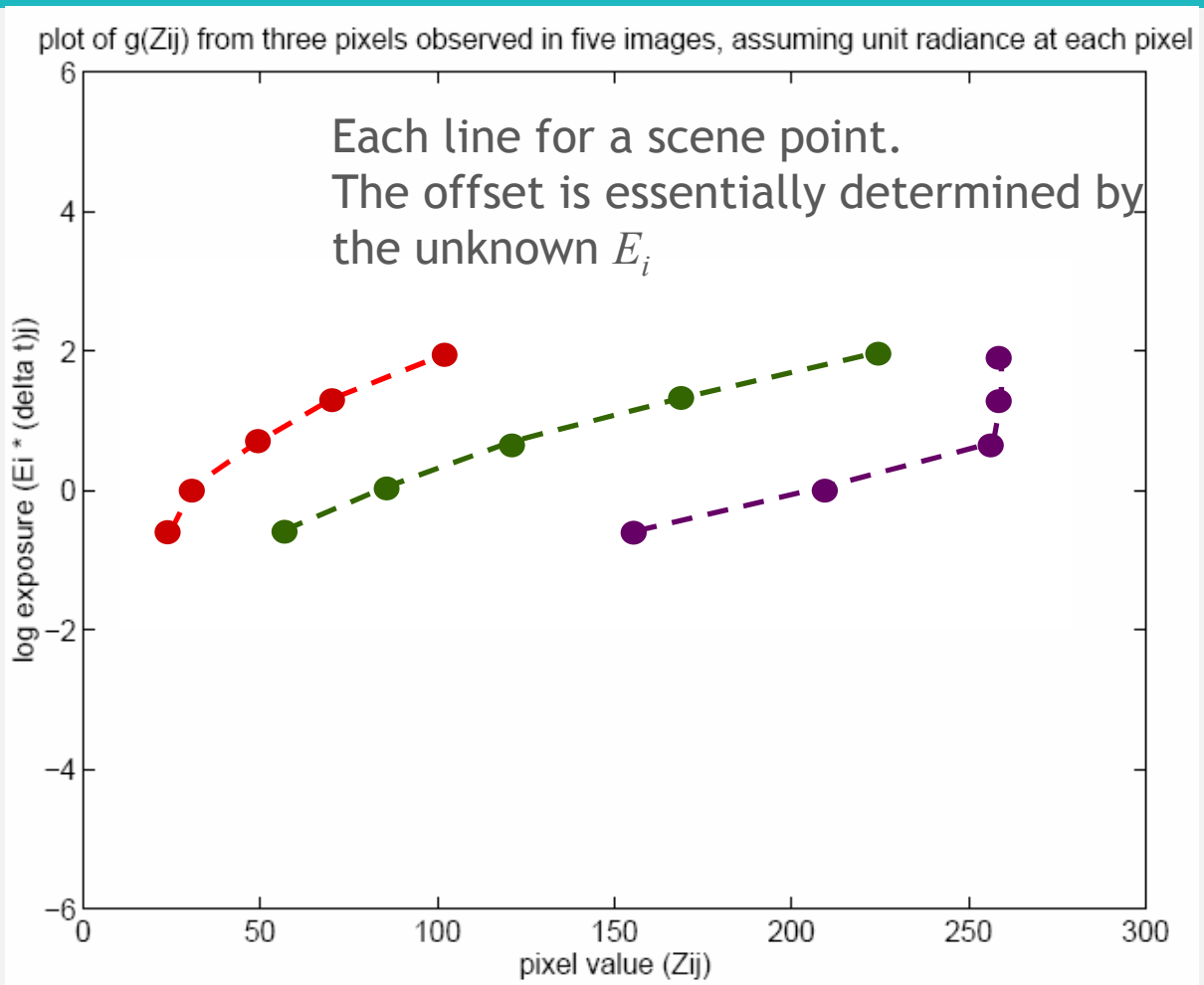


Idea Behind the Math

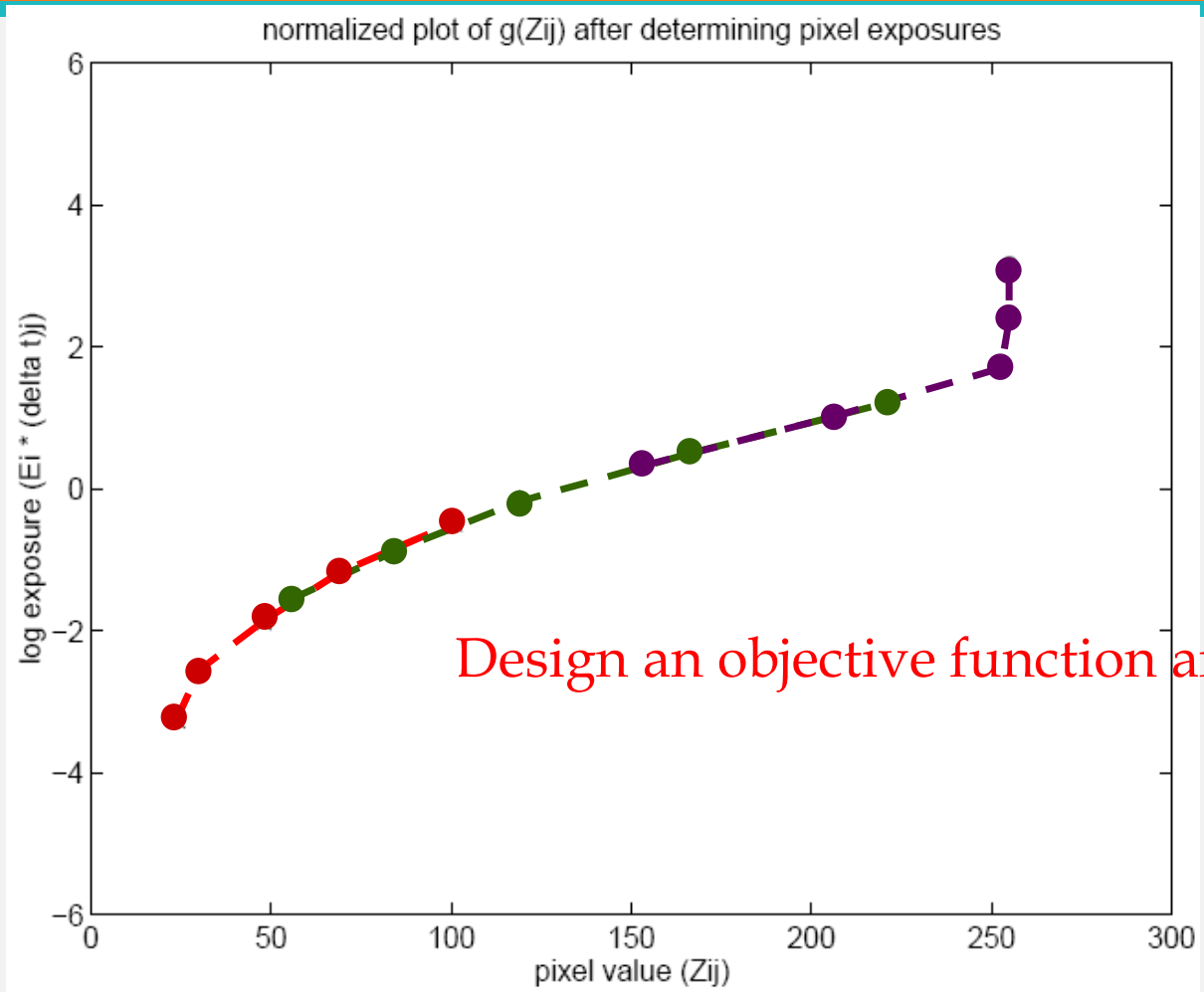


ln2

Idea Behind the Math



Idea Behind the Math



Design an objective function and optimize it

Math for Recovering Response Curve

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

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

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

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

Math for Recovering Response Curve (Cont.)

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

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

- Add a hat weighting function

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

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

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.
- It is an overdetermined system of linear equations and can be solved using SVD

How to optimize?

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

1. Set partial derivatives to zero
2.

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

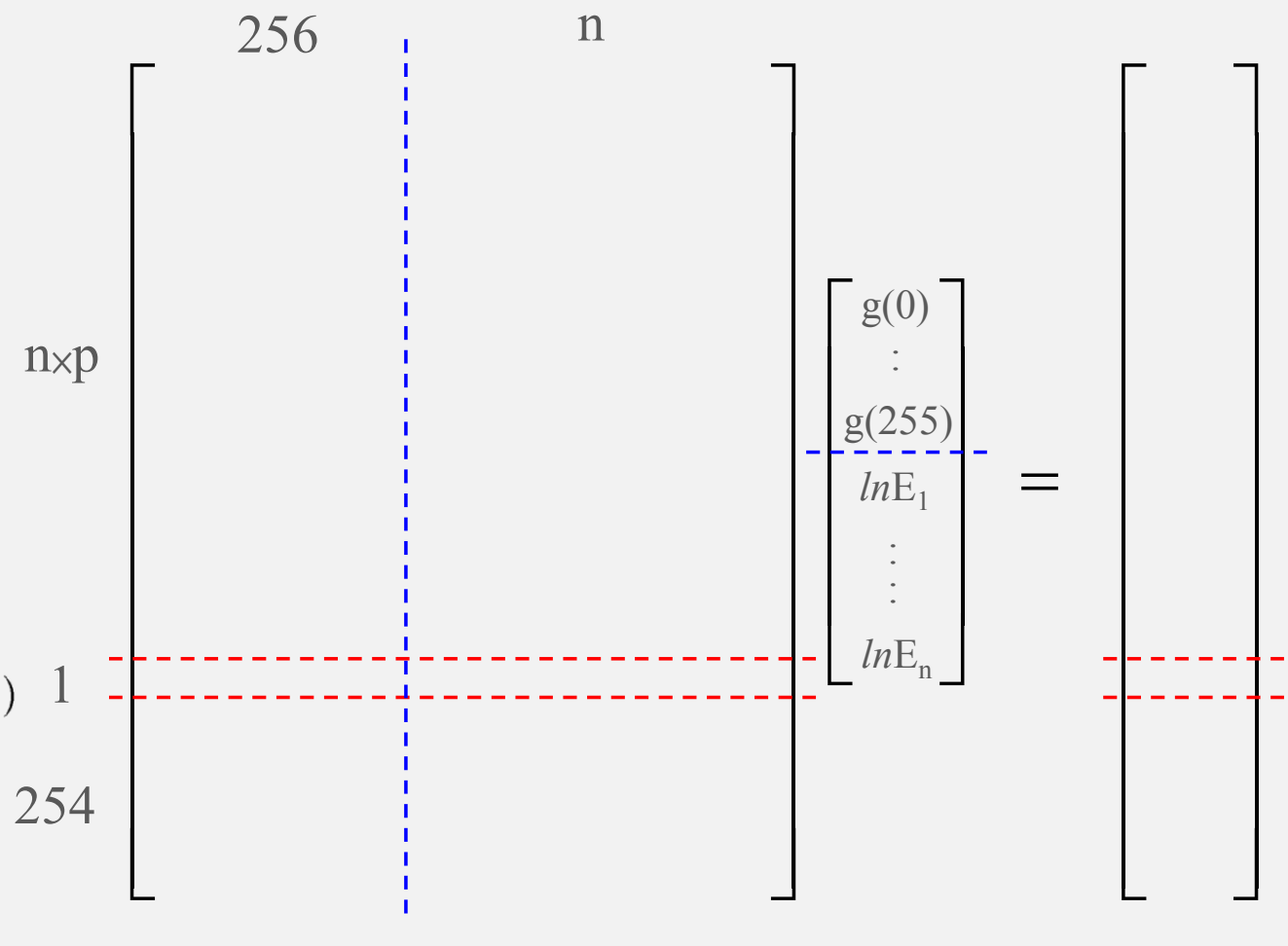
Sparse Linear System Ax=b

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

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

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

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



Least-square Solution for a Linear System

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

$m > n$

These m equations 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 **A** and rewrite it as $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}V^{\mathbf{T}}$

pseudo inverse

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

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

- Libraries for SVD
  - Matlab
  - GSL
  - Boost
  - LAPACK
  - ATLAS

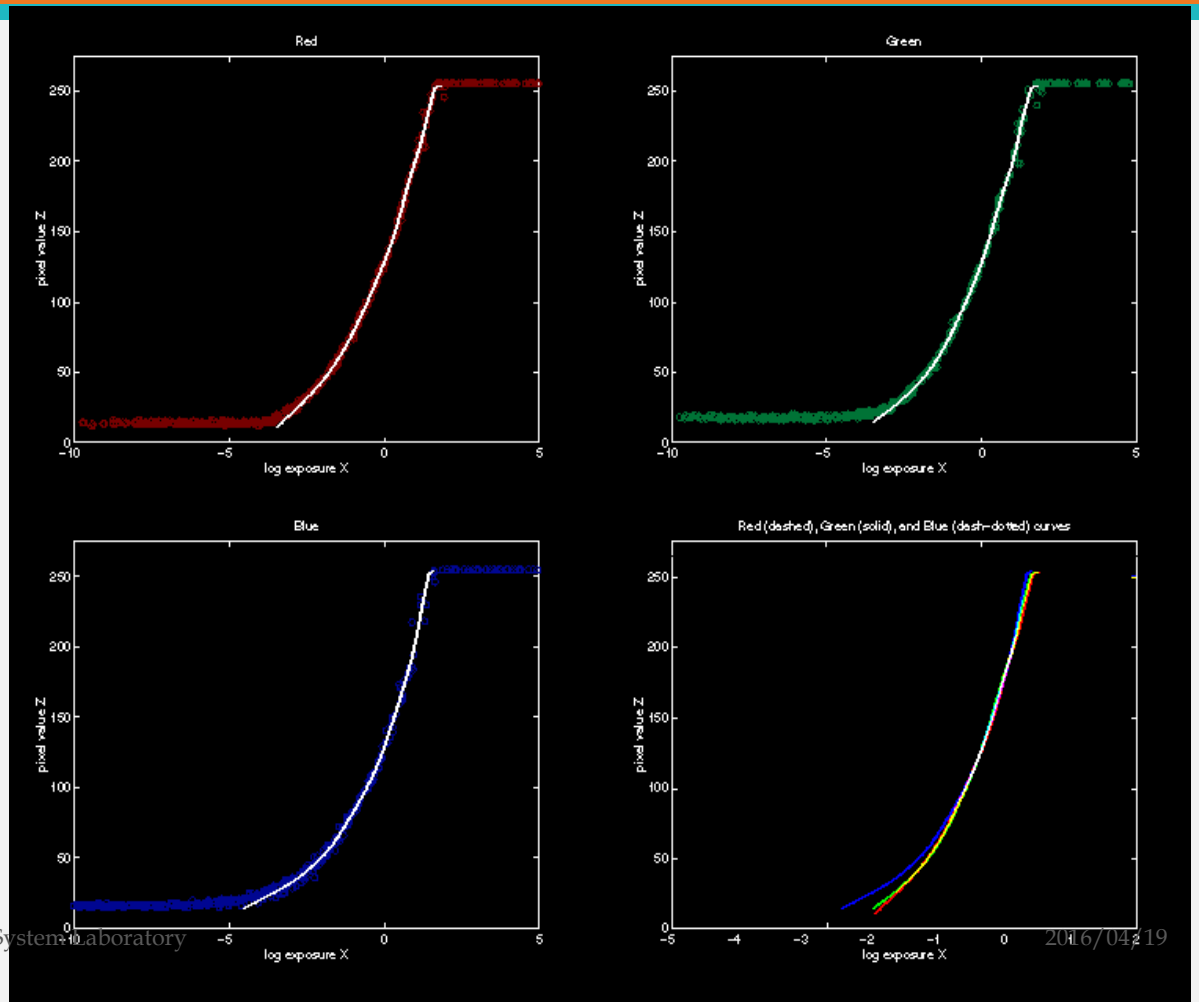
Matlab Code

```
%
% 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

```
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(i,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

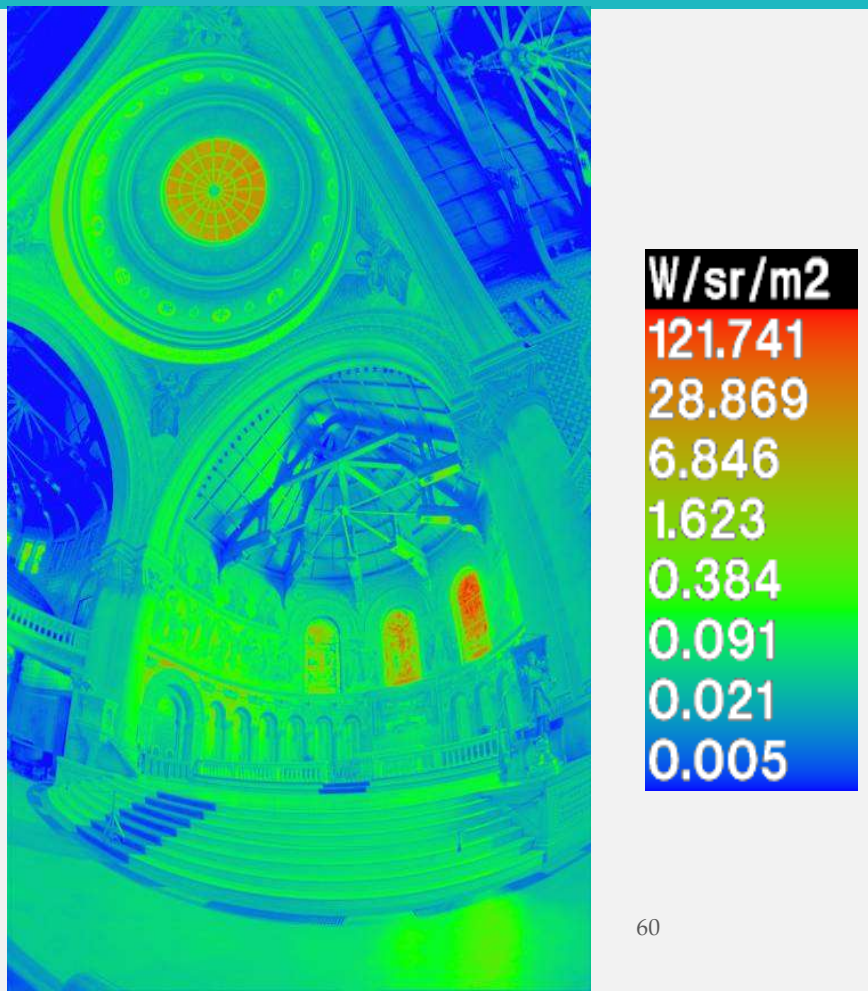


Constructing HDR Radiance Map

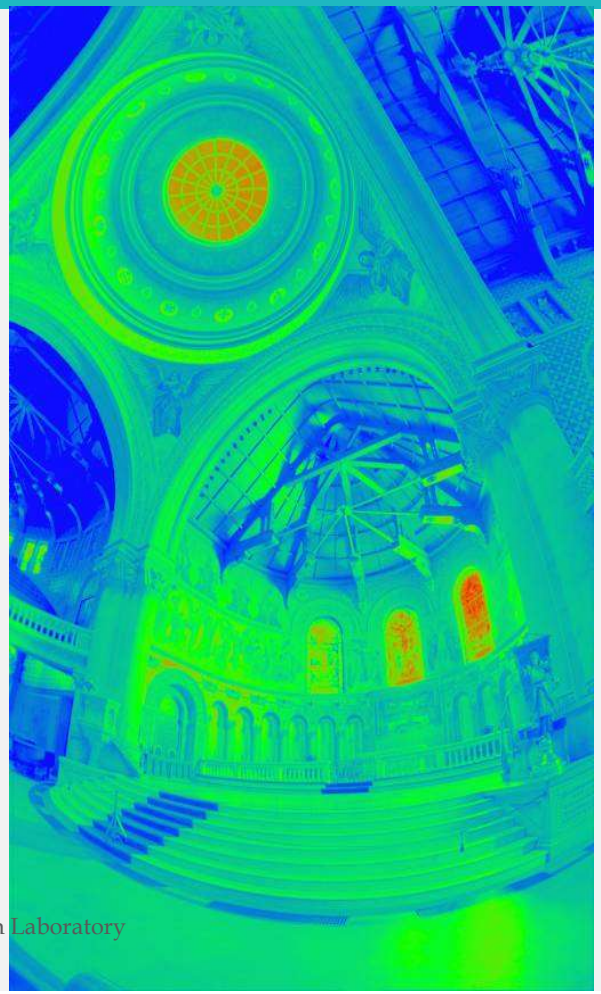
$$\ln E_i = g(Z_{ij}) - \ln \Delta t_j$$

combine pixels to reduce noise and obtain a more reliable estimation

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



What is this for?



- Human perception
- Vision/graphics applications

Automatic Ghost Removal

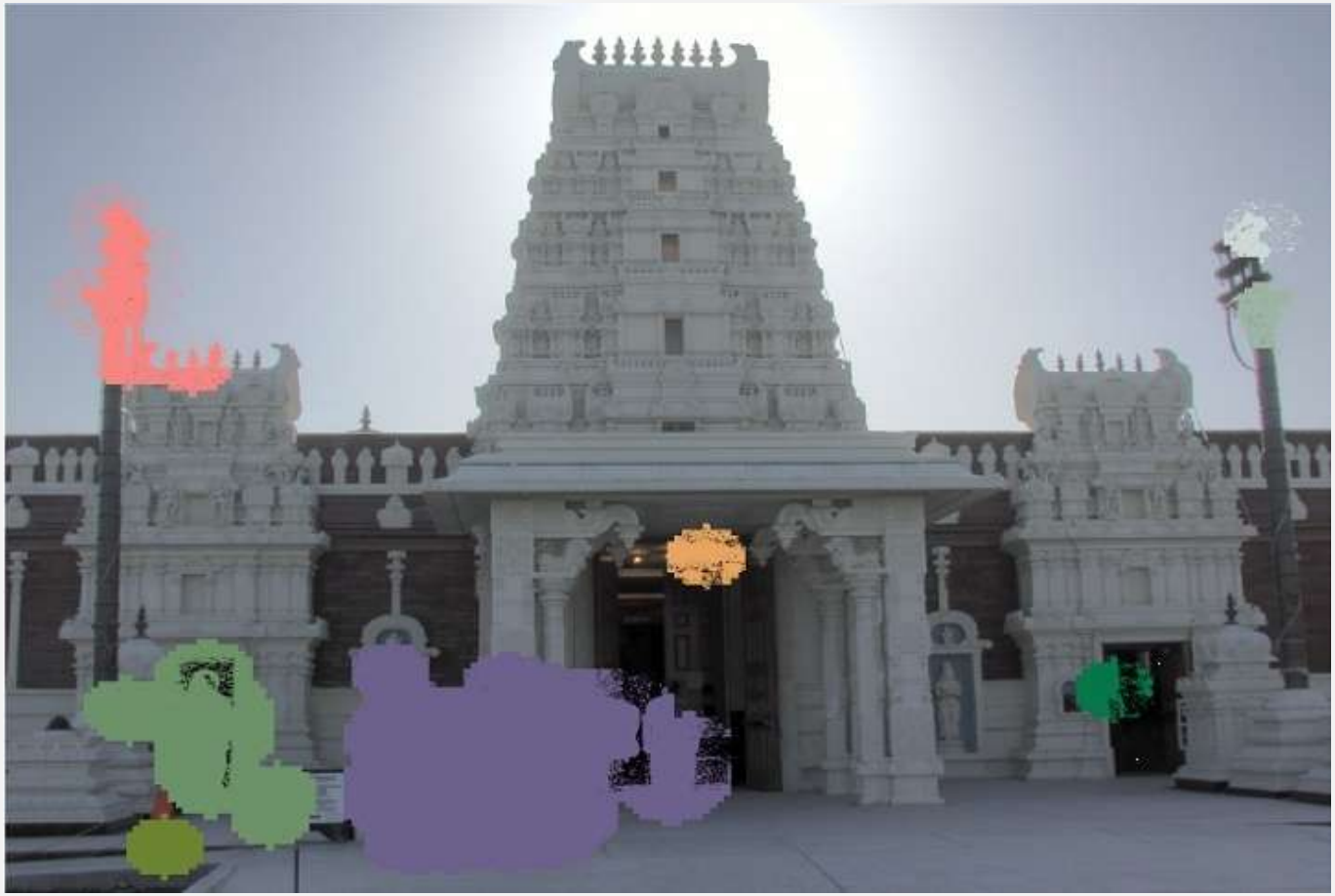
before

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

Lens Flare Removal

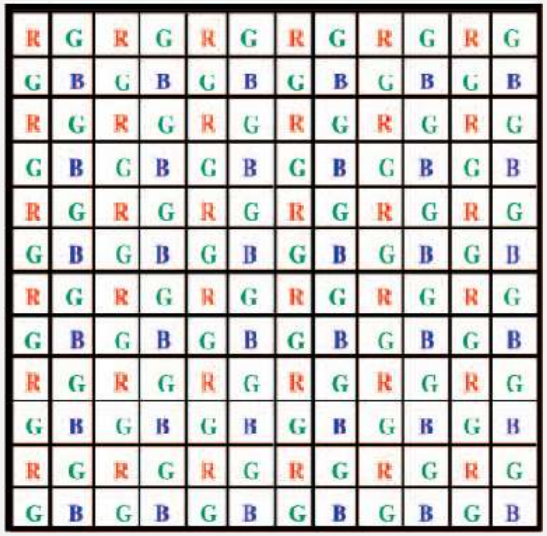
before

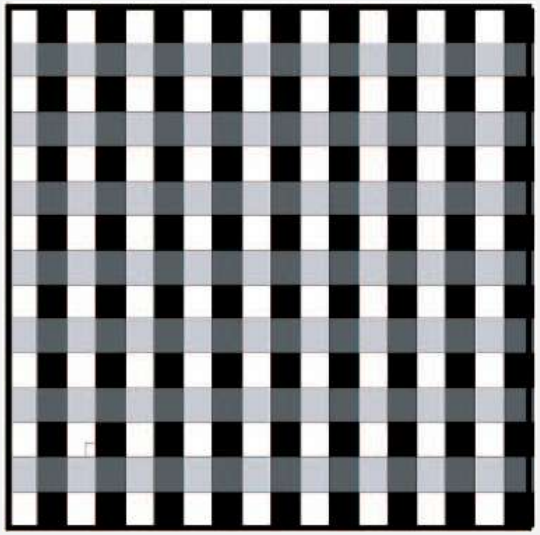
after

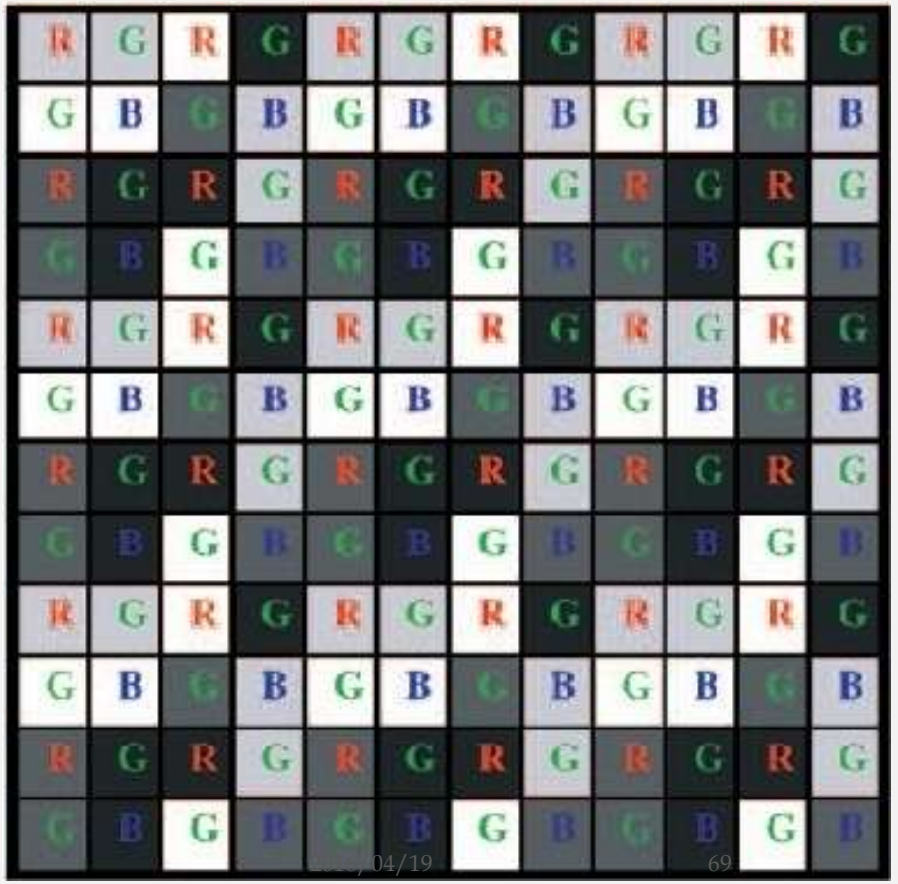
Patch-Based HDR

Robust to Camera/Scene Motion

Assorted Pixel







Assorted Pixel