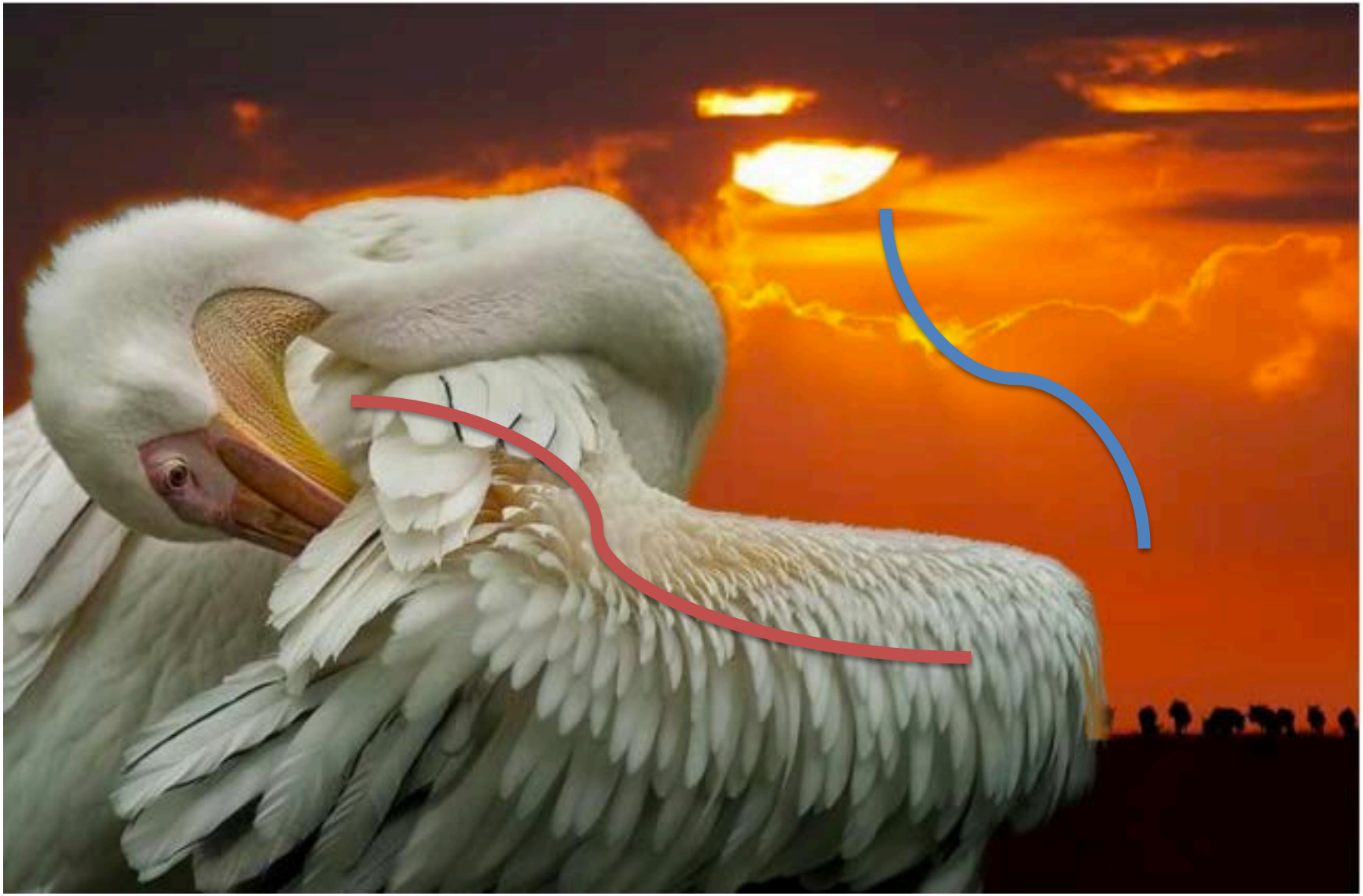
GradientShop:
A Gradient-Domain Optimization Framework for Image and Video Filtering

Pravin Bhat C. Lawrence Zitnick Michael Cohen Brian Curless

Presentation : Michael Tao
Discussion : Sean Arietta

Input Image

Input Image + few user inputs



Output Image: Relighting

Output Image: Recoloring

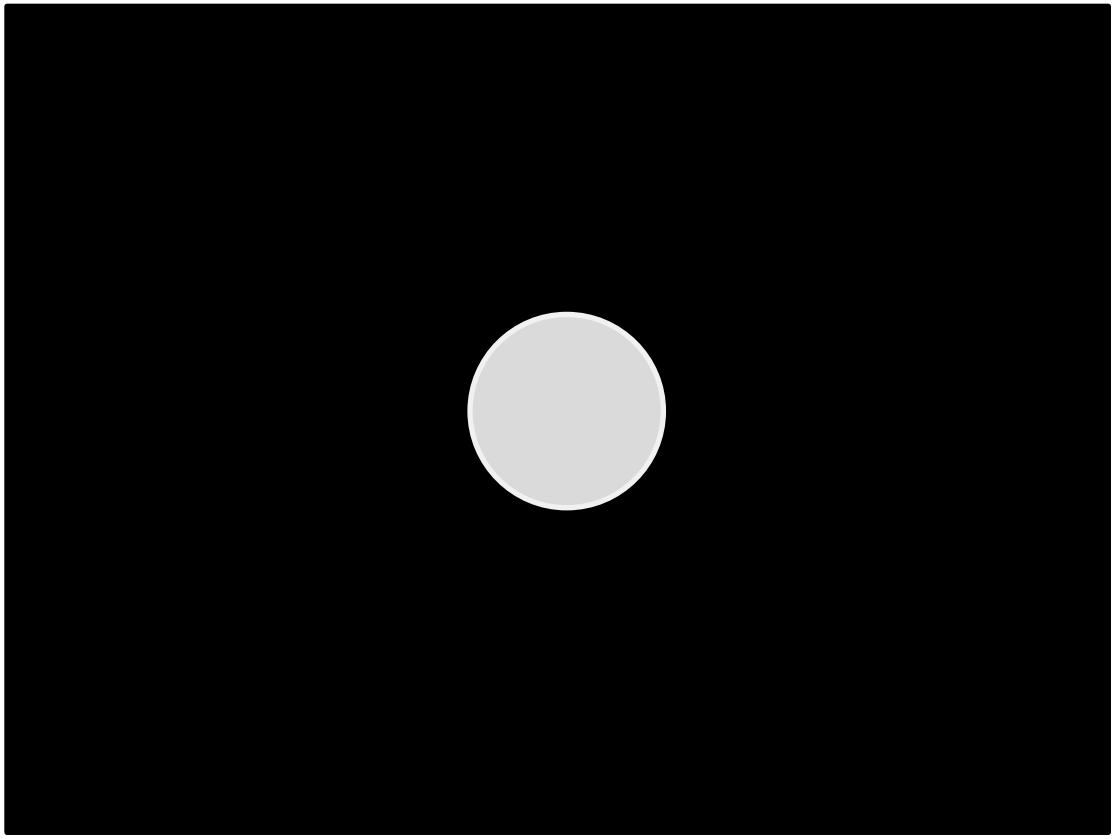
Output Image: Non-Photorealistic Effects

Agenda

- ➡ Motivation
- Goal and Algorithm
- Applications and Results
- Discussion

Motivation: Why Gradient Domain?

1. Great for human perception

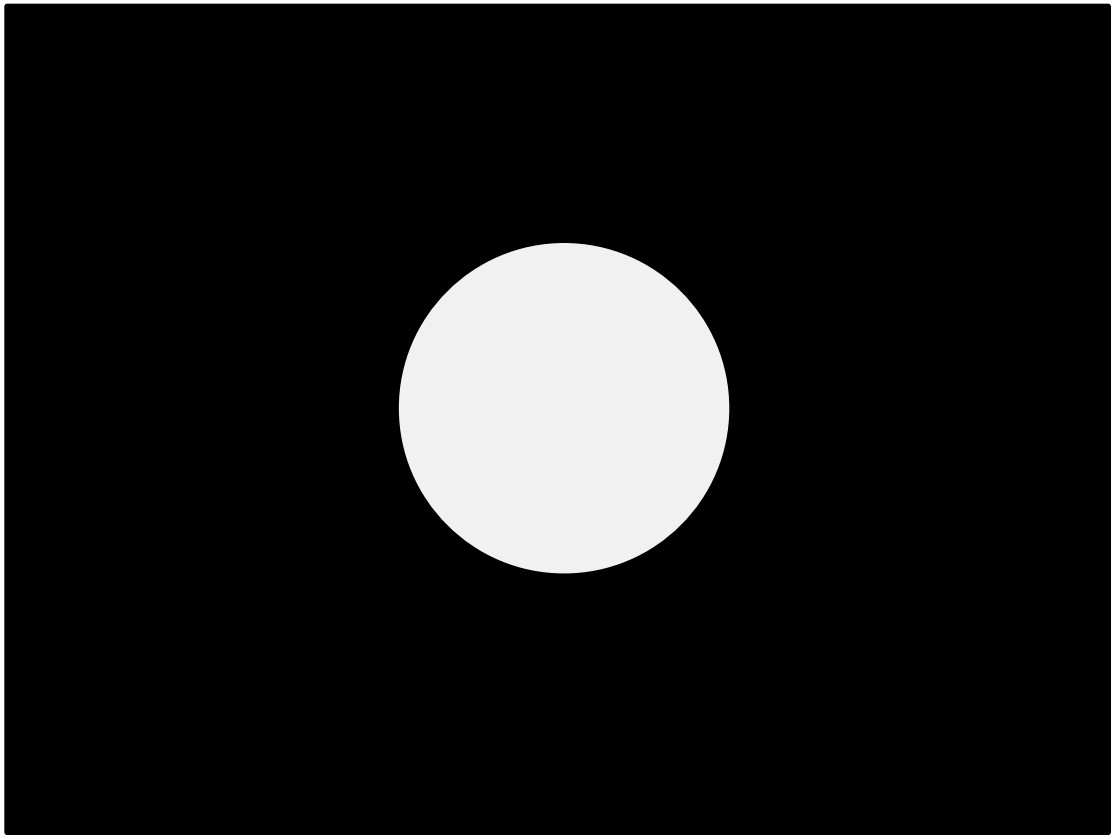


Motivation: Why Gradient Domain?

1. Great for human perception

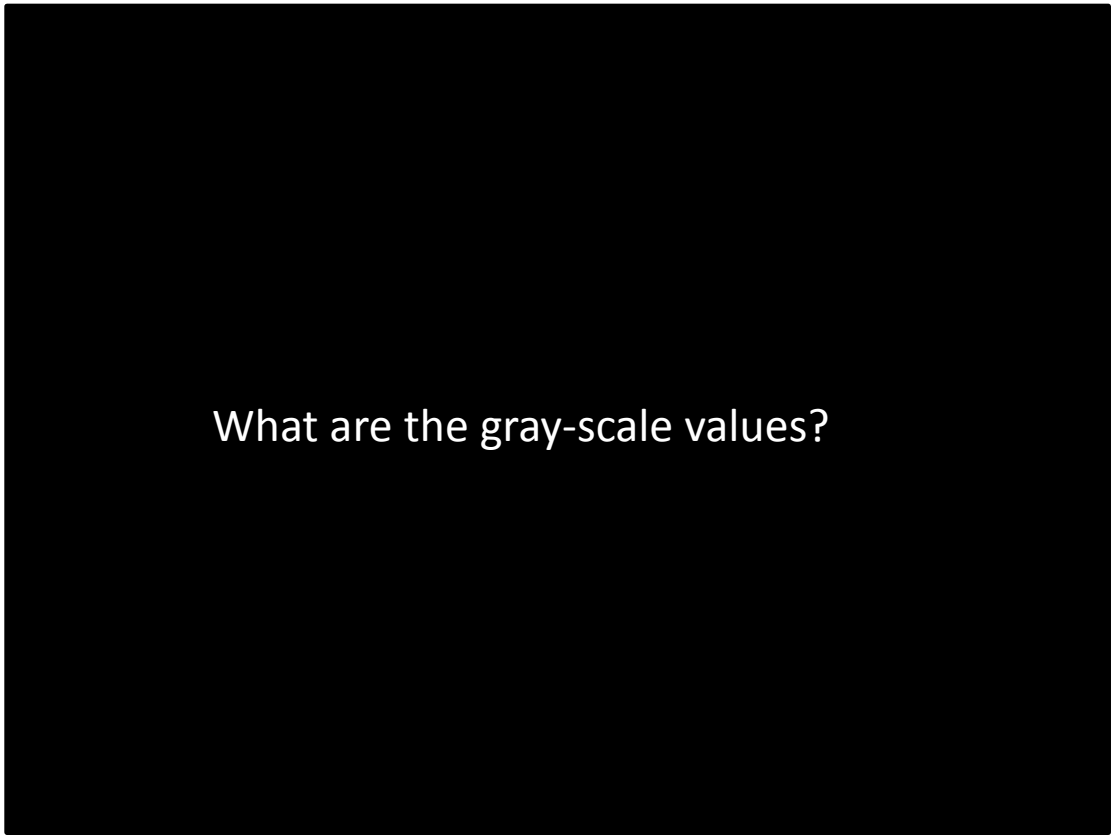
Motivation: Why Gradient Domain?

1. Great for human perception



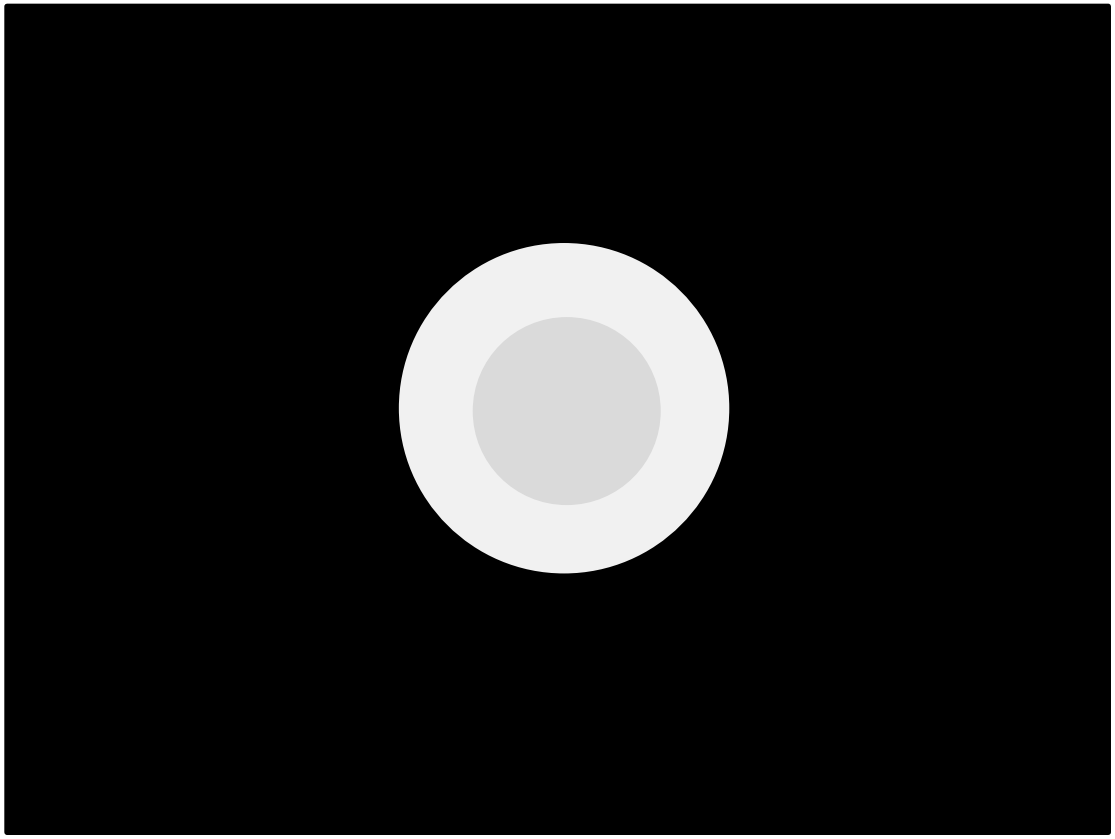
Motivation: Why Gradient Domain?

1. Great for human perception



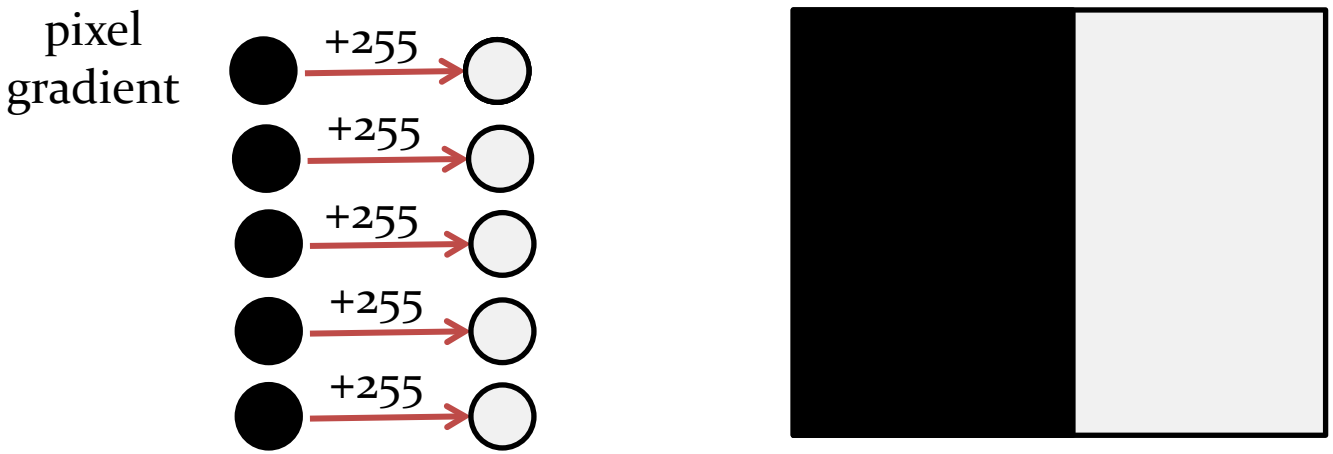
Motivation: Why Gradient Domain?

1. Great for human perception



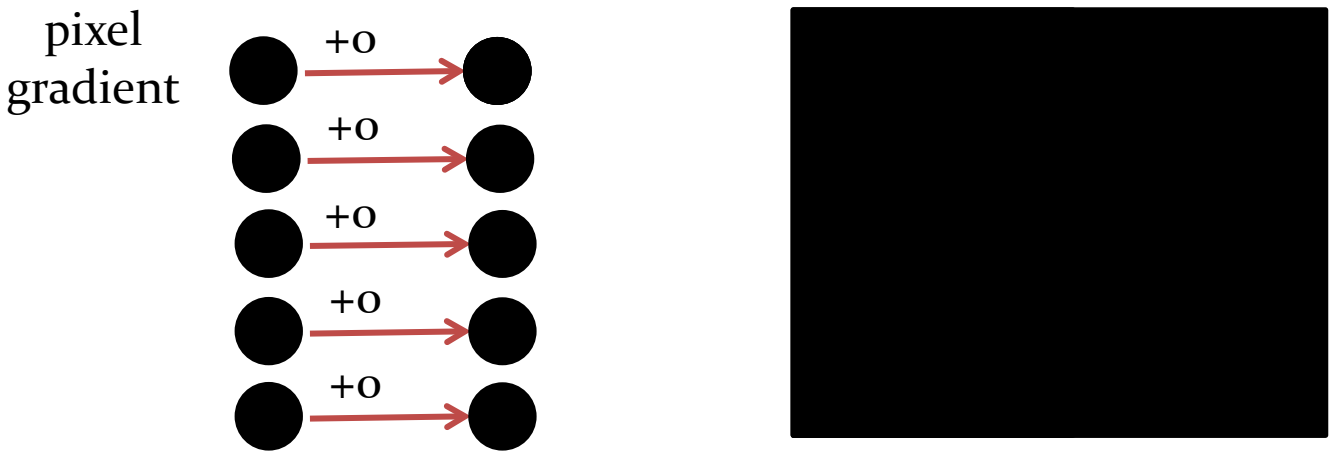
Motivation: Why Gradient Domain?

2. High-level control over images



Motivation: Why Gradient Domain?

2. High-level control over images



Motivation: Why Gradient Domain?

1. Great for human perception

2. High-level control over images

High-level Applications

- enhance texture
- change shading/lighting
- reduce artifacts
- change colors, preserve edges

Agenda

- Motivation
- ➡ Goal and Algorithm
- Applications and Results
- Discussion

Goal and Algorithm

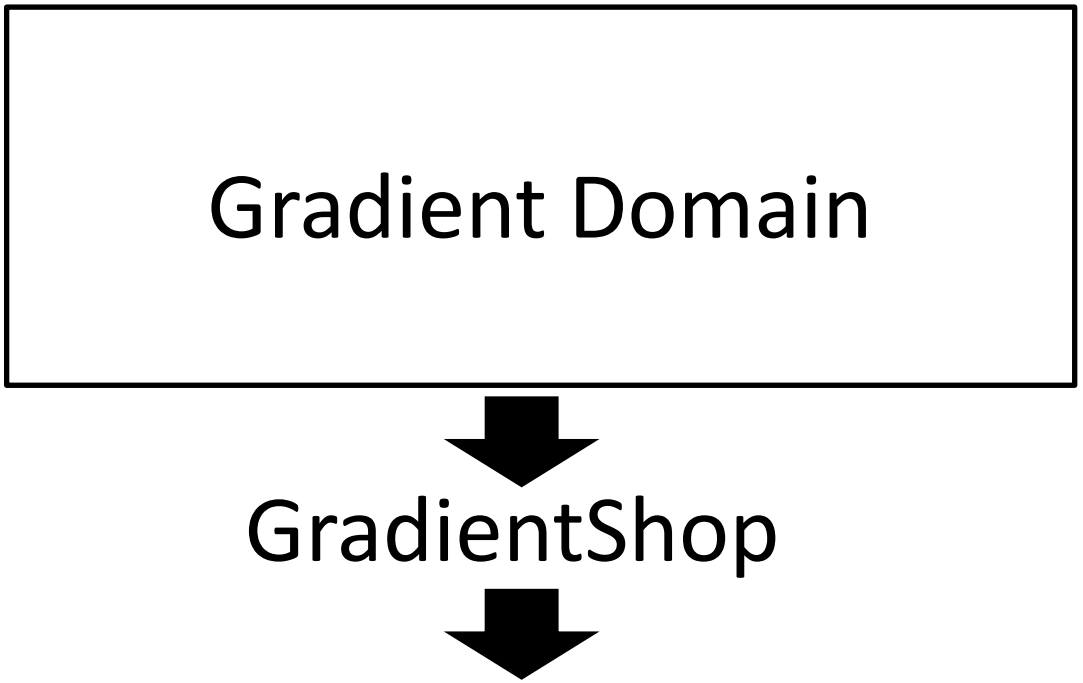
1. Great for human perception

2. High-level control over images

High-level Applications

- enhance texture
- change shading/lighting
- reduce artifacts
- change colors, preserve edges

Goal and Algorithm



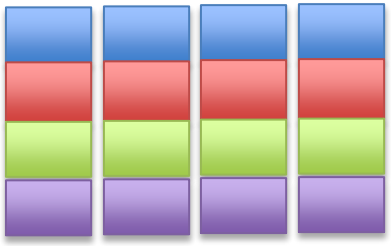
High-level Applications

- enhance texture
- change shading/lighting
- reduce artifacts
- change colors, preserve edges

Goal and Algorithm

Start with Gradient Domain: How Matrices Work?

Input Image (4x4):

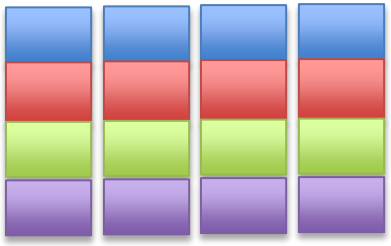


Each square = 1 pixel

Goal and Algorithm

Start with Gradient Domain: How Matrices Work?

Input Image (4x4):



What we want:

Matrix form of
Gradient Domain

$\Delta x = I(x,y) - I(x-1,y)$

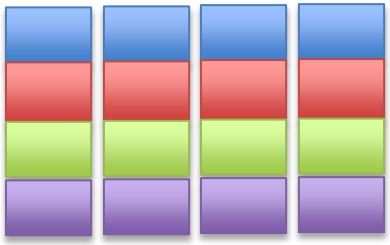
$\Delta y = I(x,y) - I(x, y-1)$

Goal and Algorithm

Start with Gradient Domain: How Matrices Work?

Image Matrix (I)

Input Image (4x4):



(16 x 1)

Goal and Algorithm

Start with Gradient Domain: How Matrices Work?

Fancy Smancy Laplacian Matrix (L)

$$\begin{bmatrix} \Delta x = I(x,y) - I(x-1,y) \\ \hline \Delta y = I(x,y) - I(x, y-1) \end{bmatrix}$$

(32 x 16)

X

Image Matrix (I)

(16 x 1)

Goal and Algorithm

Start with Gradient Domain: How Matrices Work?

Fancy Smancy Laplacian Matrix (L)

Image Matrix (I)

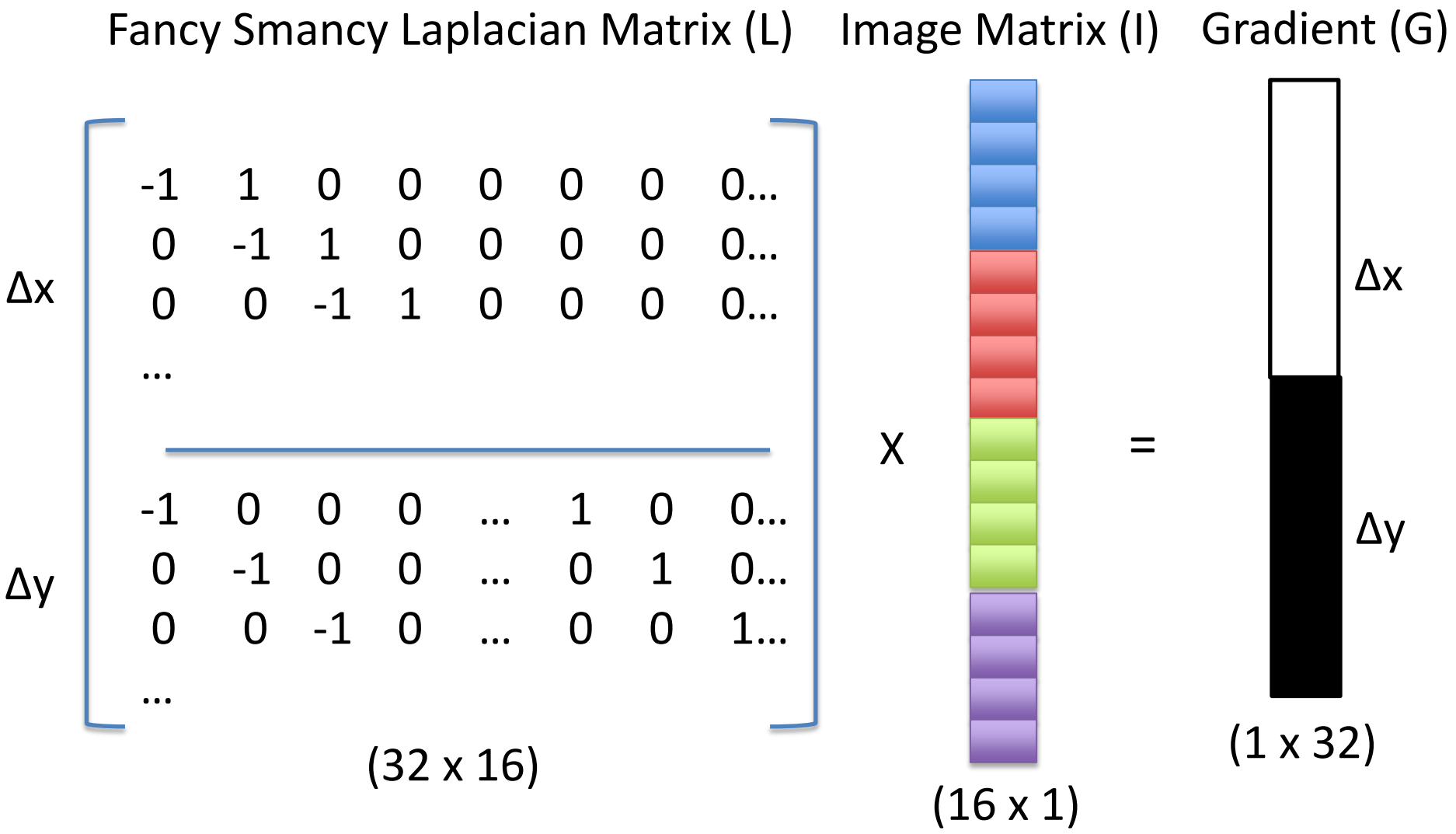
$$
\begin{matrix} \Delta x \\ \\ \\ \\ \Delta y \\ \\ \\ \end{matrix}
\begin{bmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\ldots \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0\ldots \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0\ldots \\
\ldots & & & & & & & \\
\hline
-1 & 0 & 0 & 0 & \ldots & 1 & 0 & 0\ldots \\
0 & -1 & 0 & 0 & \ldots & 0 & 1 & 0\ldots \\
0 & 0 & -1 & 0 & \ldots & 0 & 0 & 1\ldots \\
\ldots & & & & & & &
\end{bmatrix}
\quad X \quad
\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}
$$

(32 x 16)

(16 x 1)

Goal and Algorithm

Start with Gradient Domain: How Matrices Work?

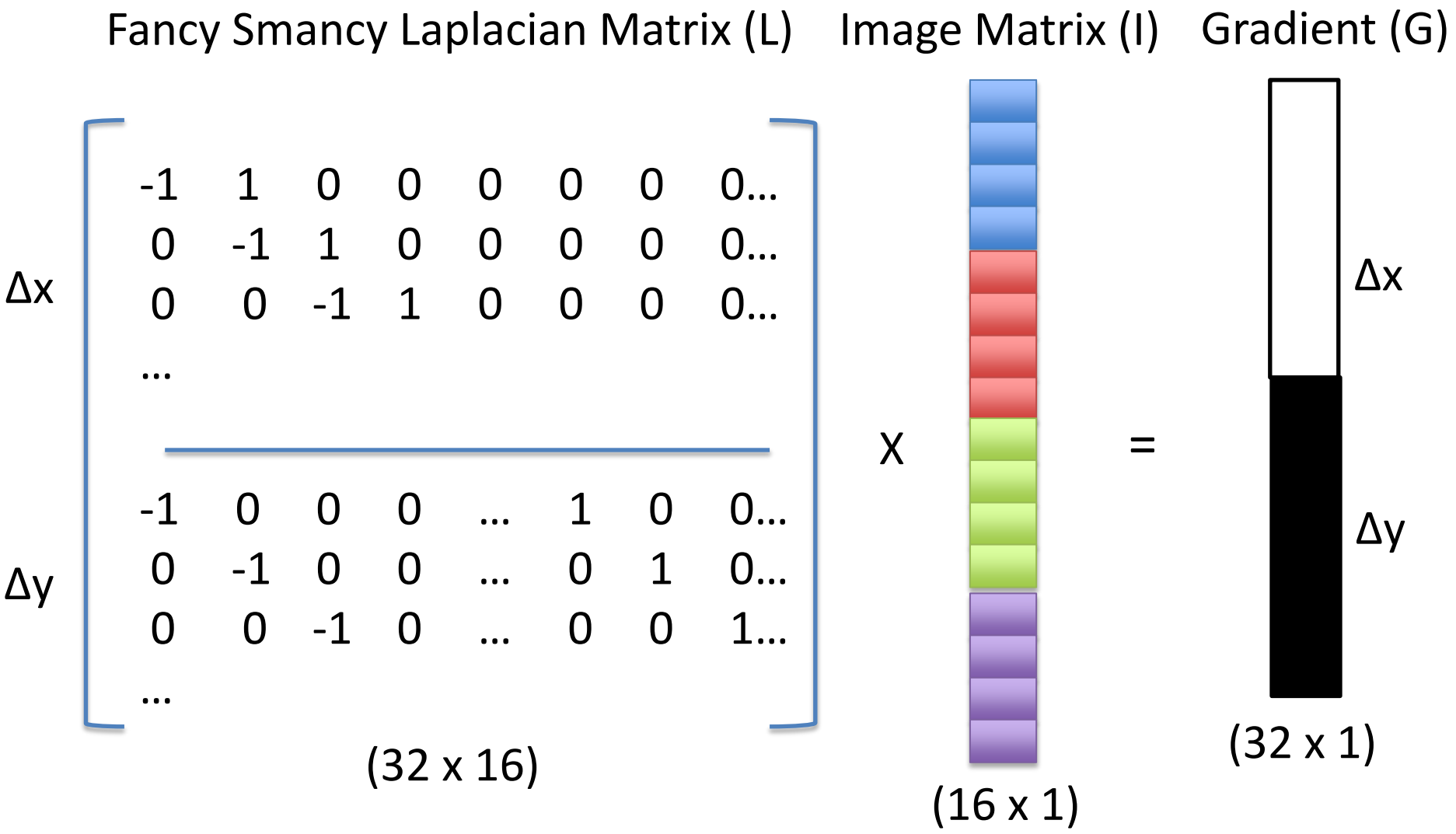


Goal and Algorithm

How to “GradientShop” This?

Goal and Algorithm

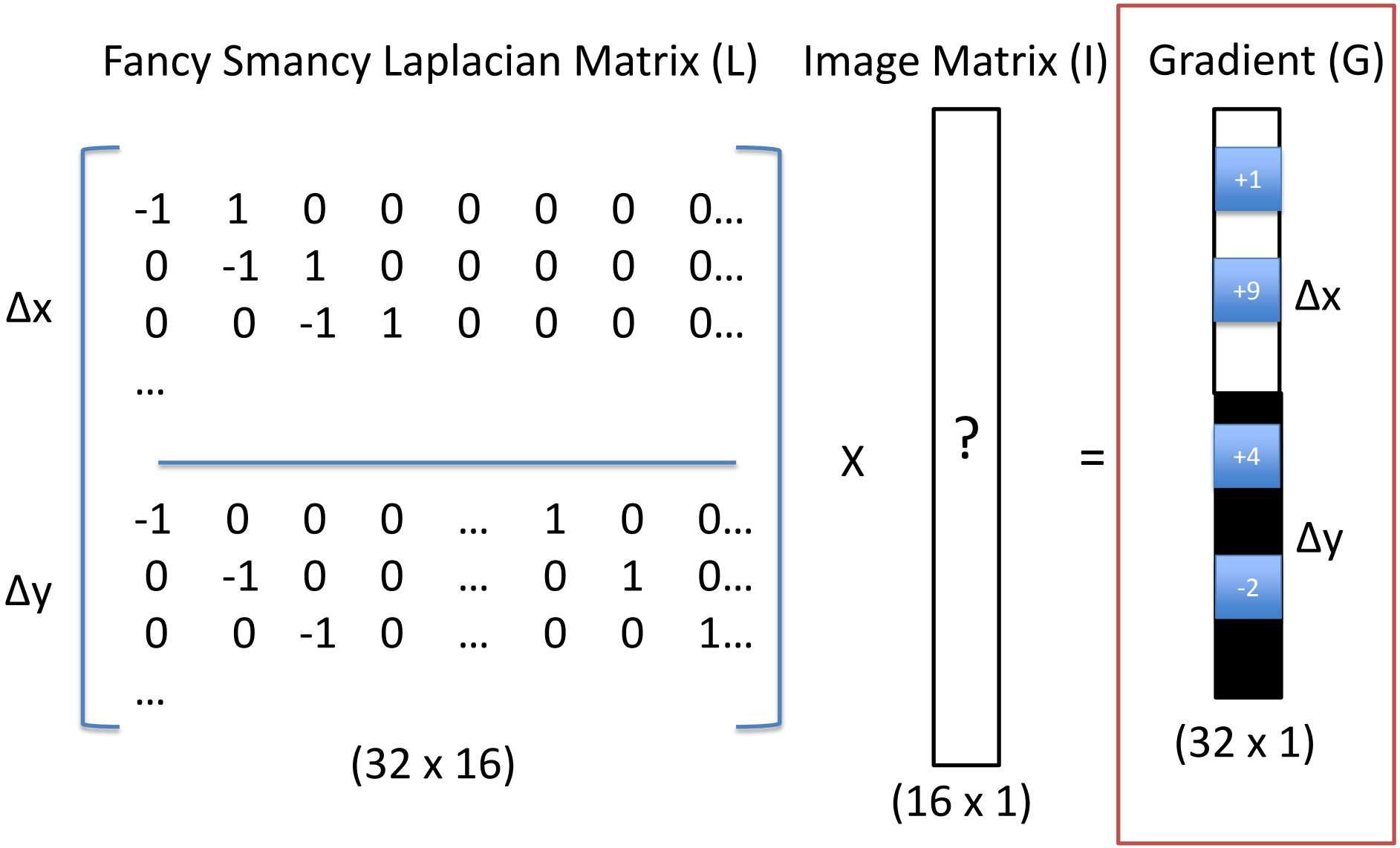
1. Give Gradient Constraints



Goal and Algorithm

1. Give Gradient Constraints

1. User Modifies:



Goal and Algorithm

1. Give Gradient Constraints

2. Solve Image:

Fancy Smancy Laplacian Matrix (L)

$$
\begin{array}{l}
\Delta x \\
\\
\Delta y
\end{array}
\begin{bmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\ldots \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0\ldots \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0\ldots \\
\ldots & & & & & & & \\
\hline
-1 & 0 & 0 & 0 & \ldots & 1 & 0 & 0\ldots \\
0 & -1 & 0 & 0 & \ldots & 0 & 1 & 0\ldots \\
0 & 0 & -1 & 0 & \ldots & 0 & 0 & 1\ldots \\
\ldots & & & & & & &
\end{bmatrix}
$$

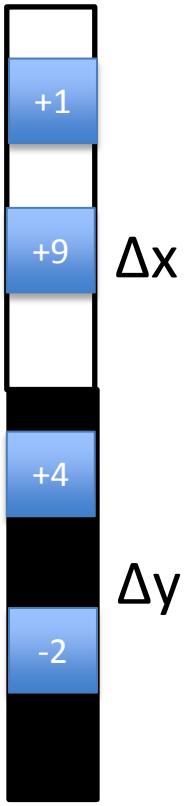
(32 x 16)

Image Matrix (I)

$$
X \quad \begin{bmatrix} ? \end{bmatrix} \quad =
$$

(16 x 1)

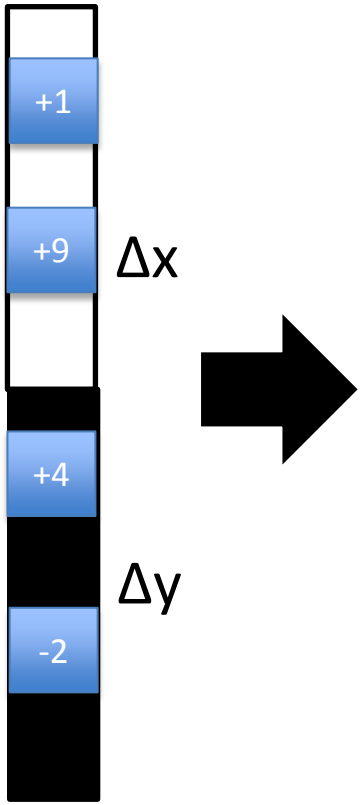
Gradient (G)



(32 x 1)

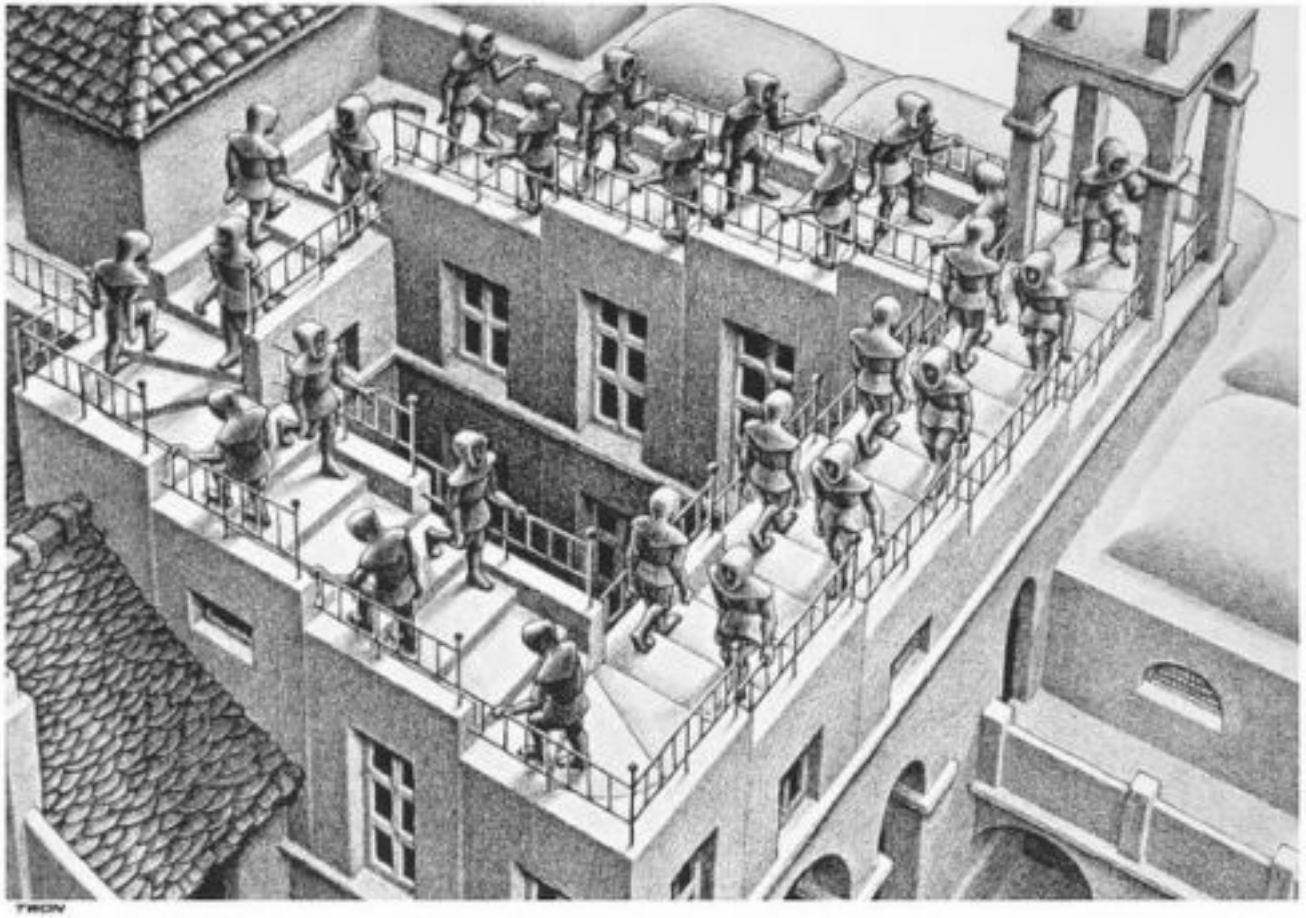
Goal and Algorithm

Gradient (G)



(32 x 1)

Curl Not Equating to Zero

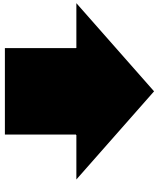


Goal and Algorithm

We want:

$$L \times I = G$$

$$L \times I - G = 0$$

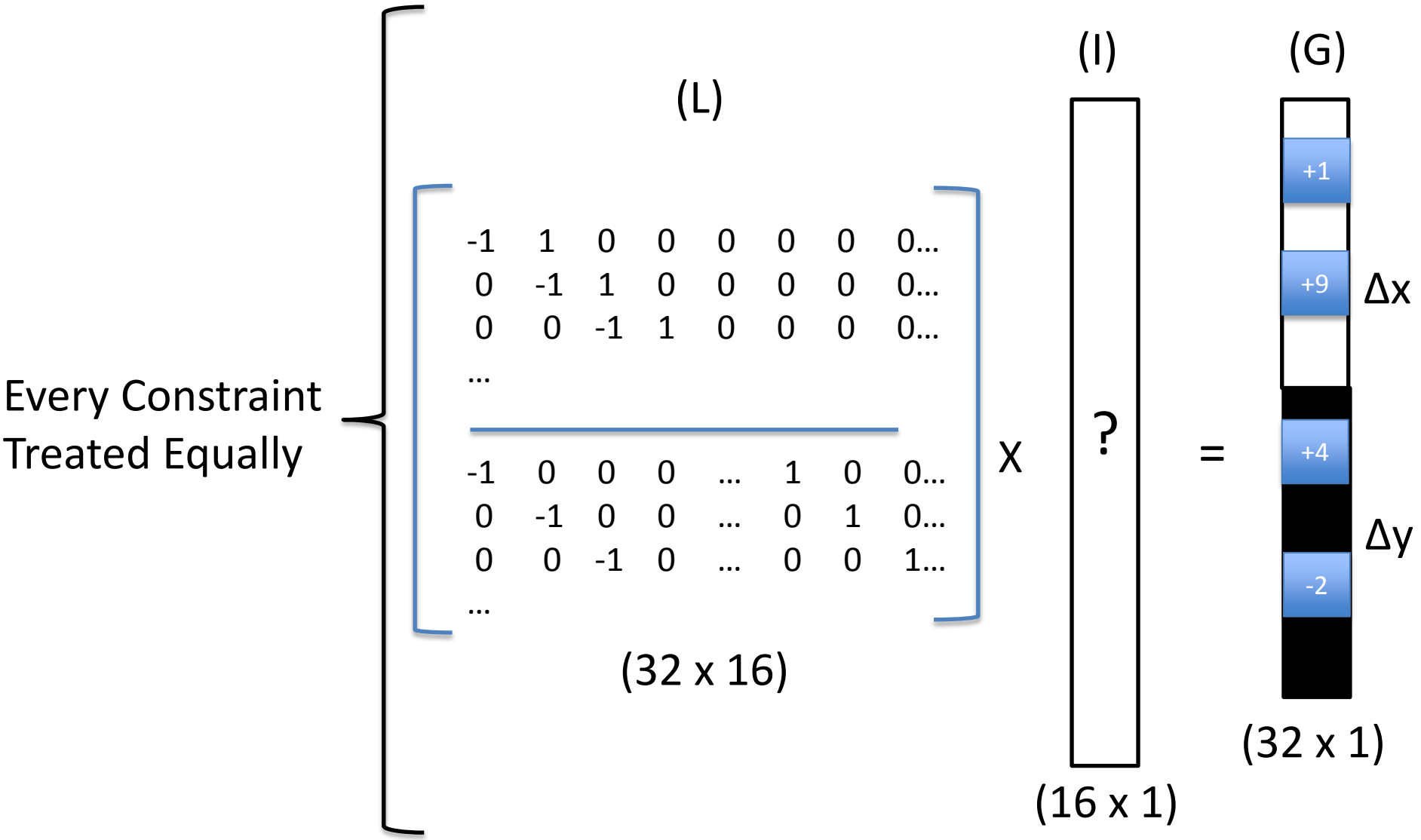


Poisson (“backslash”)
To solve for unknown I

$$L \times I - G \approx 0$$

$$\text{Error} = L \times I - G$$

Goal and Algorithm

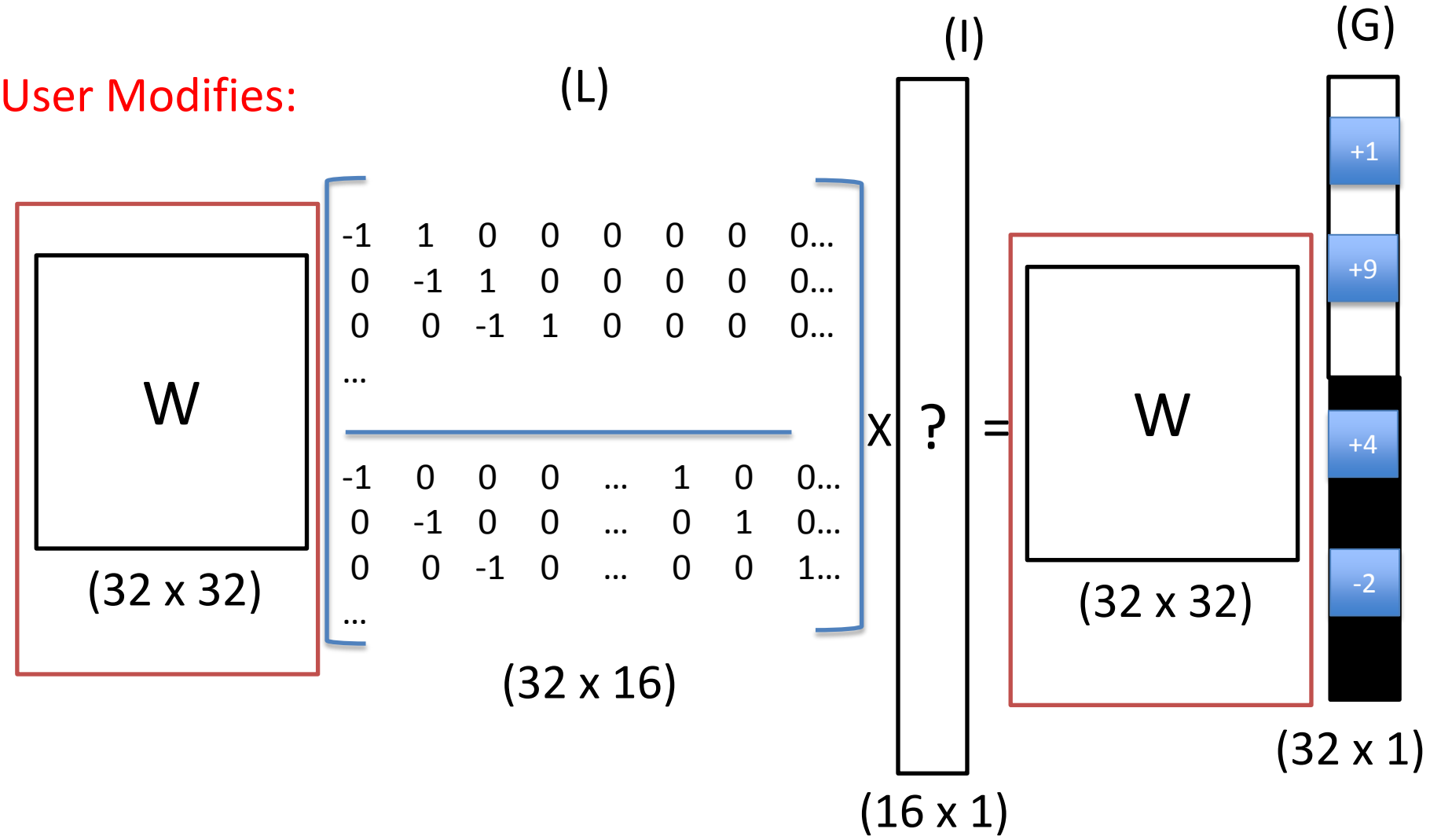


Goal and Algorithm

2. Add weights

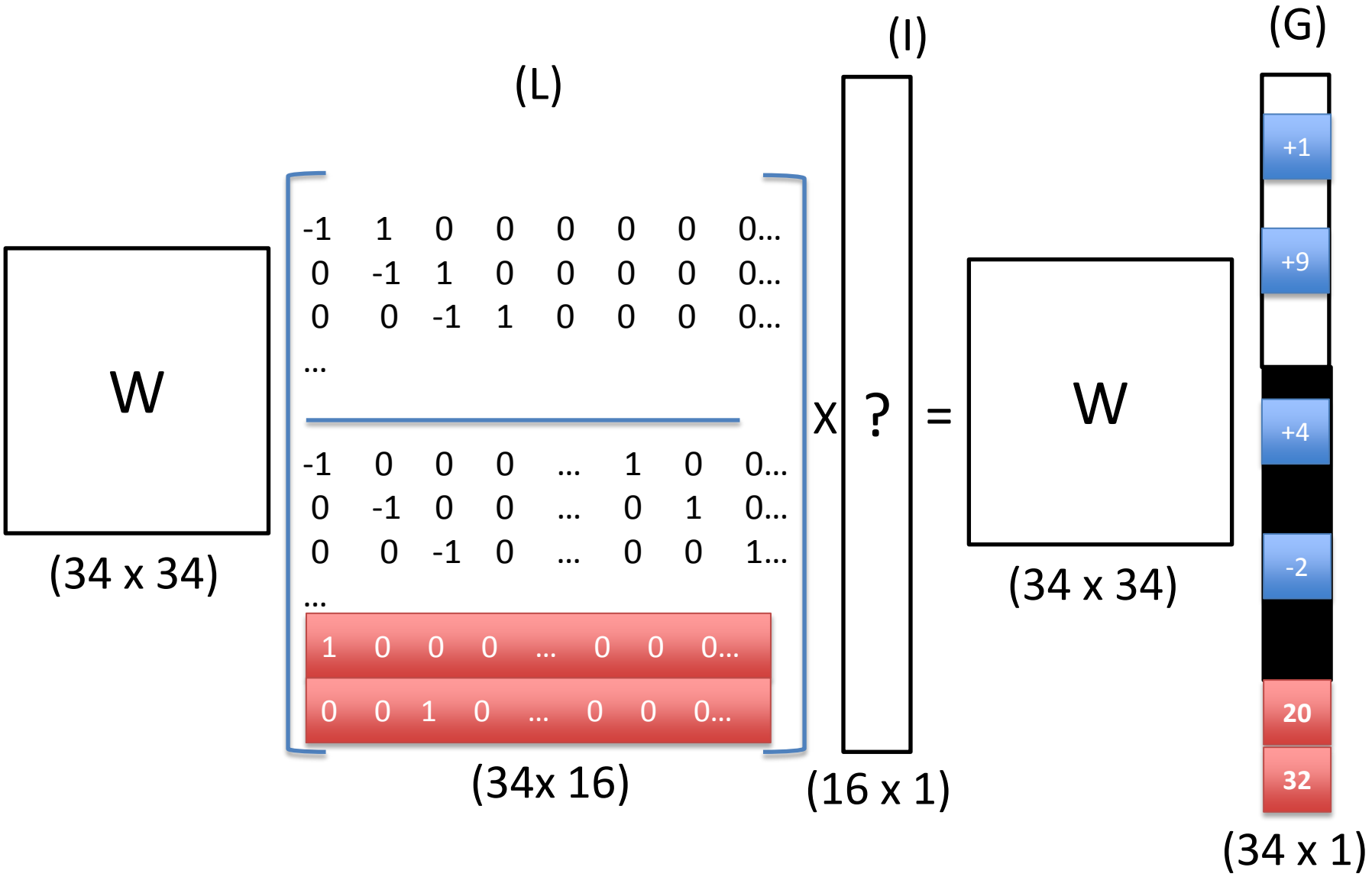
Not everyone is Equal…

User Modifies:



Goal and Algorithm

3. How about the image values themselves?



Goal and Algorithm

Previous Works:

Error = (Gradient Constraint)

1. Give Gradient Constraints
2. Add weights
3. How about the image values themselves?

Gradient Shop:

$$\text{Error} = \text{Weights}_{\text{Gradient}} * (\text{Gradient Constraint}) + \text{Weights}_{\text{Image}} * (\text{Image Constraint})$$

Agenda

- Motivation
- Goal and Algorithm
- ➡ Applications and Results
- Discussion

Applications and Results

Gradient Shop:

$$\text{Error} = \text{Weights}_{\text{Gradient}} * (\text{Gradient Constraint}) + \text{Weights}_{\text{Image}} * (\text{Image Constraint})$$

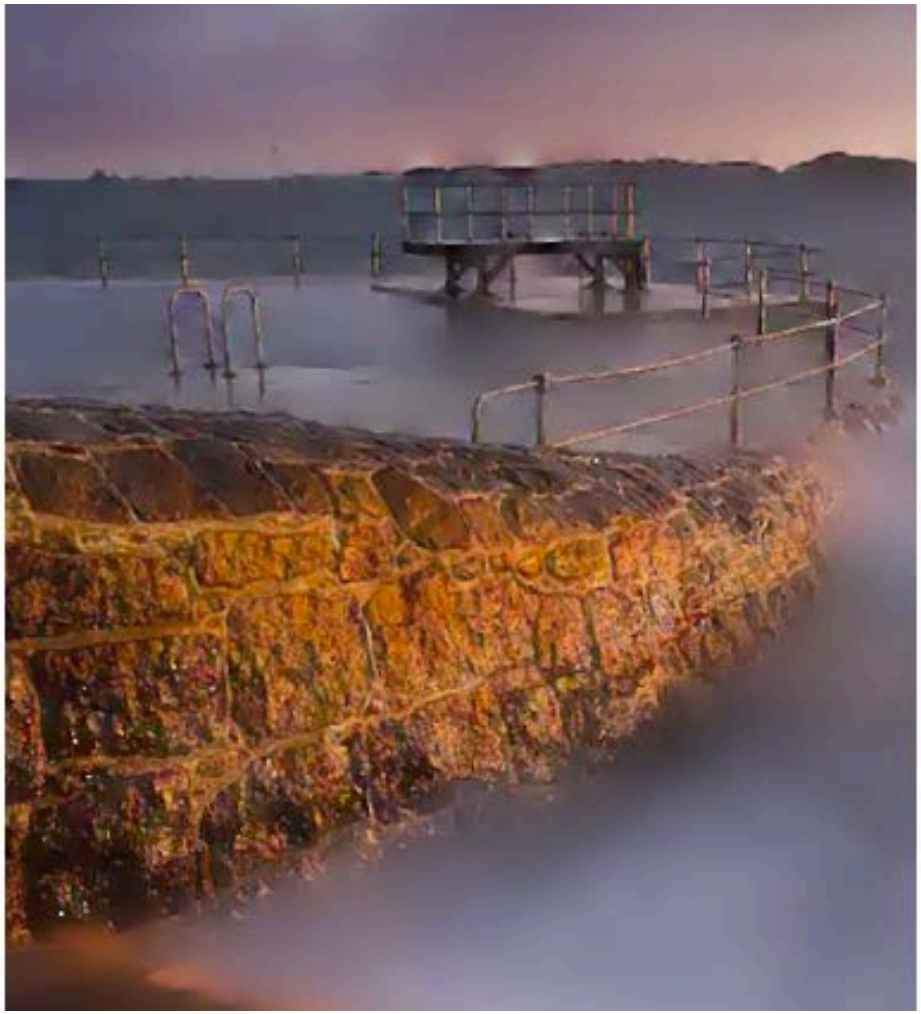
High-level Applications

- enhance texture
- change shading/lighting
- reduce artifacts
- change colors, preserve edges

Applications and Results

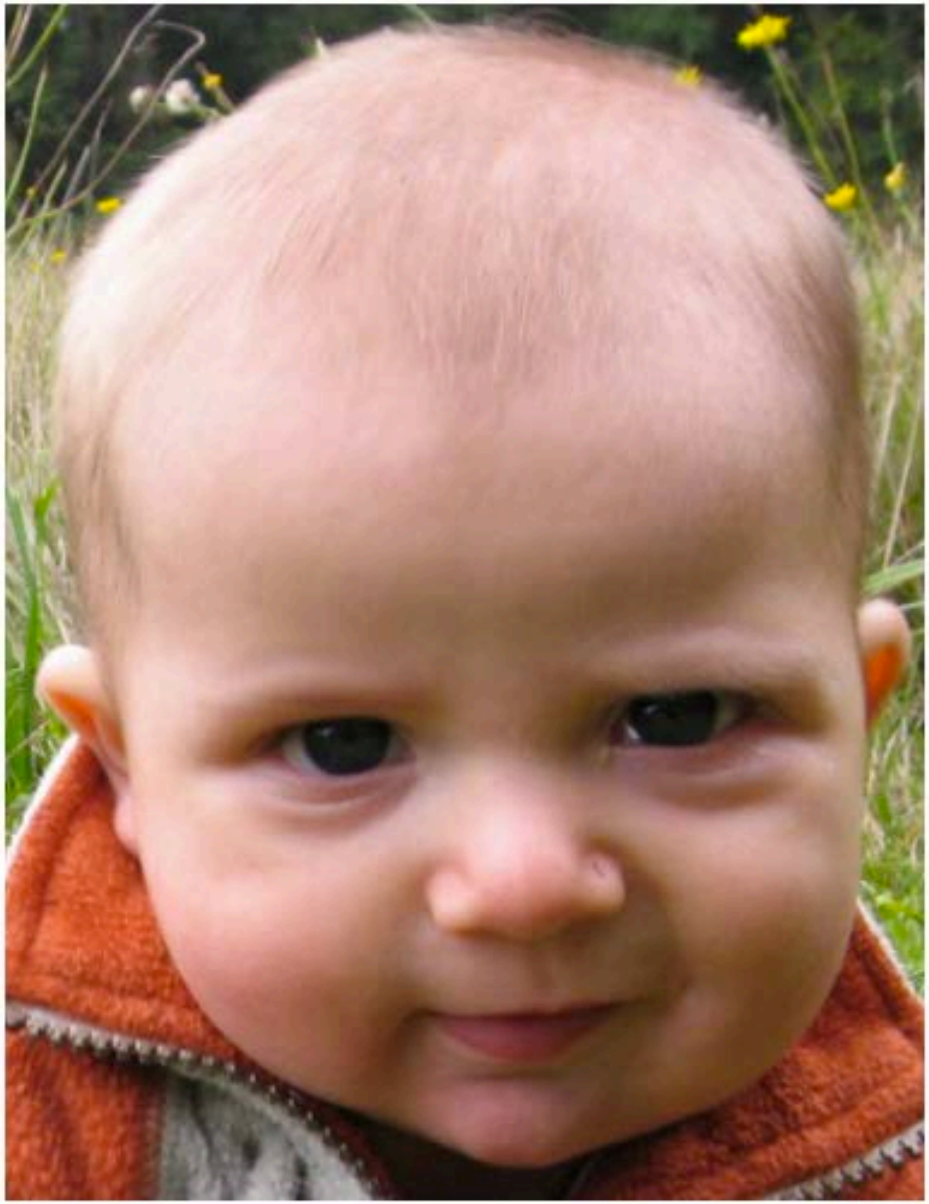
Input Image

Softened Edges



Applications and Results

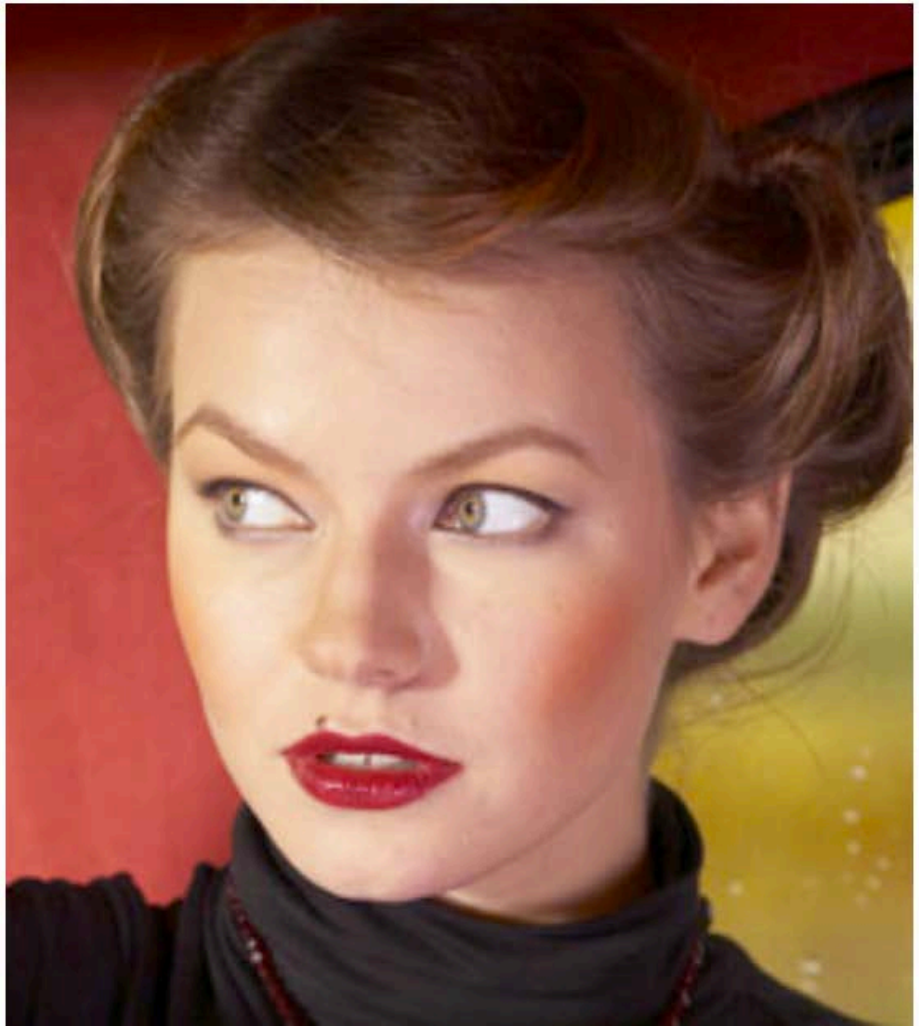
Input Image



Relit Image

Applications and Results

Input Image



Non-photorealistic Stuff

Discussion

Questions?