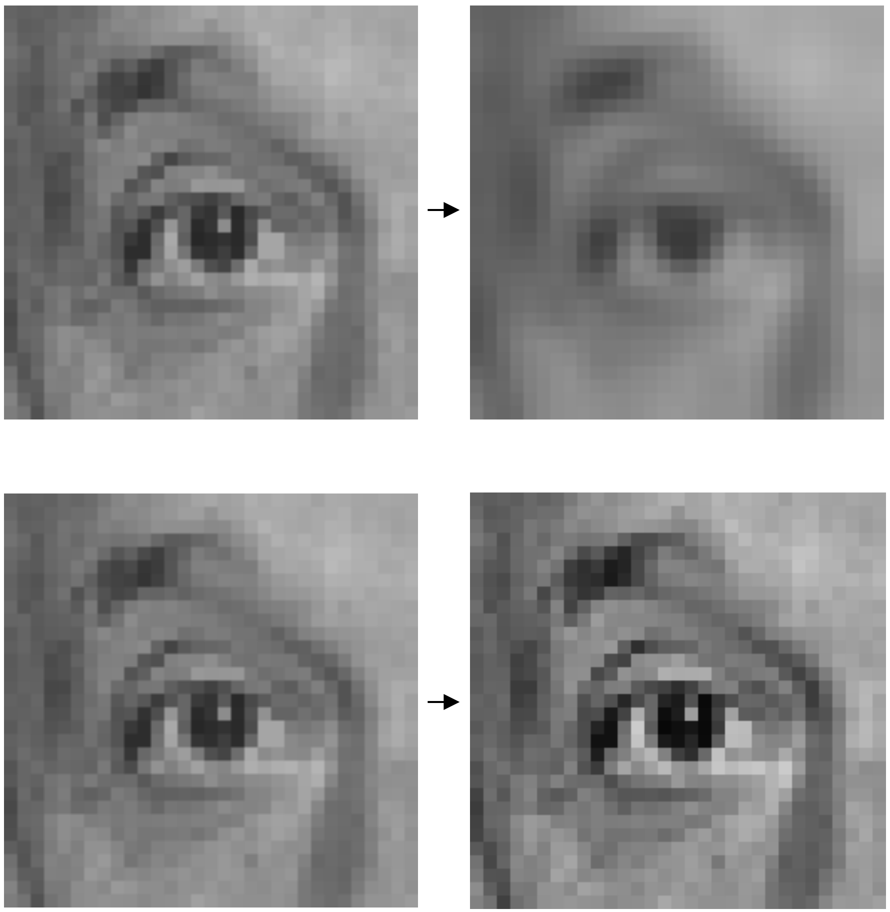
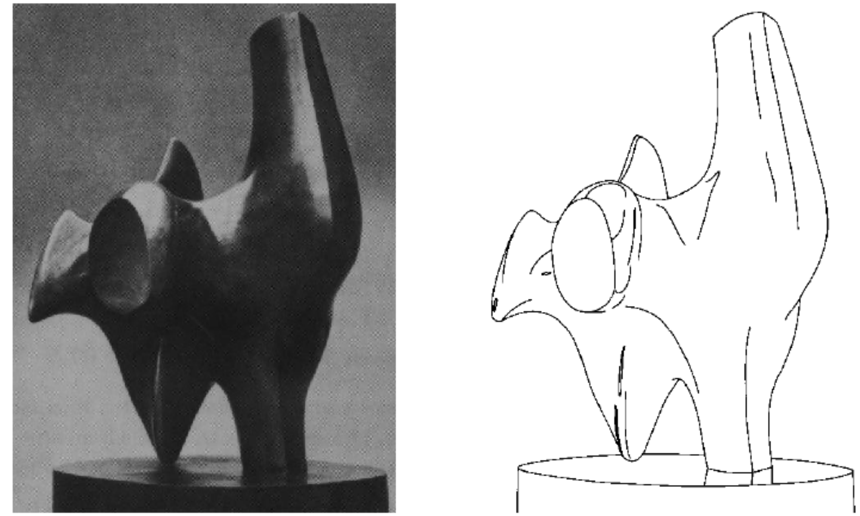
Last Lecture



Filtering



Edge Detection

Pyramid

High Dynamic Range Image Reconstruction from Hand-held Cameras

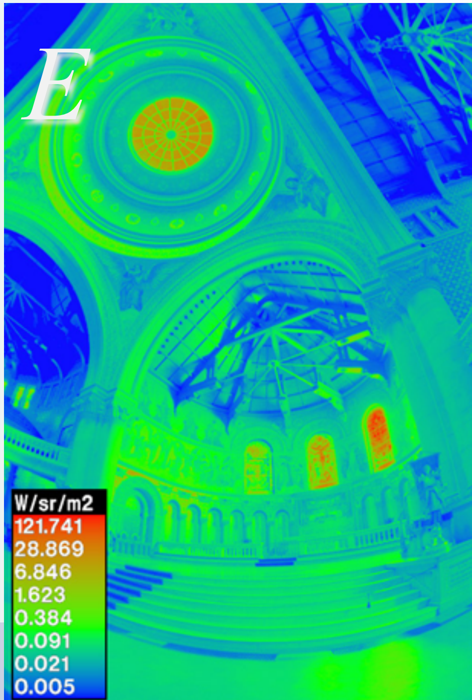
Pei-Ying Lu   Tz-Huan Huang   Meng-Sung Wu
Yi-Ting Cheng   Yung-Yu Chuang

National Taiwan University

IEEE Computer Society Conference on Computer Vision and Pattern Recognition   Fontainebleau Resort, Miami Beach, Florida   June 20-25, 2009

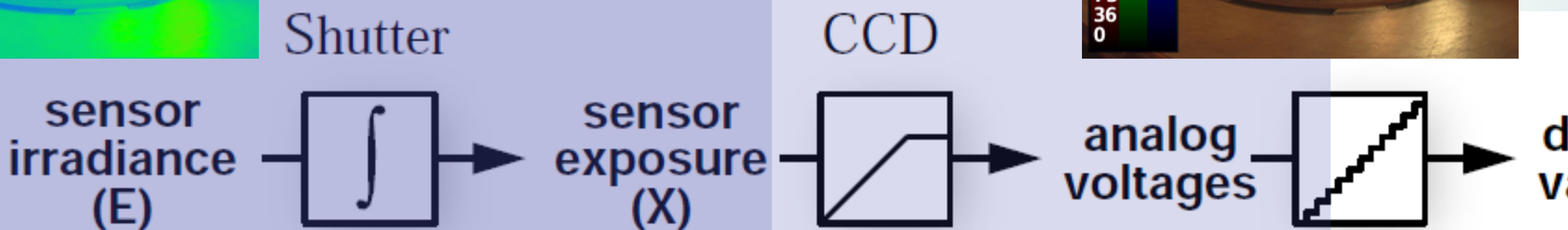
The world is of high dynamic range

Camera pipeline

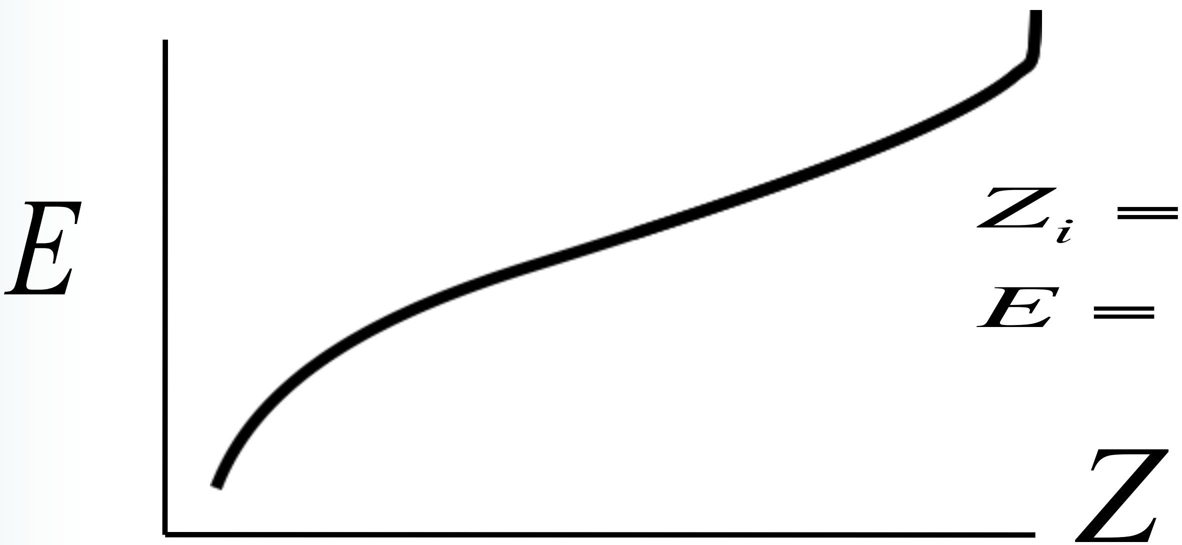


$$X = \int_{t=0}^{\Delta t} E\,dt = E\Delta t$$

$$Z = f(X) = f(E\Delta t)$$



HDR image reconstruction



$$Z_i = f(E\Delta t_i)$$
$$E = f^{-1}(Z_i) / \Delta t_i$$

HDR image reconstruction

- Recovering High Dynamic Range Radiance Maps from Photographs, SIGGRAPH 1997.
- Radiometric Self Calibration, CVPR 2001.
- Estimation-theoretic approach to dynamic range enhancement using multiple exposures, JEI 2003.
- All assume static cameras and thus require tripods.

Images from hand-held camera

- Challenge #1: image mis-alignment

Images from hand-held camera

- Challenge #1: image mis-alignment
- Challenge #2: image blur

A naïve approach

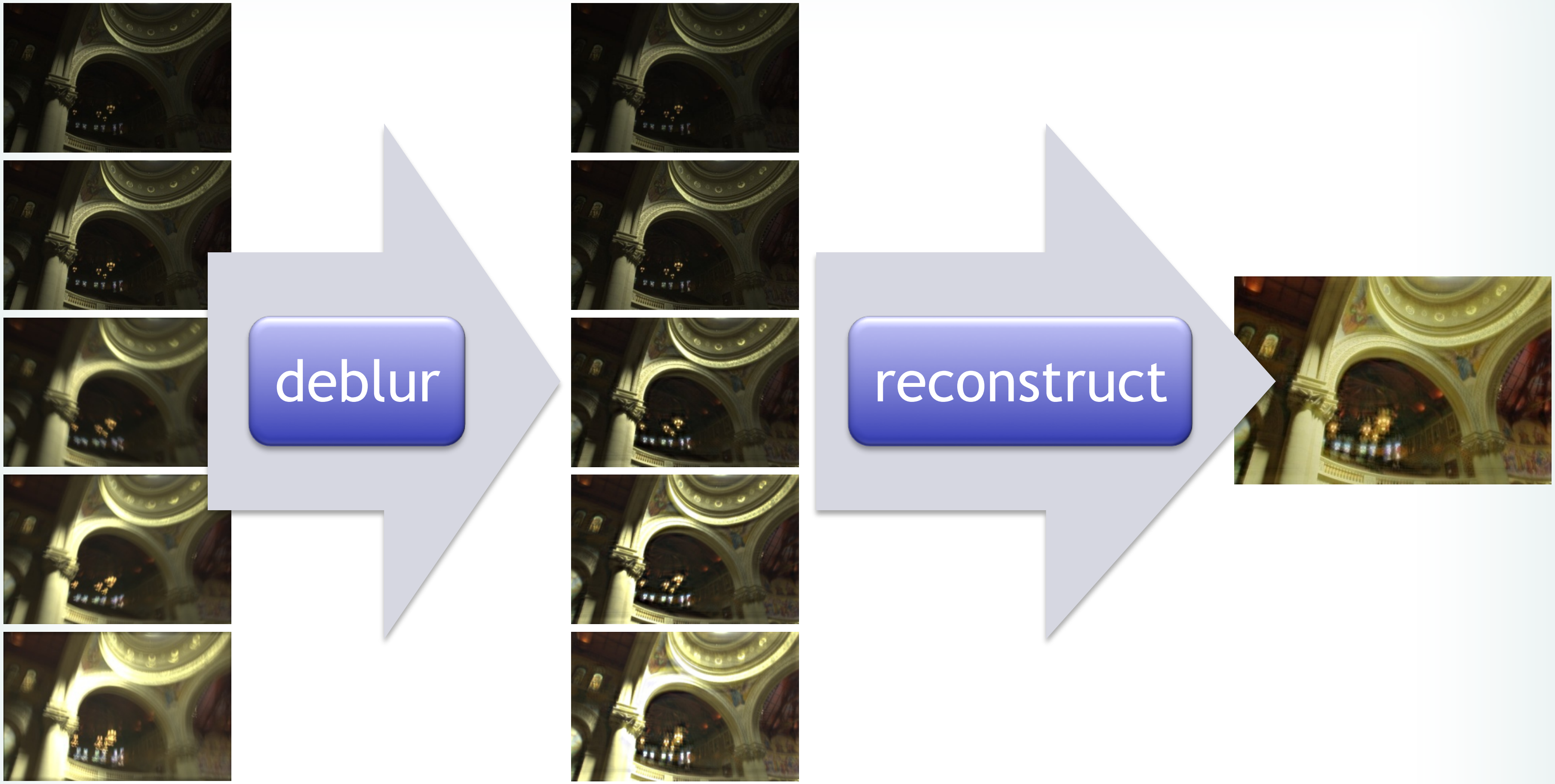


Image blurring process

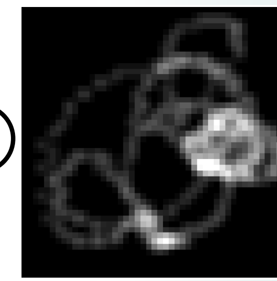
convolution

blur image

=

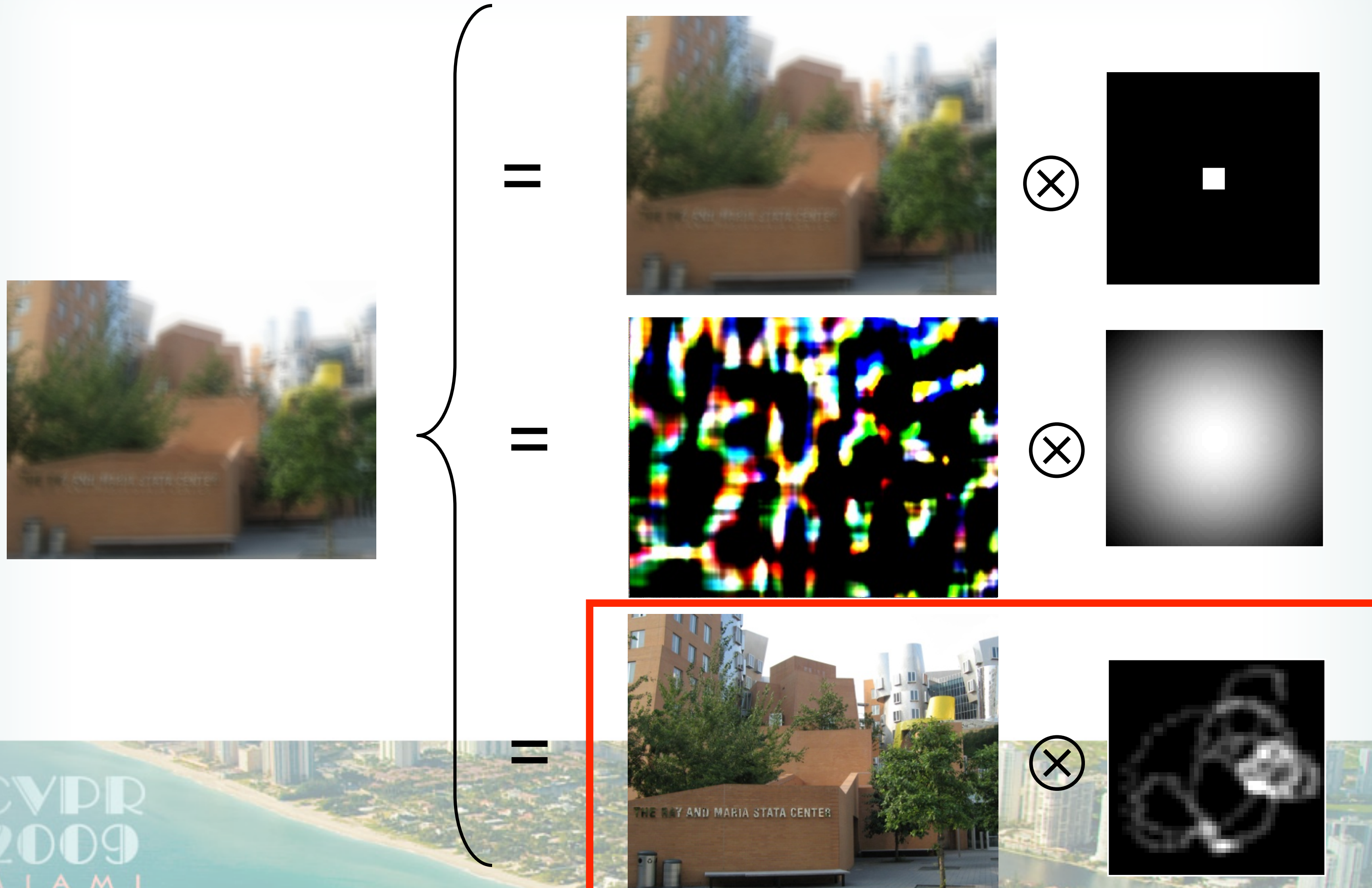
sharp image

⊗



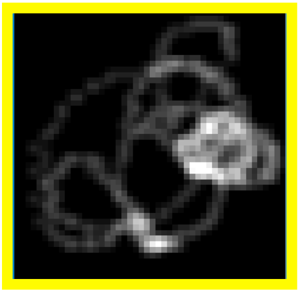
blur kernel

An under-determined problem



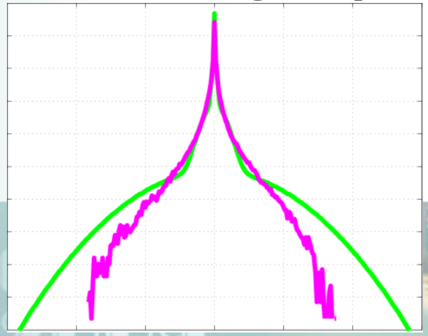
Adding priors

1. Reconstruction constraint:

 $\otimes$  = 

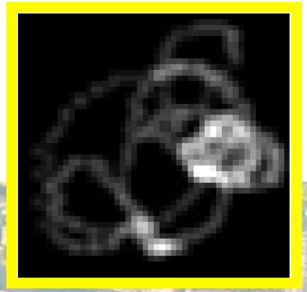
Fergus et. al. SIGGRAPH 2006

2. Image prior:



distribution of gradients

3. Blur prior:

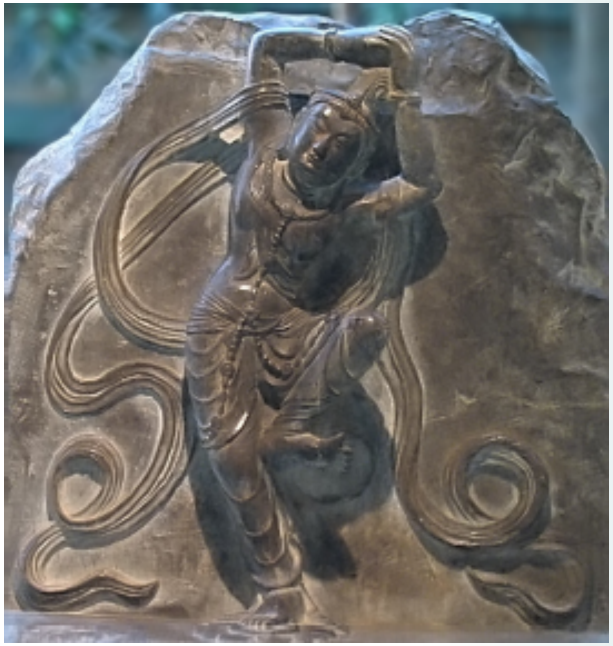


positive & sparse

Adding observation (a noisy image)


blurred image


noisy image


deblurred image

Yuan et. al. SIGGRAPH 2007

A naïve approach

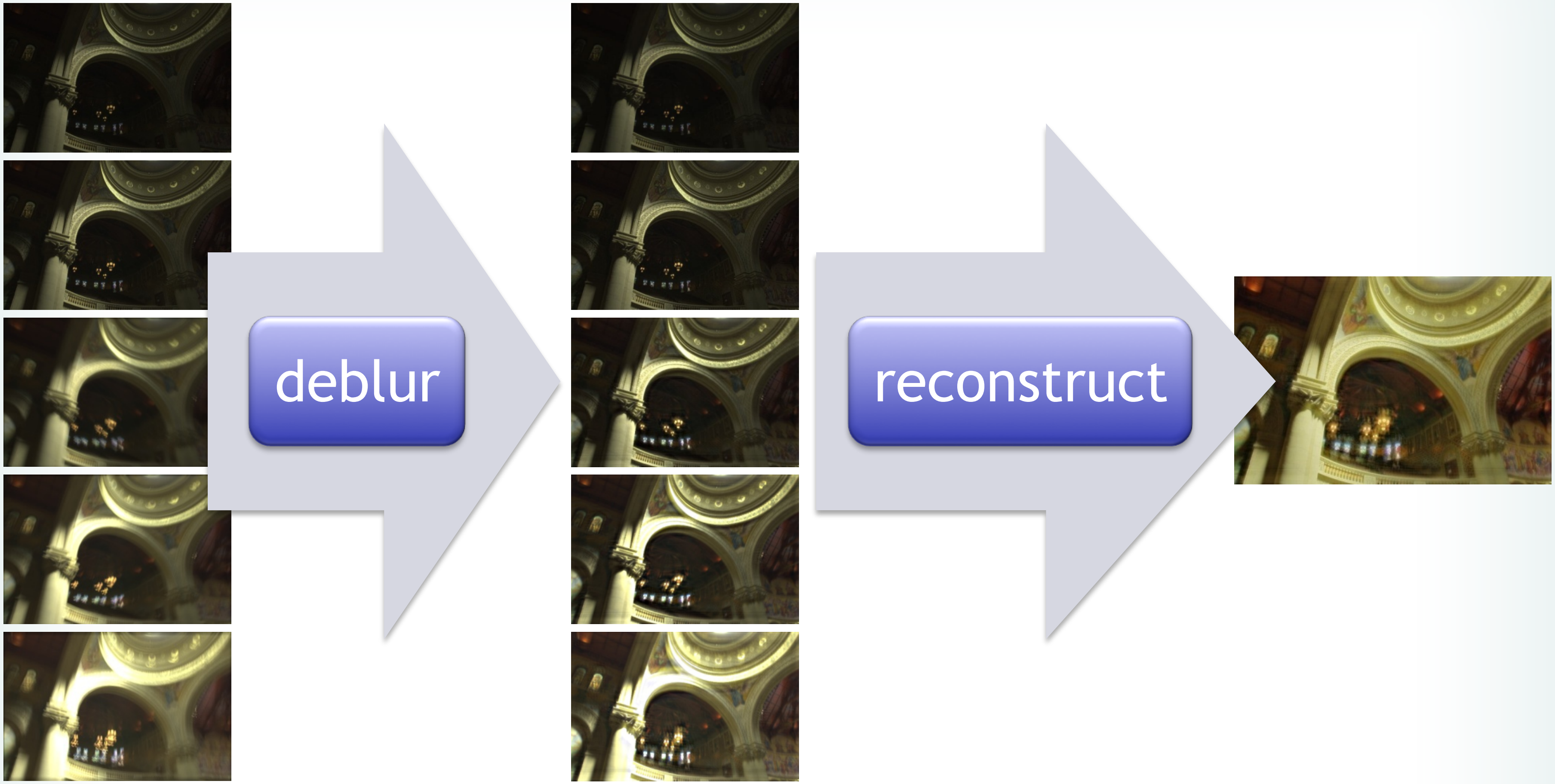


Image formation model

Observed LDR images | Exposure time

$$Z_i = f(\otimes K_i \Delta t_i)$$

Camera curve | Irradiance | Blur kernels

Image formation model

Observed LDR images

Exposure time

$$Z_i = f(\otimes K_i \Delta t_i)$$

Camera curve

Irradiance

Blur kernels

Main Idea

$$Z_{ij} = f((E \otimes K_j)_i \Delta t_j)$$

$$f^{-1}(Z_{ij}) / \Delta t_j = (E \otimes K_j)_i$$

$$\arg\min_{E, K_i, f} \sum_{i=1}^{N} \sum_{j=1}^{P} \left\| (E \otimes K_j)_i - f^{-1}(Z_{ij}) / \Delta t_j \right\|^2$$

Iterative solution

Fix E, f to solve K_j

Fix $K_j, f,$ to solve E

Fix E, K_j to solve f

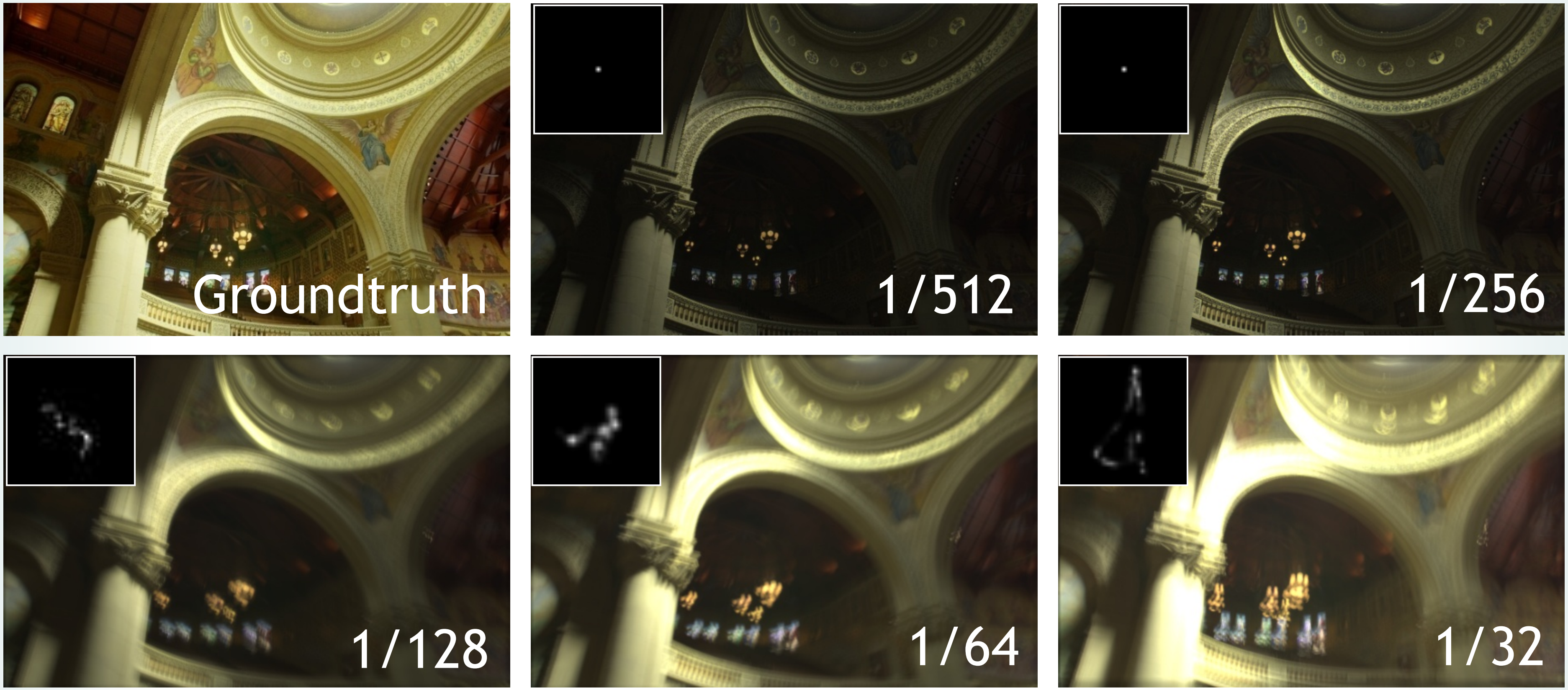
Optimization of f

$$g = f^{-1}$$

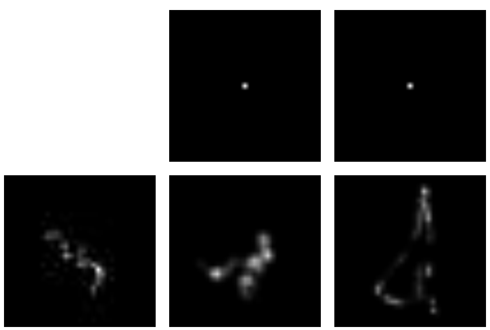
$$\arg\min_{g} \sum_{i=1}^{N} \sum_{j=1}^{P} \left\| (E \otimes K_j)_i - g(Z_{ij}) / \Delta t_j \right\|^2 + \lambda \sum_{z=Z_{\min}+1}^{Z_{\max}-1} g''(z)^2$$

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

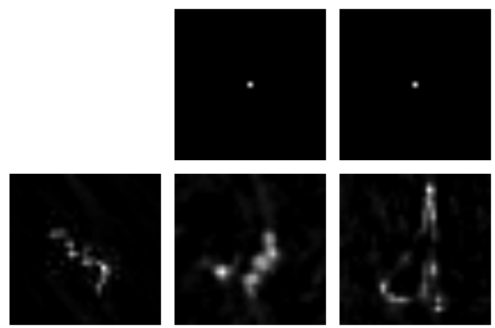
A synthetic example



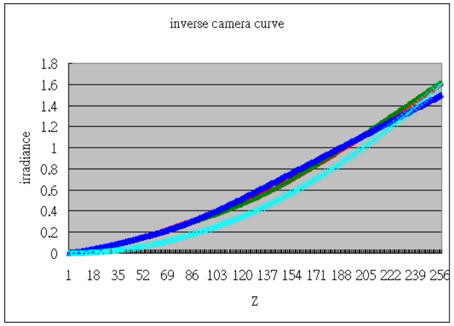
Our results



Groundtruth

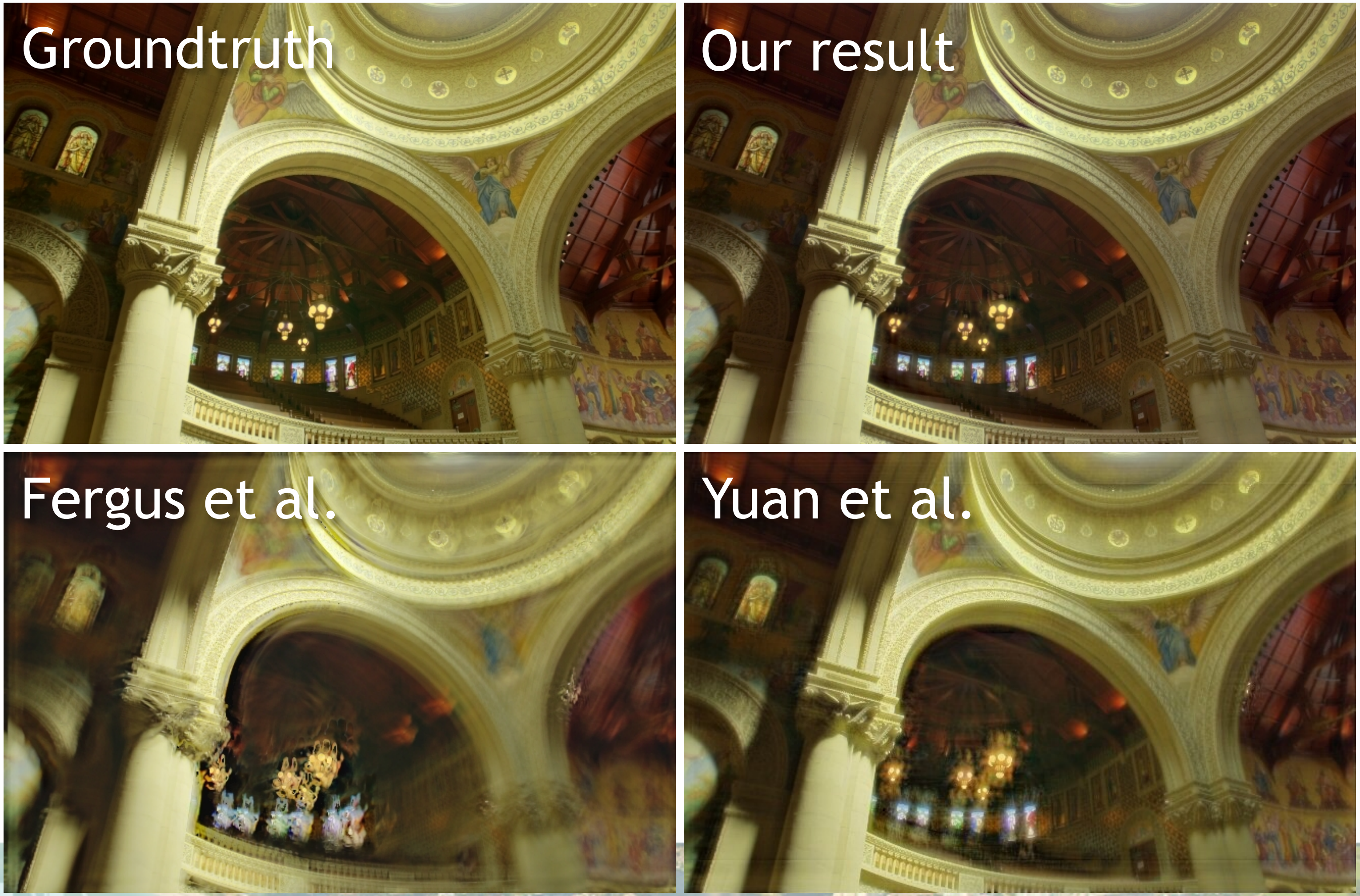


Our result



Camera curve

Comparisons



Groundtruth

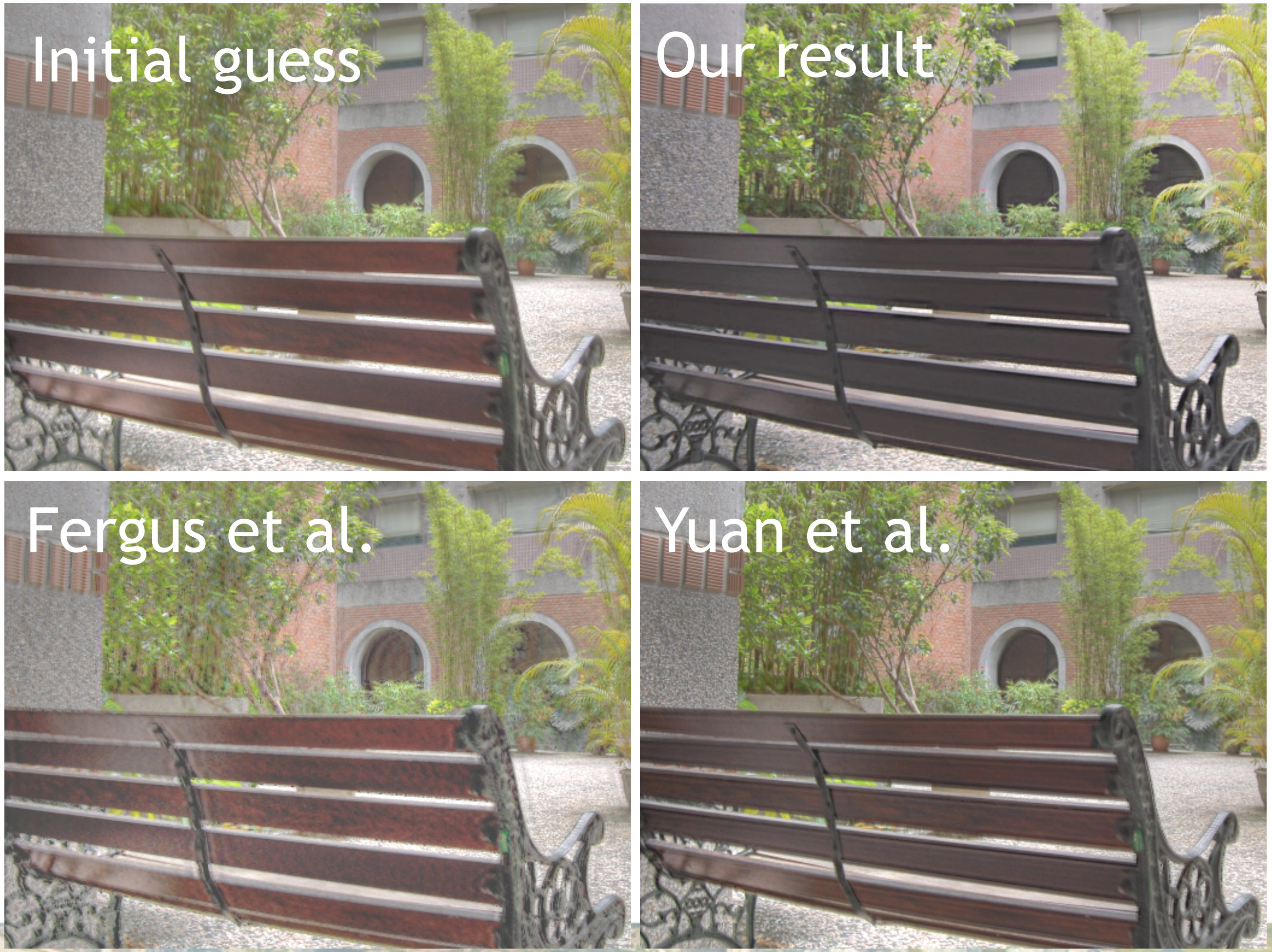
Our result

Yuan et al.

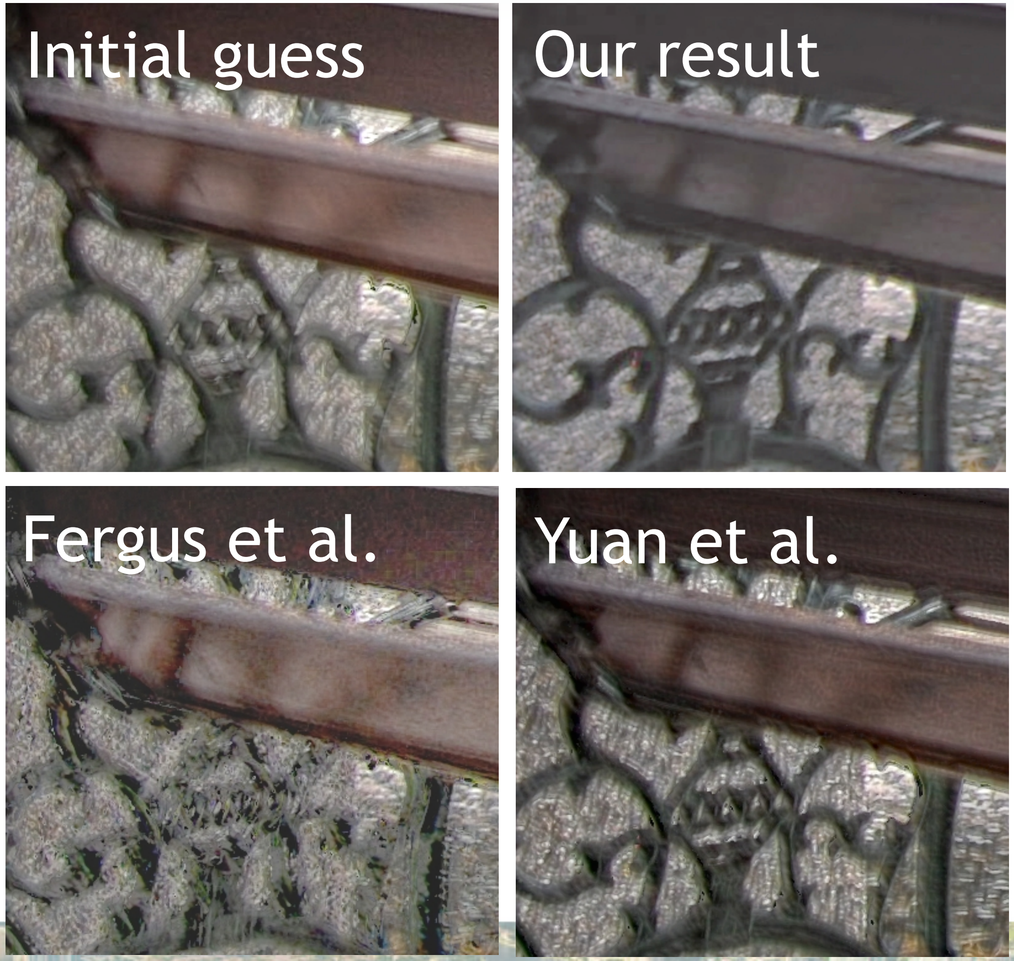
Fergus et al.

Real example I

Comparisons



Close-up comparisons

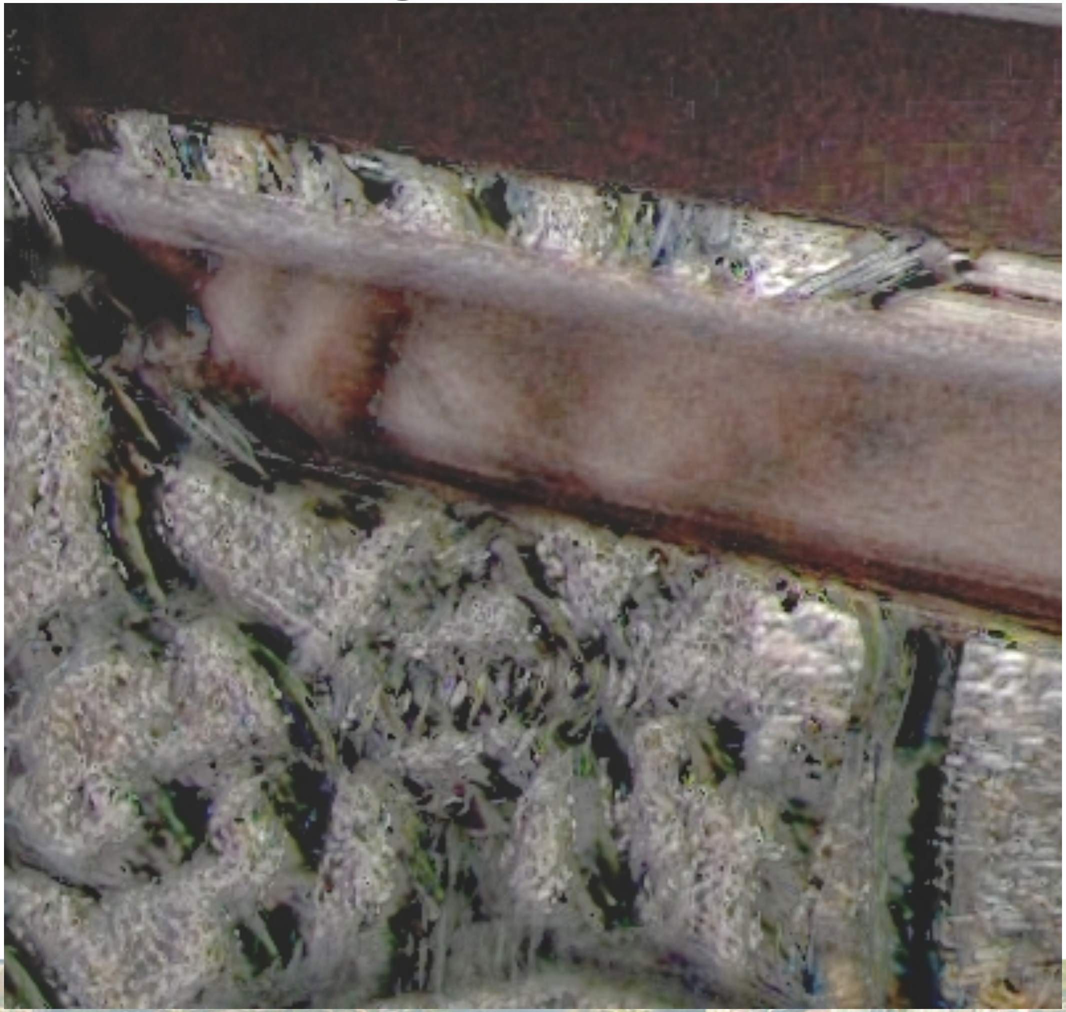


Initial guess

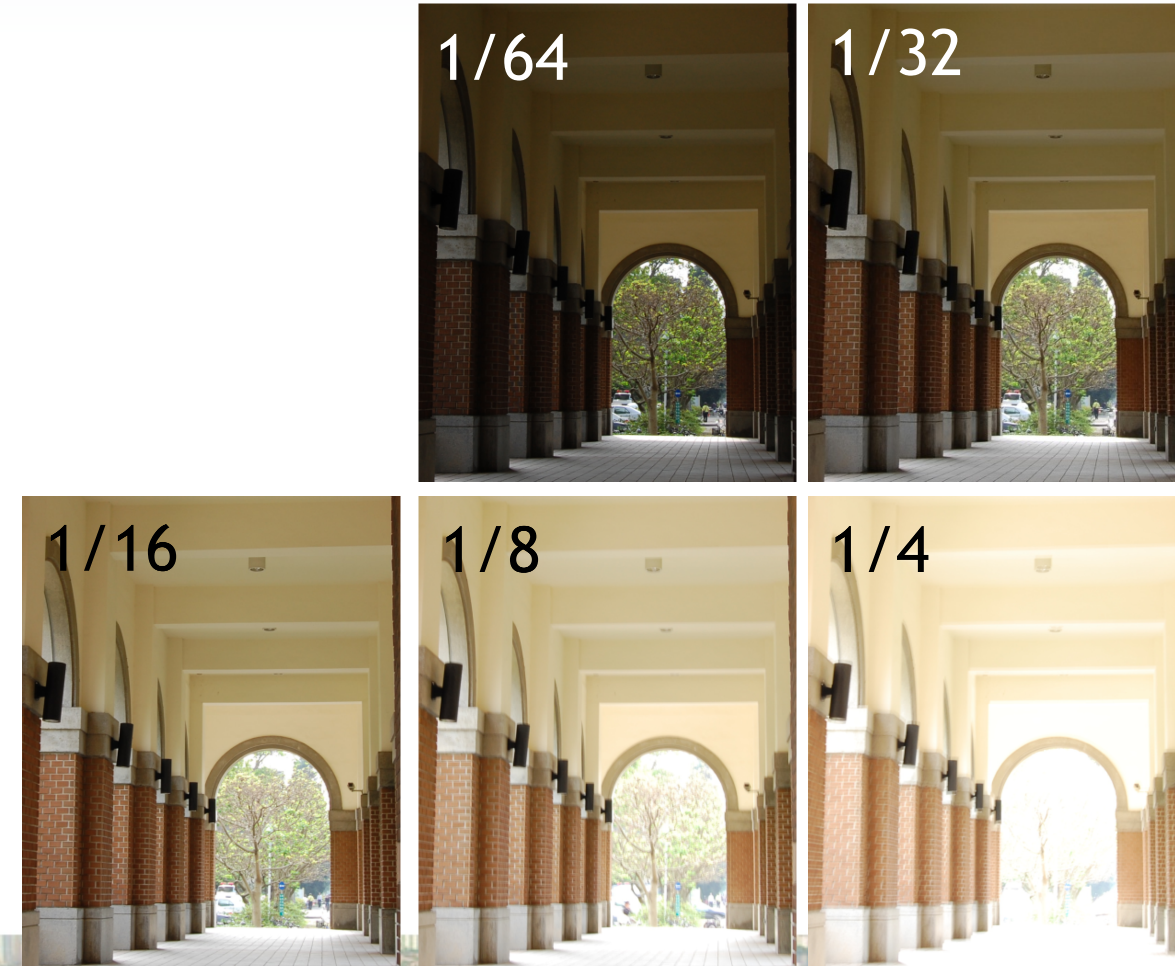
Our result

Yuan et al.

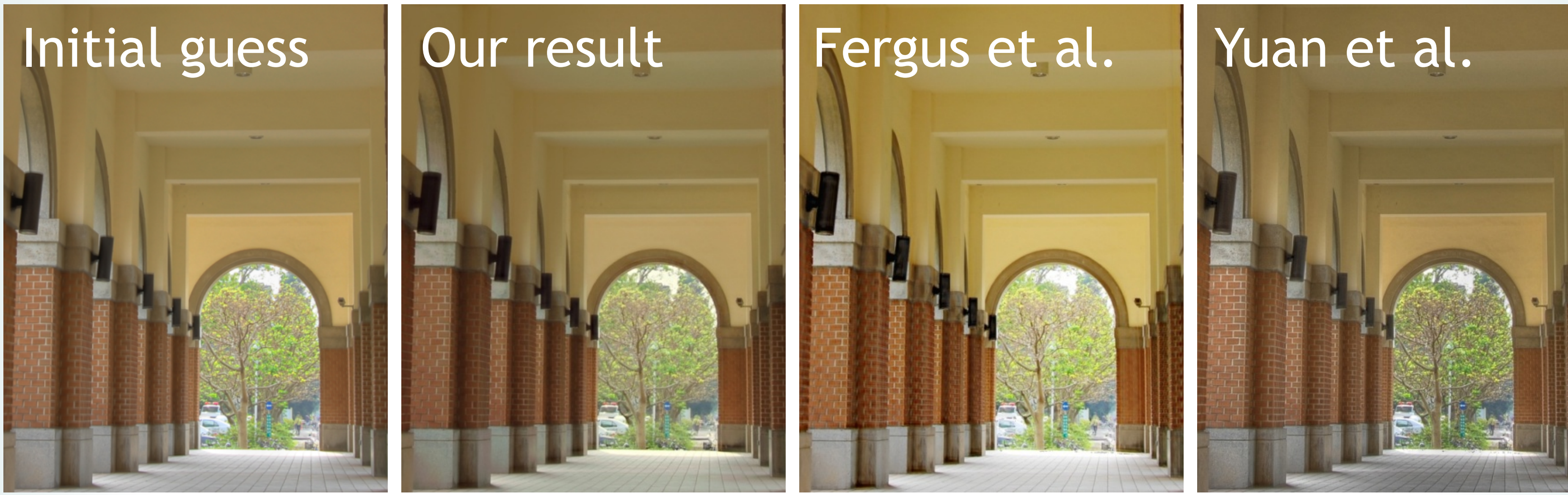
Fergus et al.



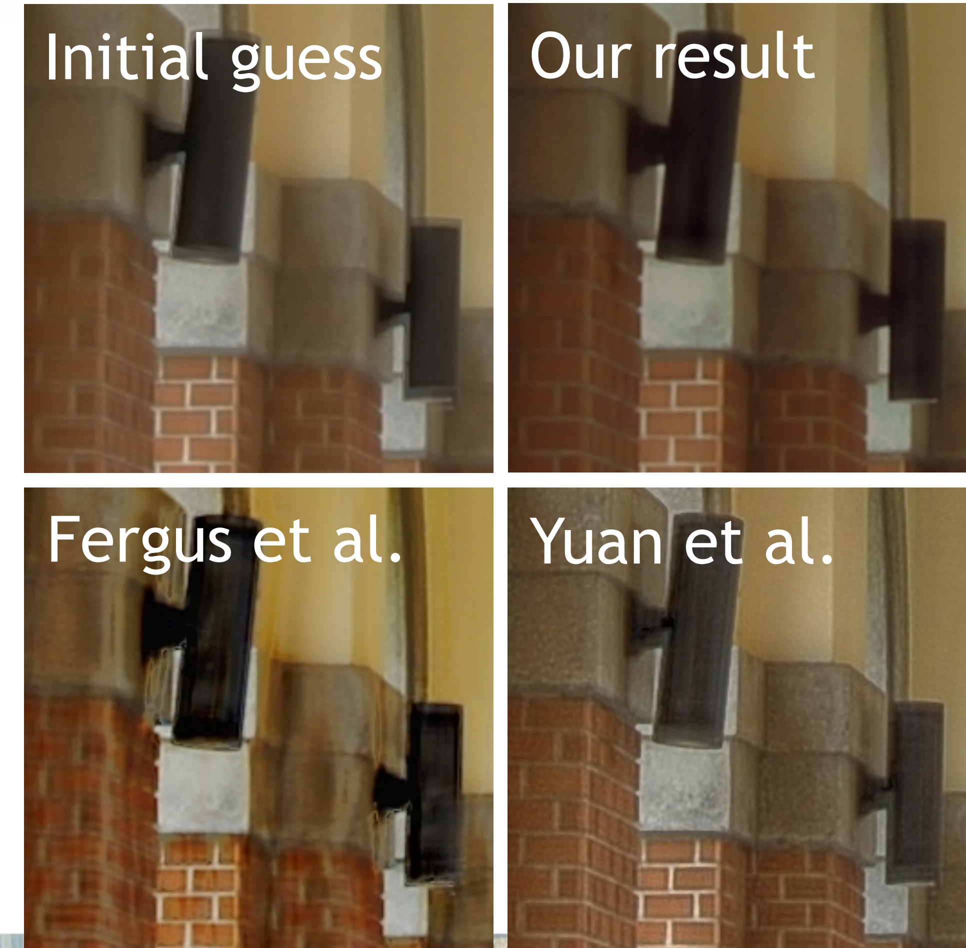
Real example II



Comparisons



Close-up comparisons



Initial guess

Our result

Yuan et al.

Fergus et al.

Conclusions

- A technique for reconstructing a non-blurred HDR image from a set of differently exposed and blurred images taken with a hand-held camera.
- A unified formulation for recovering the irradiance image, blur kernels and the camera response curve.

Future Work

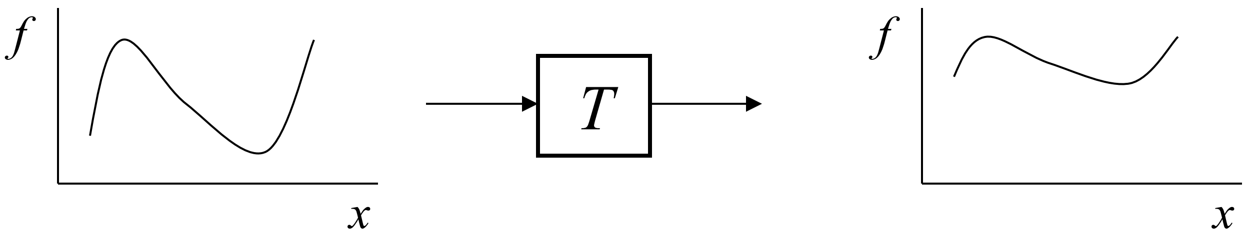
- Adding priors for kernels, HDR images and camera curves.
- Handling the shift-variant cases.

Next

Image Transformation

Image Warping

- image filtering: change ***range*** of image
  - *g(x) = T(f(x))*



- image warping: change ***domain*** of image
  - *g(x) = f(T(x))*

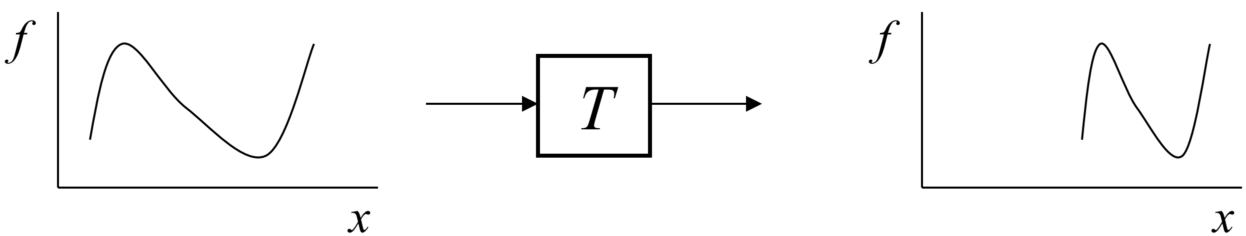
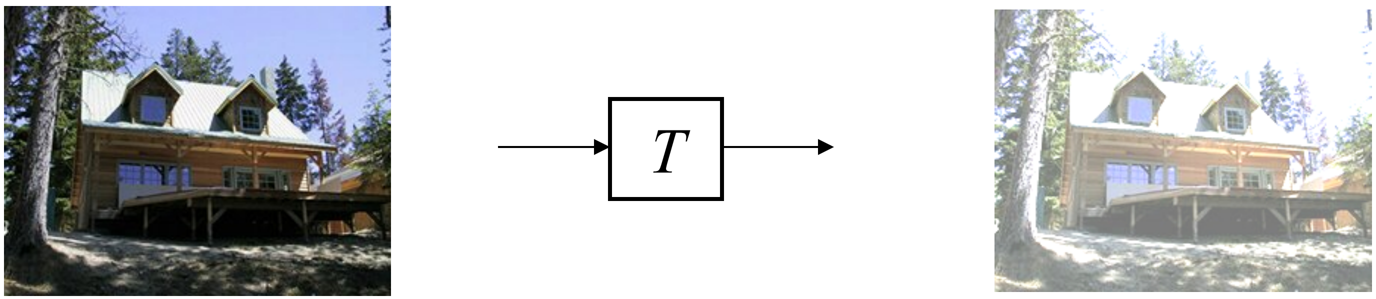
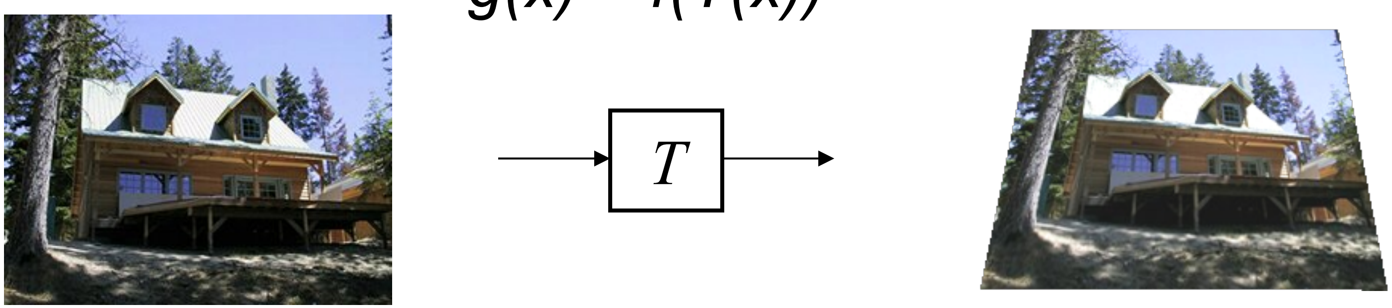


Image Warping

- image filtering: change ***range*** of image
  - *g(x) = T(f(x))*



- image warping: change ***domain*** of image
  - *g(x) = f(T(x))*



Parametric (global) warping

- Examples of parametric warps:

translation

rotation

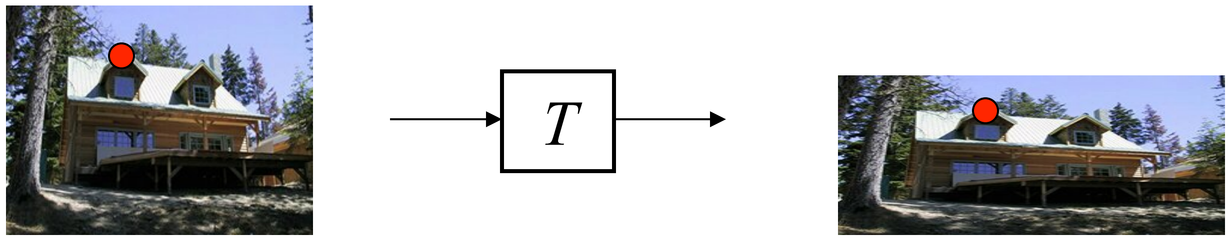
aspect

affine

perspective

cylindrical

Parametric (global) warping



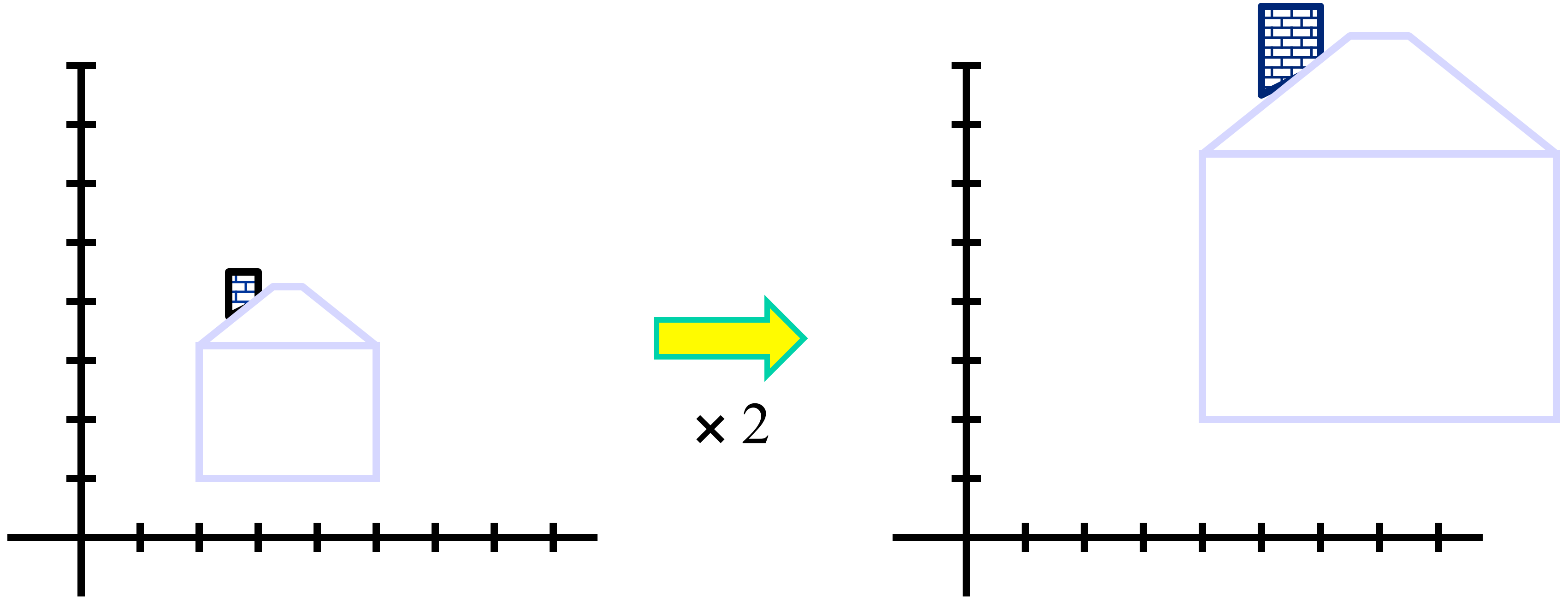
$\mathbf{p} = (x,y)$ $\mathbf{p'} = (x',y')$

- Transformation T is a coordinate-changing machine:
- $p' = T(p)$
- What does it mean that *T* is global?
  - Is the same for any point p
  - can be described by just a few numbers (parameters)
- Let's represent *T* as a matrix:
- $p' = \mathbf{M}p$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

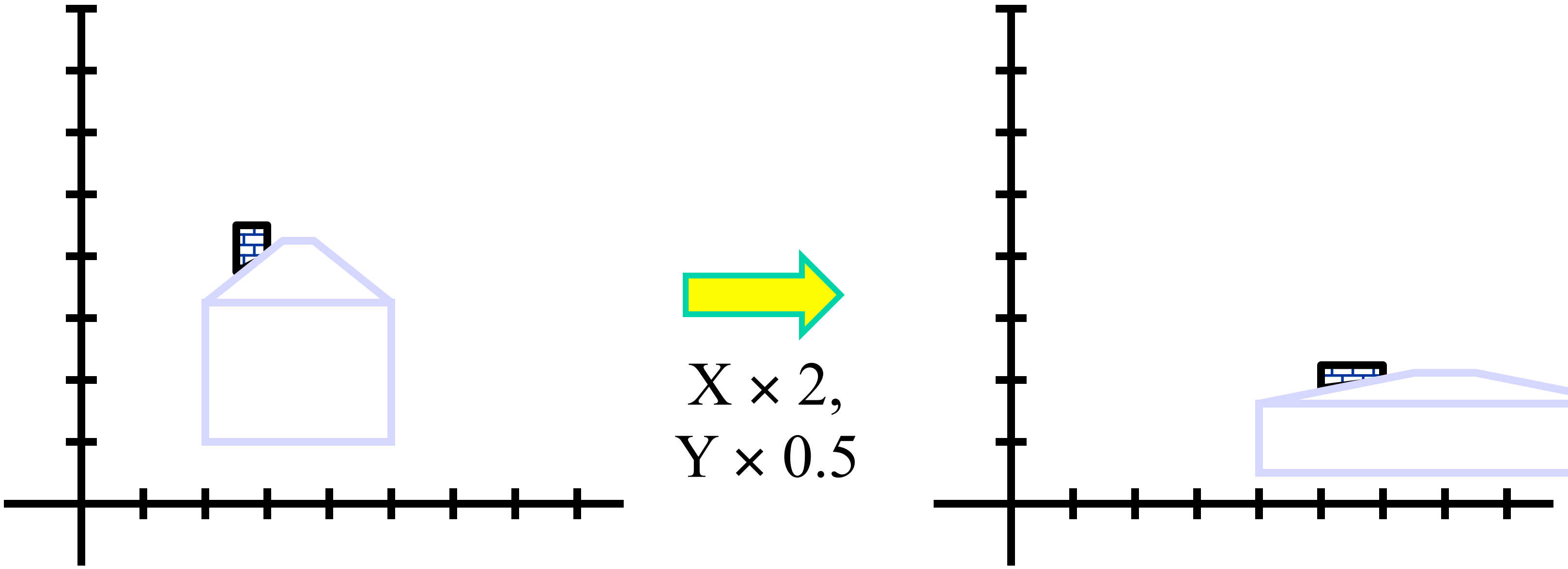
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



Scaling

- *Non-uniform scaling*: different scalars per component:



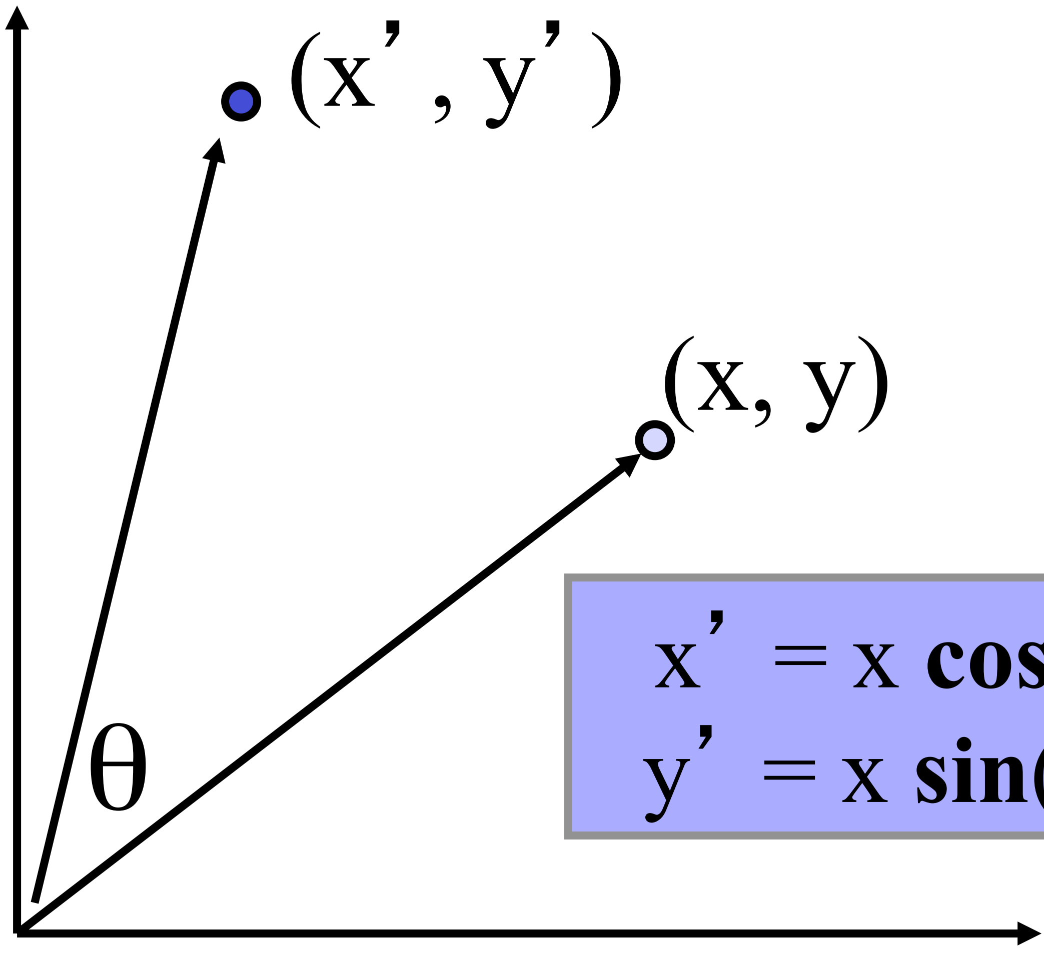
Scaling

- Scaling operation:
$$x' = ax$$
$$y' = by$$

- Or, in matrix form:
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

What's inverse of S?

2-D Rotation



$$x' = x\,\mathbf{cos}(\theta) - y\,\mathbf{sin}(\theta)$$
$$y' = x\,\mathbf{sin}(\theta) + y\,\mathbf{cos}(\theta)$$

2-D Rotation

- This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Even though sin(θ) and cos(θ) are nonlinear functions of θ,
  - ***x' is a linear combination of x and y***
  - ***y' is a linear combination of x and y***

- What is the inverse transformation?
  - Rotation by –θ
  - For rotation matrices

$$\mathbf{R}^{-1} = \mathbf{R}^T$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Identity?

$$x' = x \qquad y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Scale around (0,0)?

$$\boldsymbol{x}' = \boldsymbol{s}_x * \boldsymbol{x} \qquad \boldsymbol{y}' = \boldsymbol{s}_y * \boldsymbol{y}$$

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 \\ 0 & \boldsymbol{s}_y \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Rotate around (0,0)?

$$x' = \cos\Theta * x - \sin\Theta * y$$
$$y' = \sin\Theta * x + \cos\Theta * y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Shear?

$$x' = x + sh_x * y$$
$$y' = sh_y * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

$$x' = -x$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Mirror over (0,0)?

$$x' = -x$$
$$y' = -y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$x' = x + t_x$$
$$y' = y + t_y$$

NO!

Only linear 2D transformations can be represented with a 2x2 matrix

All 2D Linear Transformations

- Linear transformations are combinations of …
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Properties of linear transformations:
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Homogeneous Coordinates

- **Q: How can we represent translation as a 3x3 matrix?**

$$x' = x + t_x$$

$$y' = y + t_y$$

Homogeneous Coordinates

- ***Homogeneous coordinates***
  – represent coordinates in 2 dimensions with a 3-vector

$$\begin{bmatrix} x \\ y \end{bmatrix} \longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

- **Q: How can we represent translation as a 3x3 matrix?**

$$x' = x + t_x$$

$$y' = y + t_y$$

- A: Using the rightmost column:

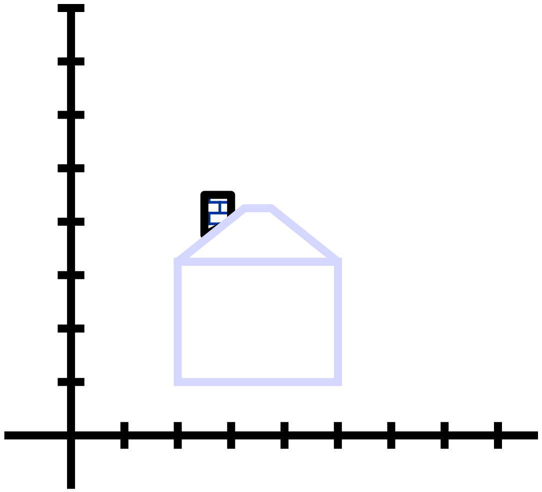
$$\mathbf{T}ranslation = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Translation

- Example of translation

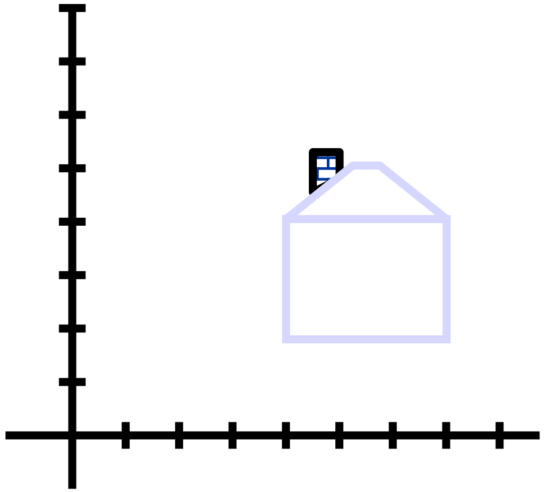
Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$



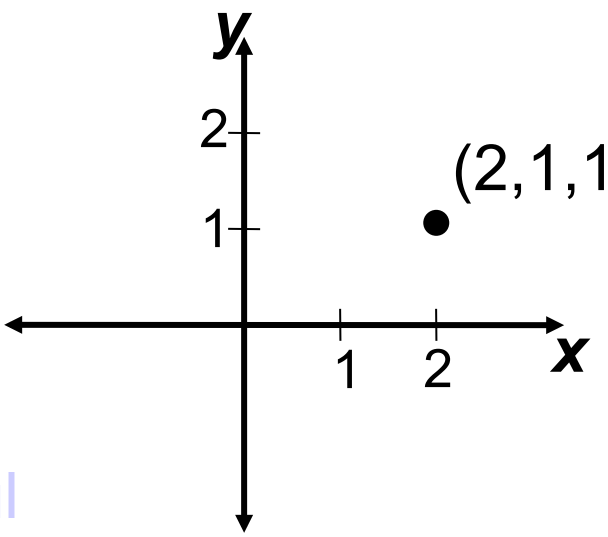
$t_x = 2$

$t_y = 1$



Homogeneous Coordinates

- Add a 3rd coordinate to every 2D point
  - (x, y, w) represents a point at location (x/w, y/w)
  - (x, y, 0) represents a point at infinity
  - (0, 0, 0) is not allowed



Convenient coordinate system to represent many useful transformations

Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_x} & 0 & 0 \\ 0 & \boldsymbol{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh_x} & 0 \\ \boldsymbol{sh_y} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

Affine Transformations

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

Projective Transformations

- Projective transformations …
  - Affine transformations, and
  - Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition

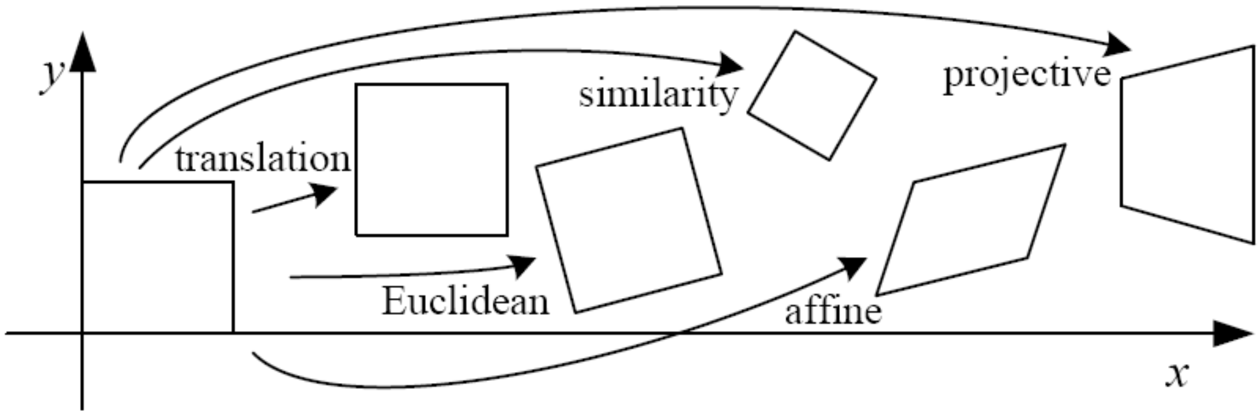
Matrix Composition

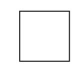
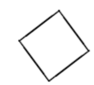
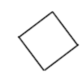
- Transformations can be combined by matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\mathbf{p}' \quad = \quad T(t_x,t_y) \qquad R(\Theta) \qquad S(s_x,s_y) \qquad \mathbf{p}$$

2D image transformations

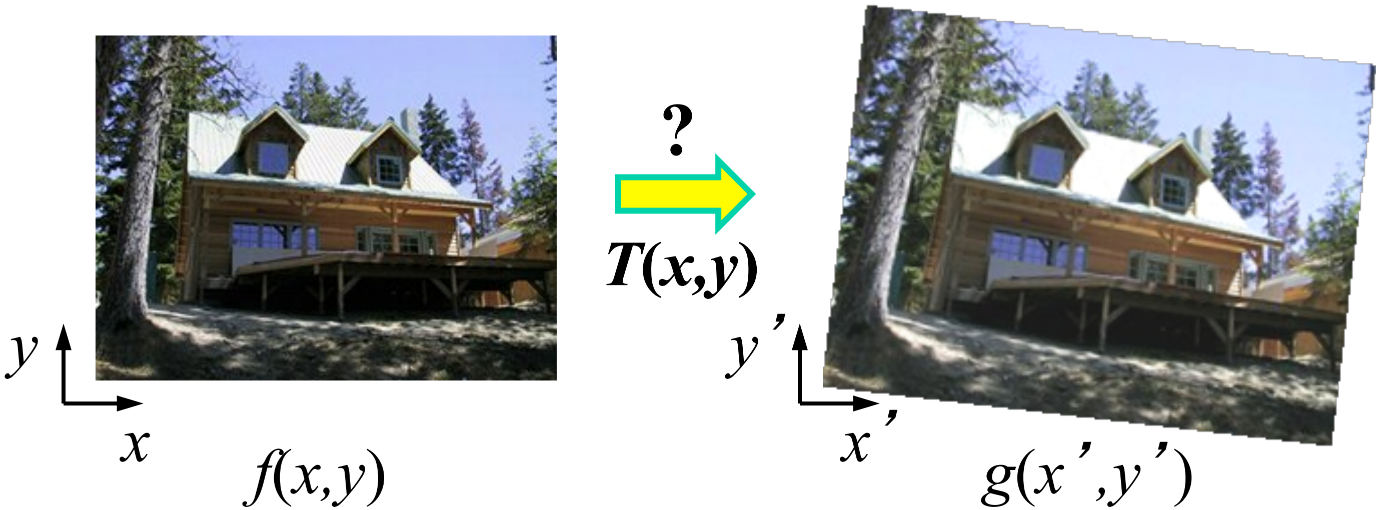


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$			
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$			
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$			
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$			
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$			

These transformations are a nested set of groups

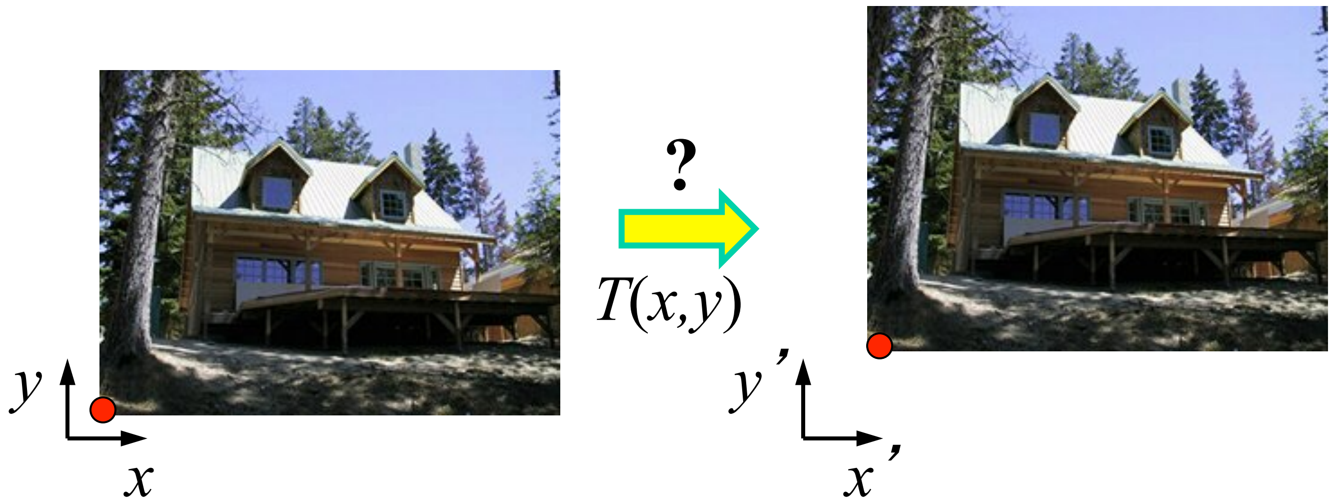
- Closed under composition and inverse is a member

Recovering Transformations



- What if we know *f* and *g* and want to recover the transform T?
  - Using correspondences
    - How many do we need?

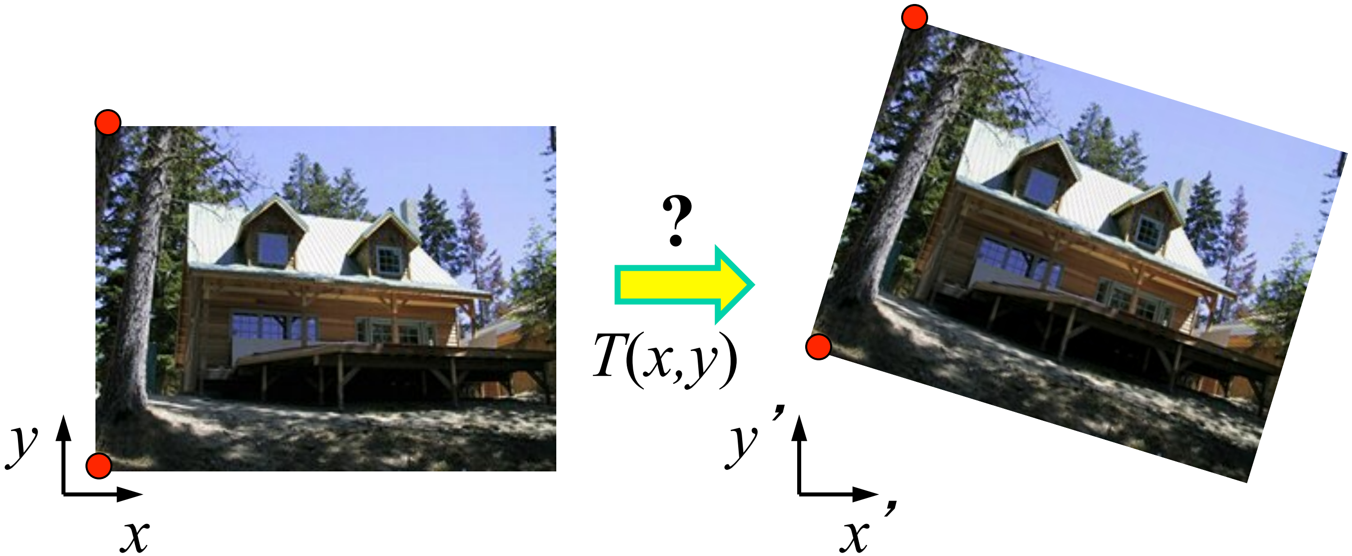
Translation: # correspondences?



- How many correspondences needed for translation?
- How many Degrees of Freedom?
- What is the transformation matrix?

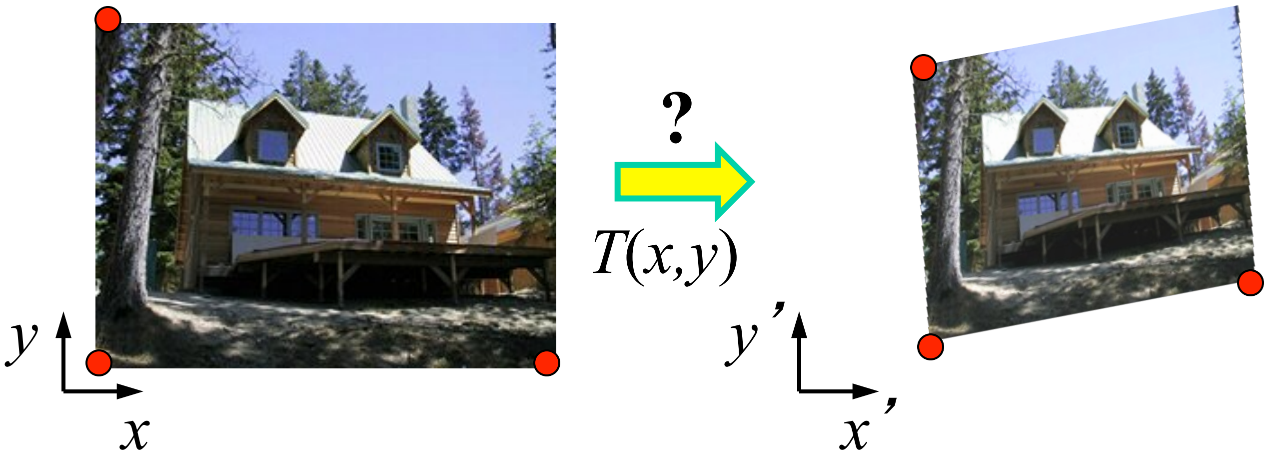
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Euclidian: # correspondences?



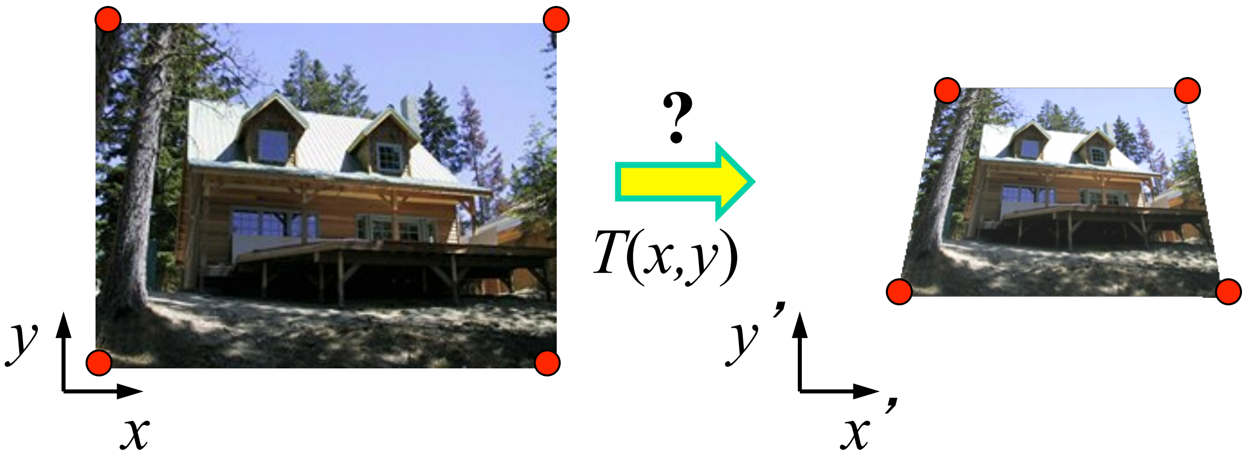
- How many correspondences needed for translation+rotation?
- How many DOF?

Affine: # correspondences?



- How many correspondences needed for affine?
- How many DOF?

Projective: # correspondences?



- How many correspondences needed for projective?
- How many DOF?