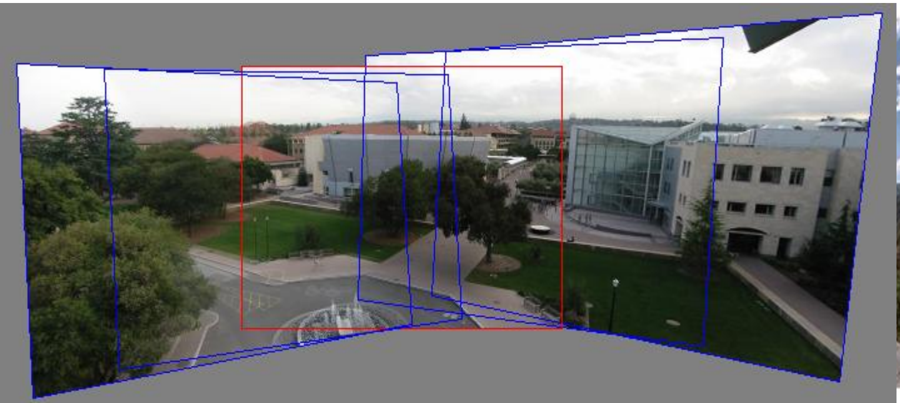
计算机视觉

Computer Vision

Lecture 8: Homography and Image Alignment

张 超

信息科学技术学院 智能科学系



Today

- Image Alignment
  - Fitting a 2D transformation
    - Affine, Homography
  - Computing an image mosaic
  - 2D image warping
  - Image Blending

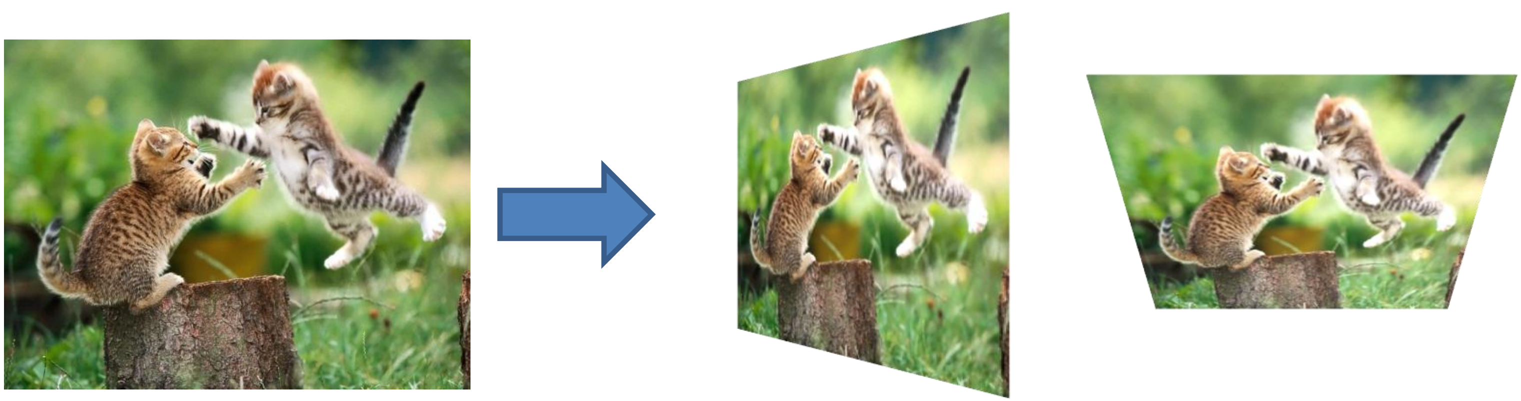
Planar Projective transformations

a.k.a. Homographies

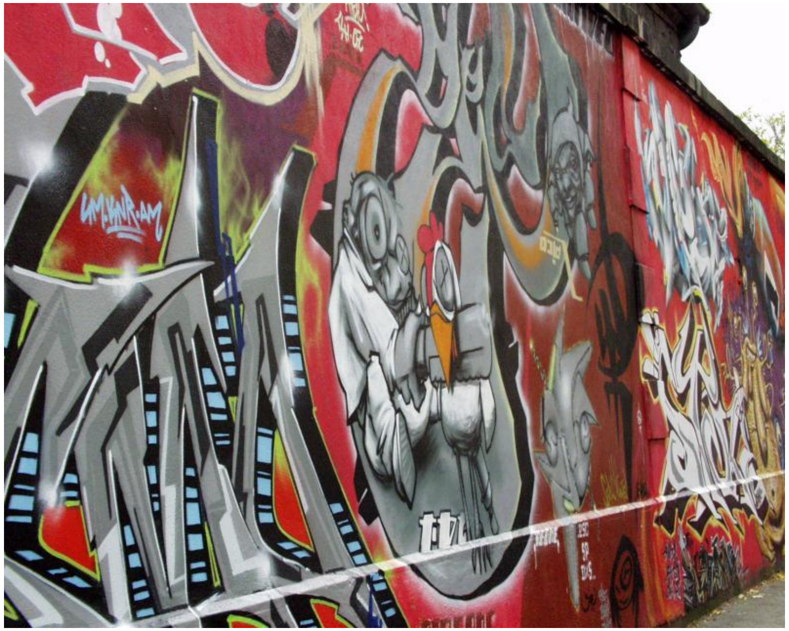
$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$x' = u/w$$
$$y' = v/w$$

"keystone" distortions

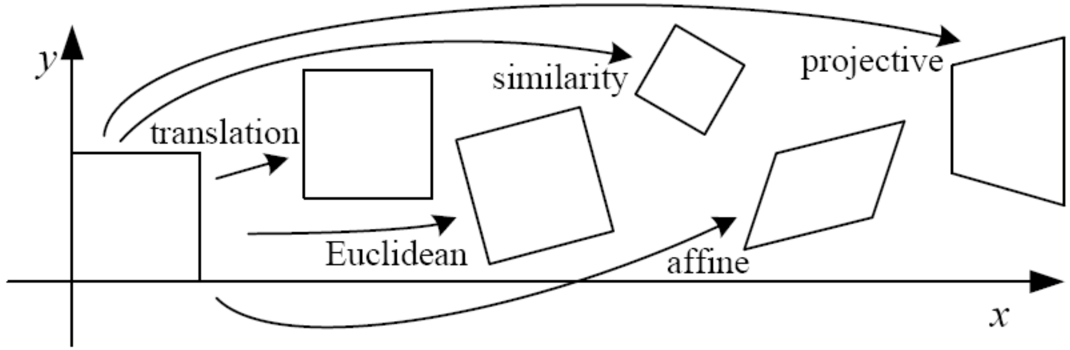


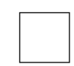
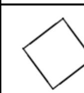
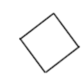
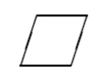
Finding the transformation



- How can we find the transformation between these images?
- How many corresponding points do we need to solve?

2D image transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

These transformations are a nested set of groups

- Closed under composition and inverse is a member

Finding the transformation

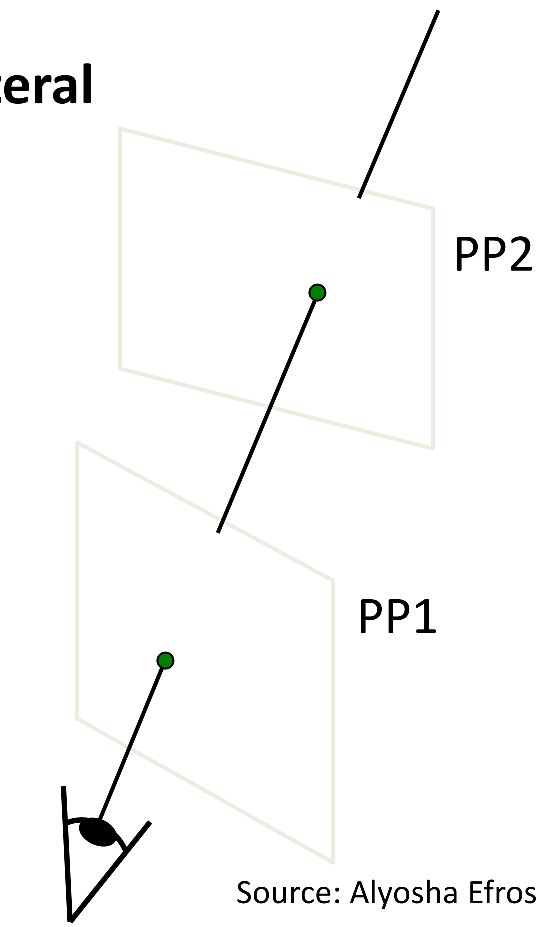
Translation	=	2 degrees of freedom
Similarity	=	4 degrees of freedom
Affine	=	6 degrees of freedom
Homography	=	8 degrees of freedom

How many corresponding points do we need to solve?

Homography

- **How to relate two images from the same camera center?**
  - **how to map a pixel from PP1 to PP2?**
- **Think of it as a 2D image warp from one image to another.**
- **A projective transform is a mapping between any two PPs with the same center of projection**
  - **rectangle should map to arbitrary quadrilateral**
  - **parallel lines aren't**
  - **but must preserve straight lines**
- **called Homography**

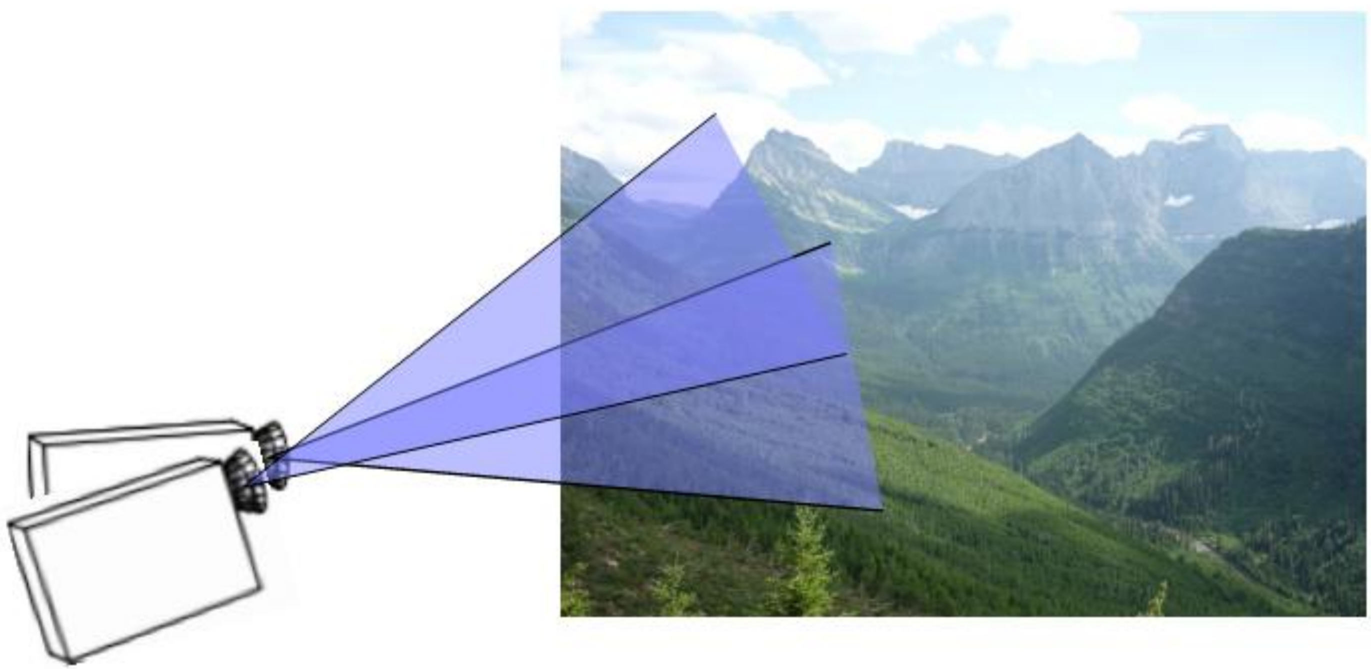
$$\underset{\mathbf{p'}}{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}} = \underset{\mathbf{H}}{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}} \underset{\mathbf{p}}{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$



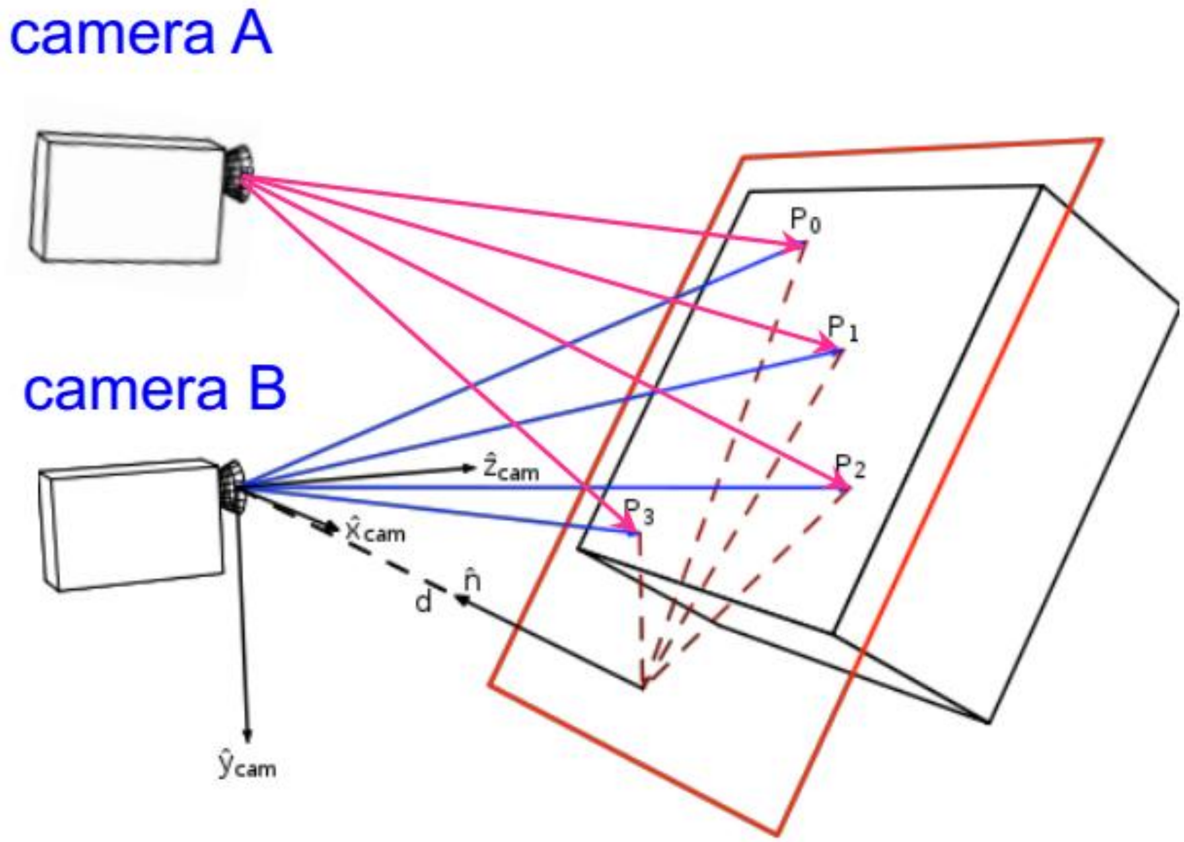
Source: Alyosha Efros

Homography

camera rotation

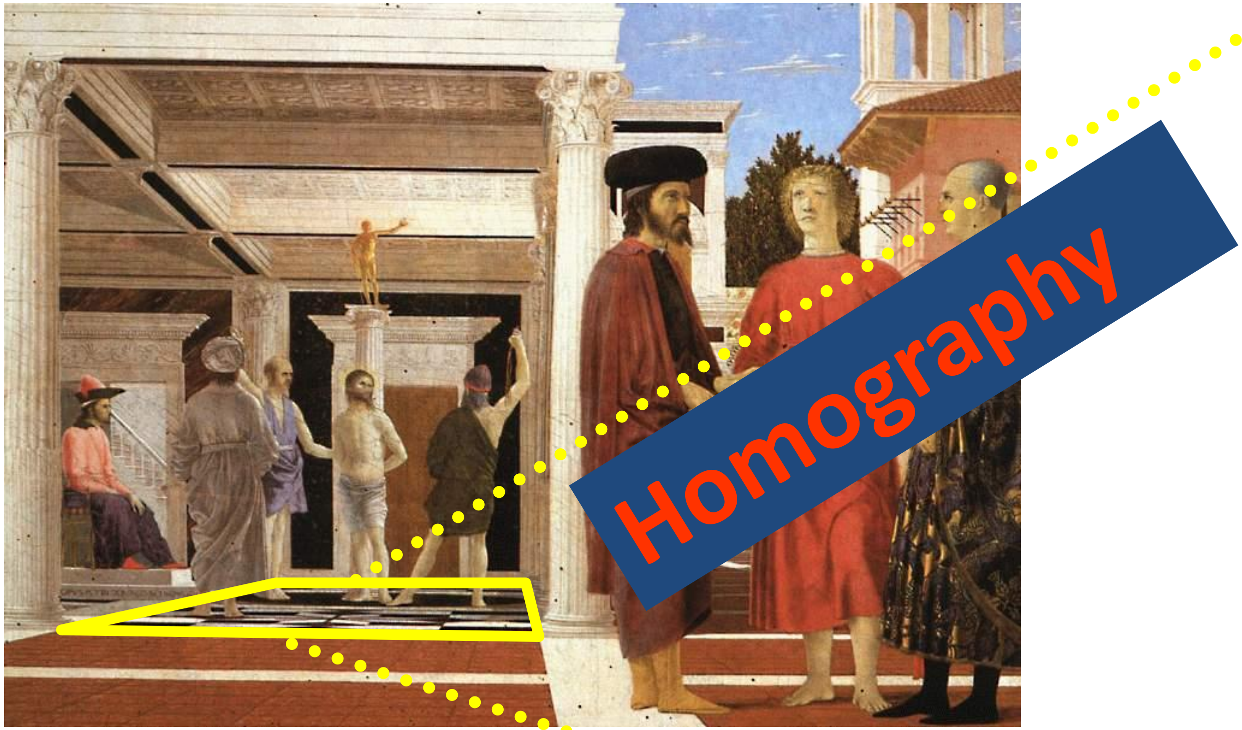


Homography



Analysing patterns and shapes

What is the shape of the b/w floor pattern?



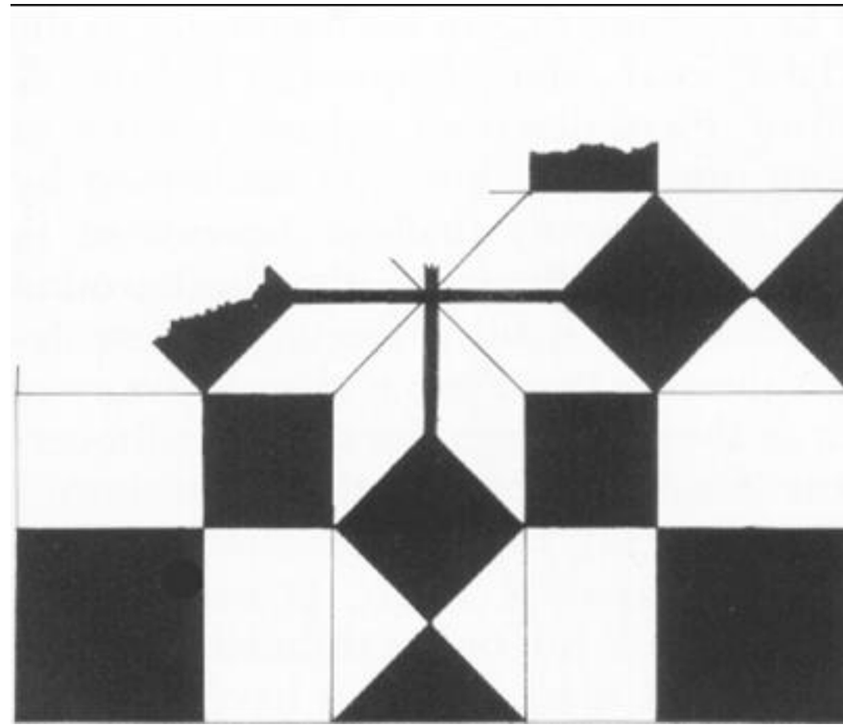
The floor (enlarged)

Automatically rectified floor

Slide from Criminisi

Analysing patterns and shapes

Automatic rectification



From Martin Kemp *The Science of Art*
(manual reconstruction)

Slide from Criminisi

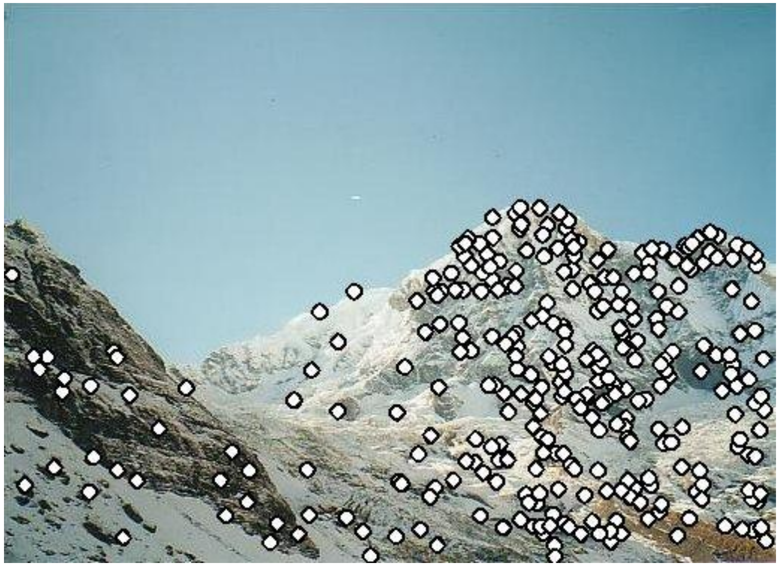
Image mosaics

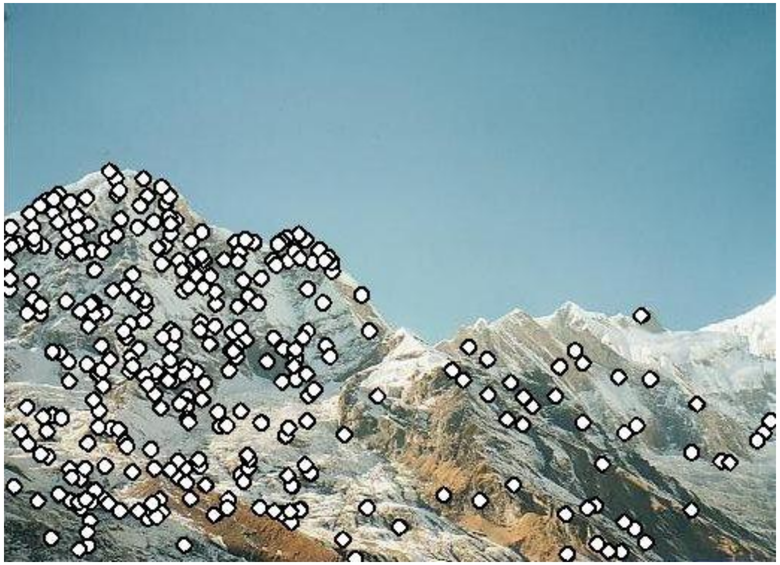
How do we build panorama?

- **We need to match (align) images**

Matching with Features

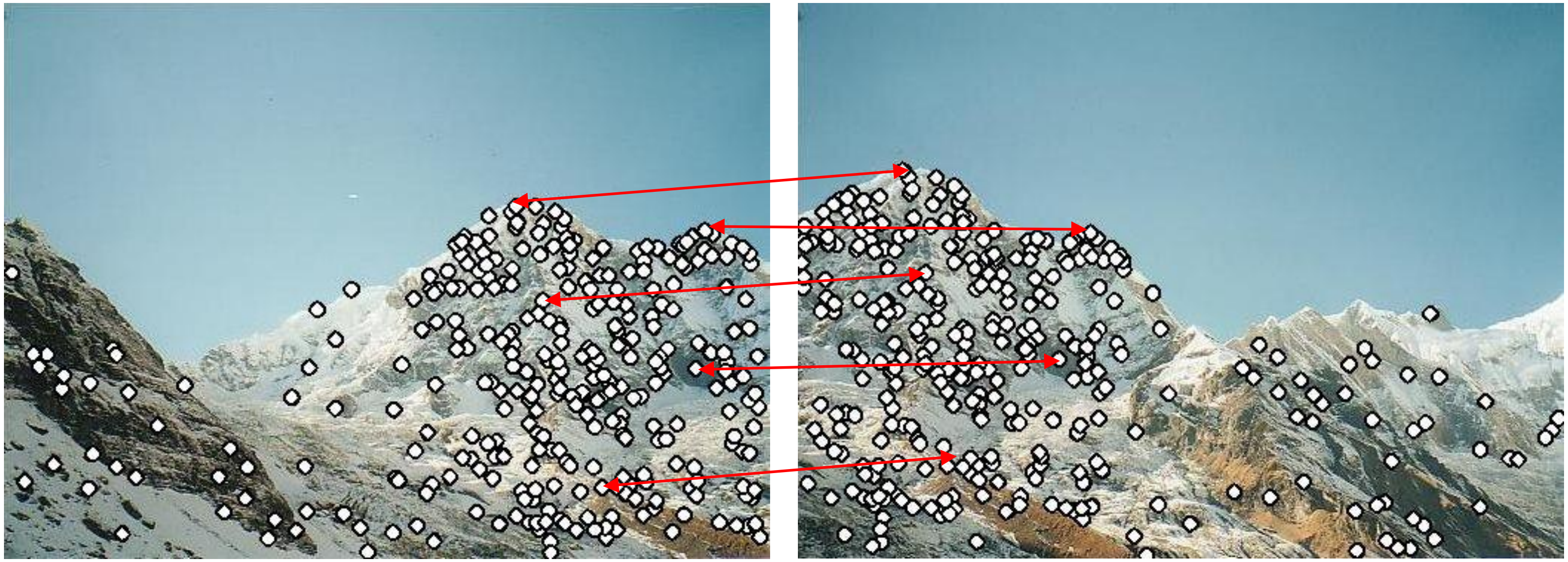
- Detect feature points in both images





Matching with Features

- Detect feature points in both images
- Find corresponding pairs



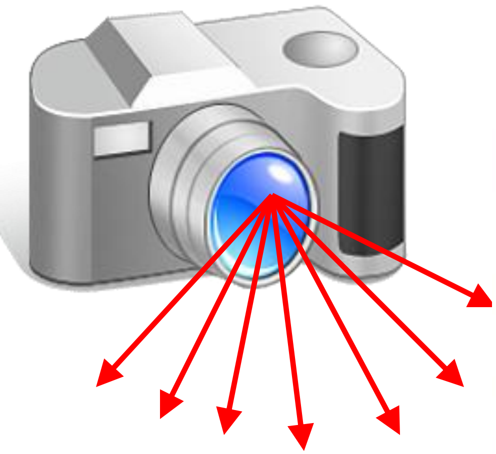
Aligning the Images

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Building panorama

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Mosaics



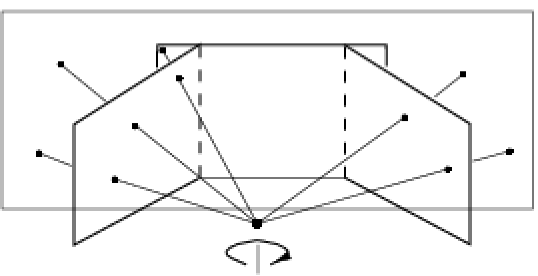


image from S. Seitz

Obtain a wider angle view by combining multiple images.

How to stitch together a panorama (a.k.a. mosaic)?

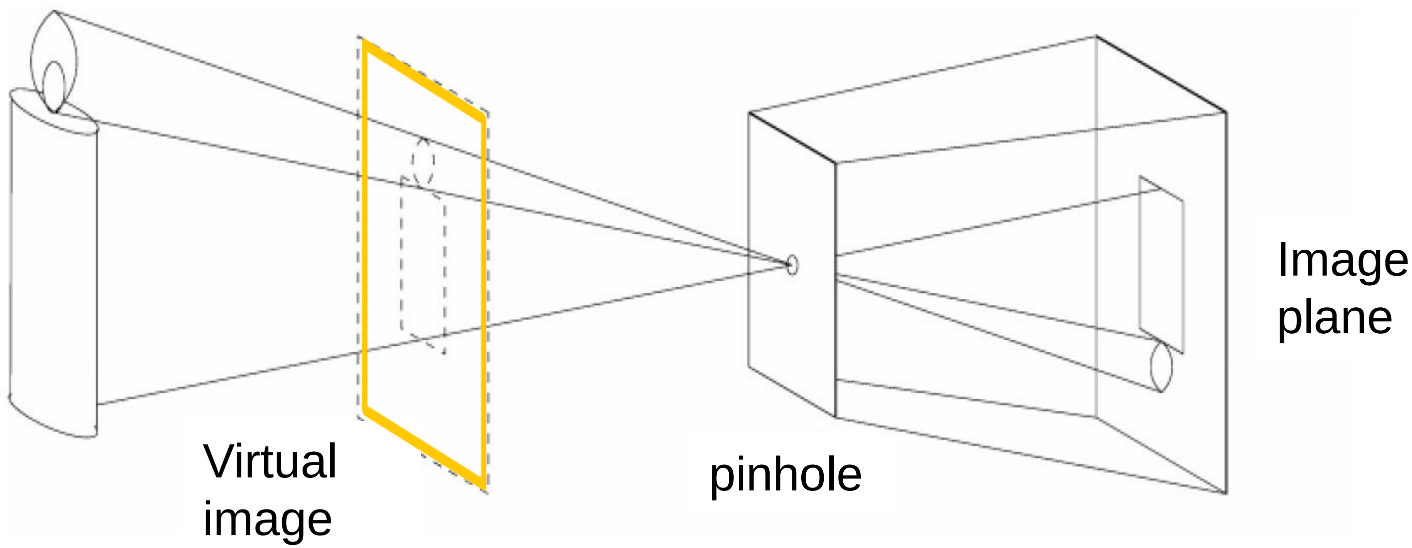
- **Basic Procedure**
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - (If there are more images, repeat)

- …but wait, why should this work at all?
  - What about the 3D geometry of the scene?
  - Why aren’t we using it?

Source: Steve Seitz

Pinhole camera

- **Pinhole camera is a simple model to approximate imaging process, perspective projection.**



If we treat pinhole as a point, only one ray from any given point can enter the camera.

Fig from Forsyth and Ponce

Mosaics

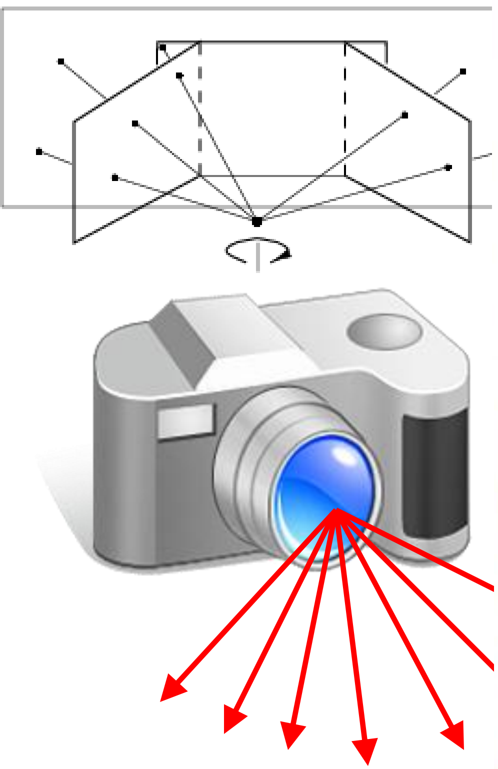
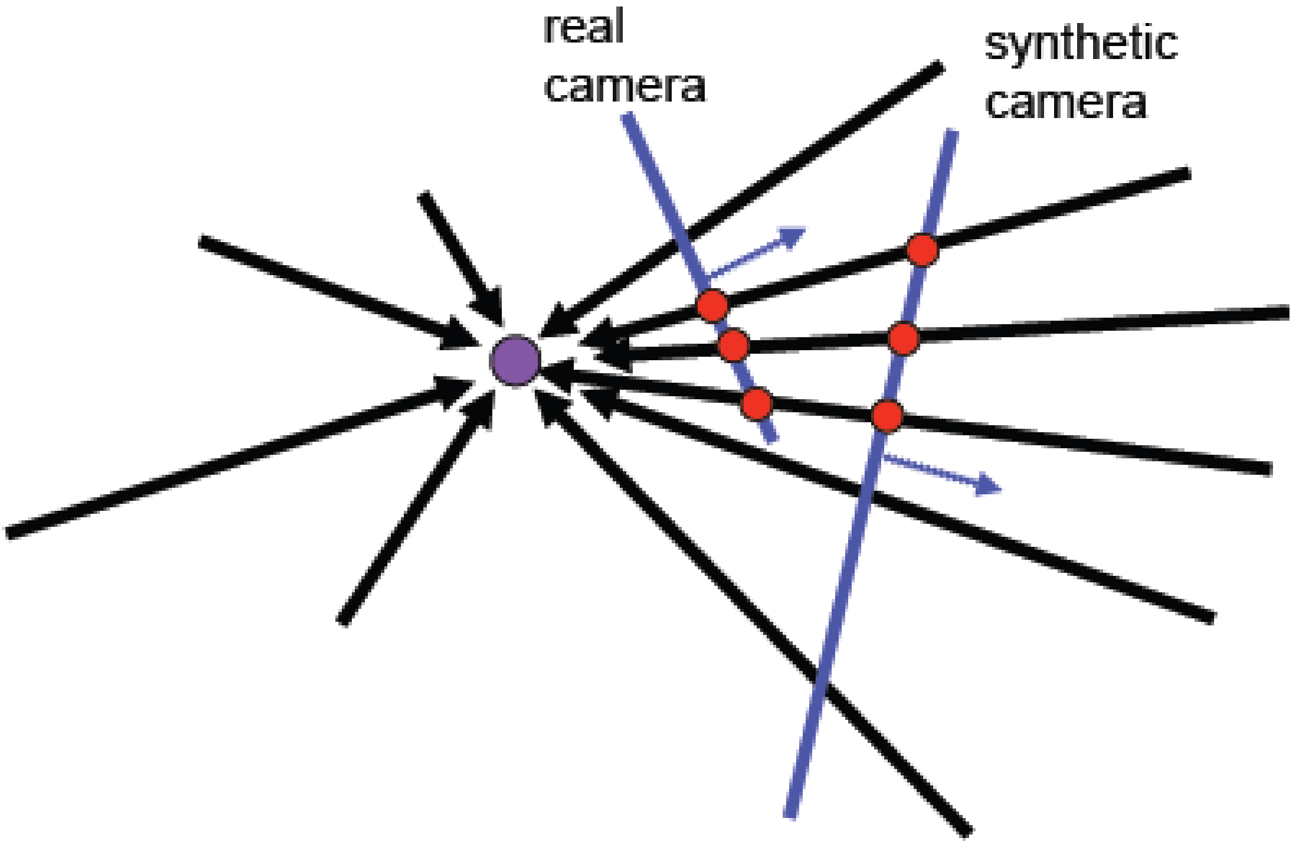


image from S. Seitz

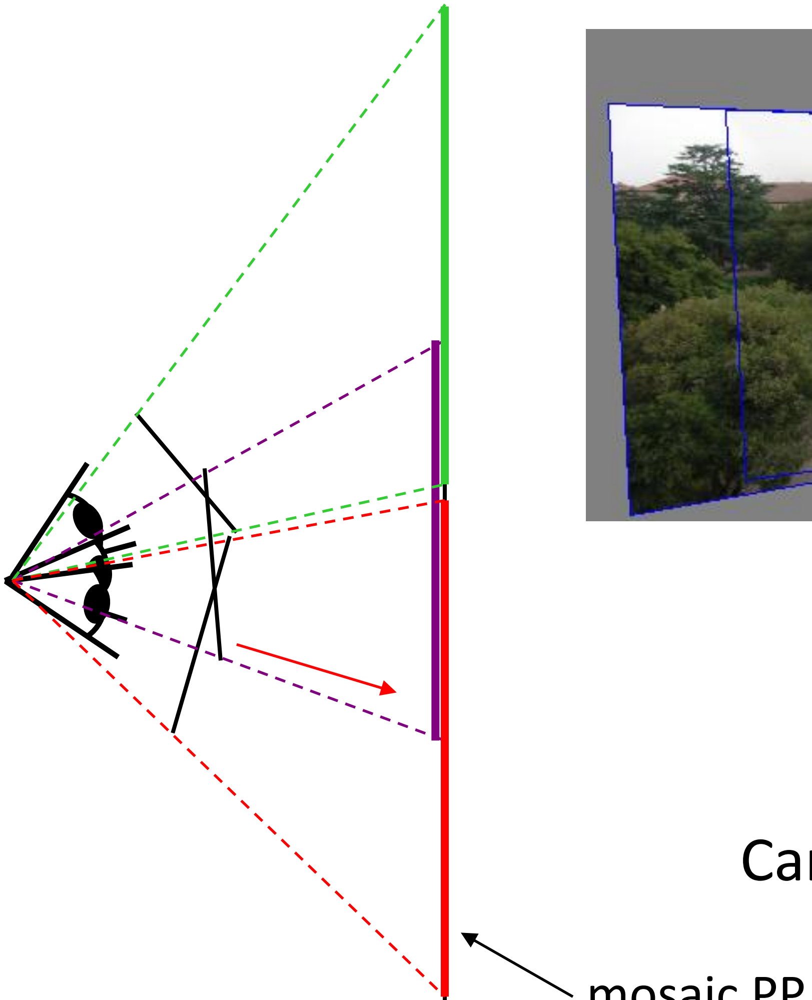
Obtain a wider angle view by combining multiple images.

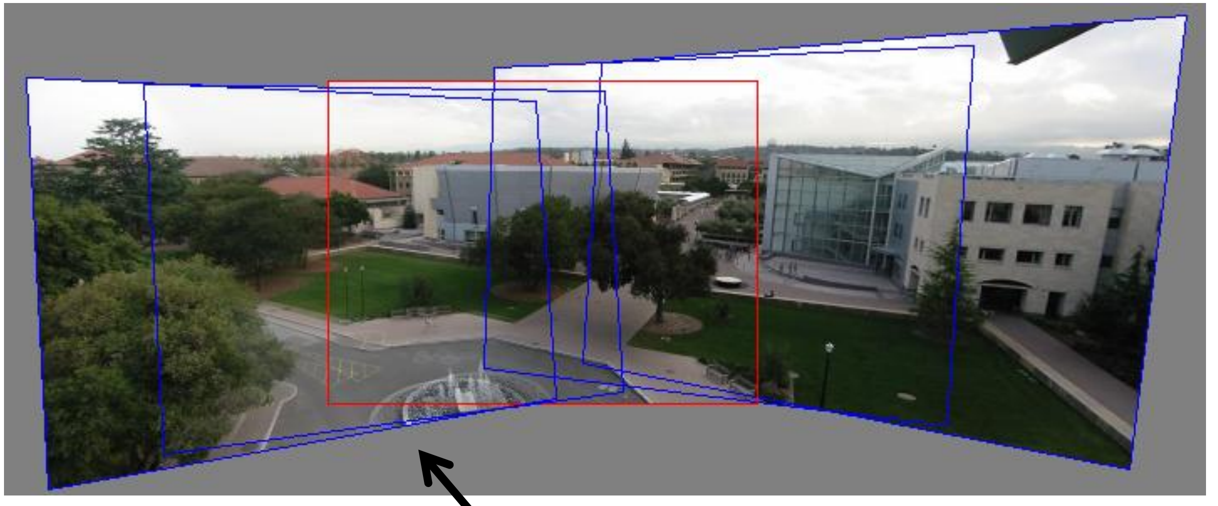
A pencil of rays contains all views



Can generate any synthetic camera view
as long as it has **the same center of projection**!

Projecting images onto a common plane



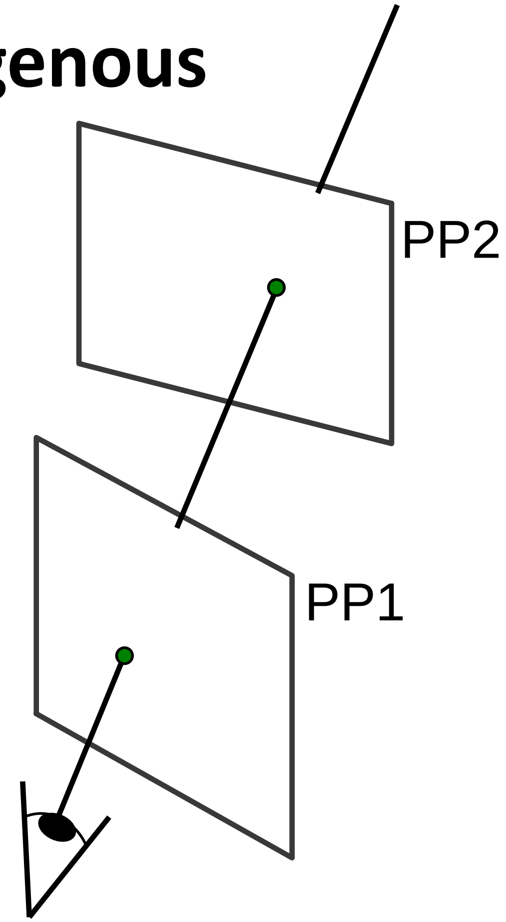


each image is warped with a homography $\mathbf{H}$

Can't create a 360 panorama this way…

Homography

- **Projective – mapping between any two projection planes with the same center of projection**
- **represented as 3x3 matrix in homogenous coordinates**



$$\underset{\mathbf{p}'}{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}} = \underset{\mathbf{H}}{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}} \underset{\mathbf{p}}{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$

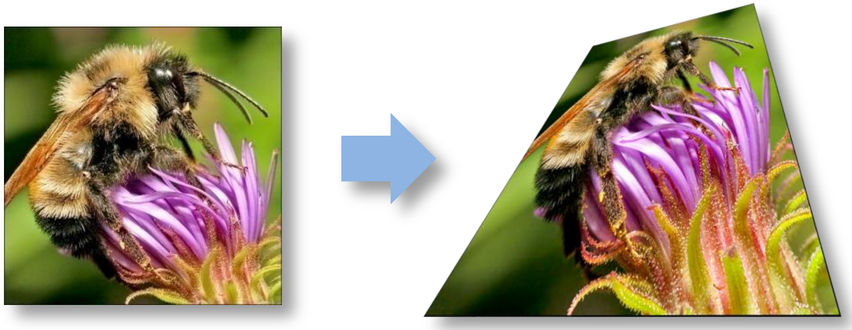
To apply a homography **H**

- Compute **p'** = **Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates (divide by w)

Projective Transformations aka Homographies aka Planar Perspective Maps

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)



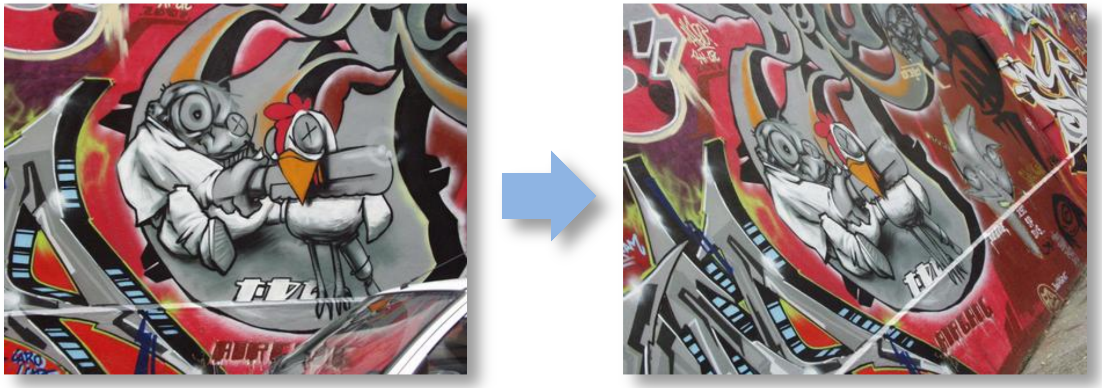
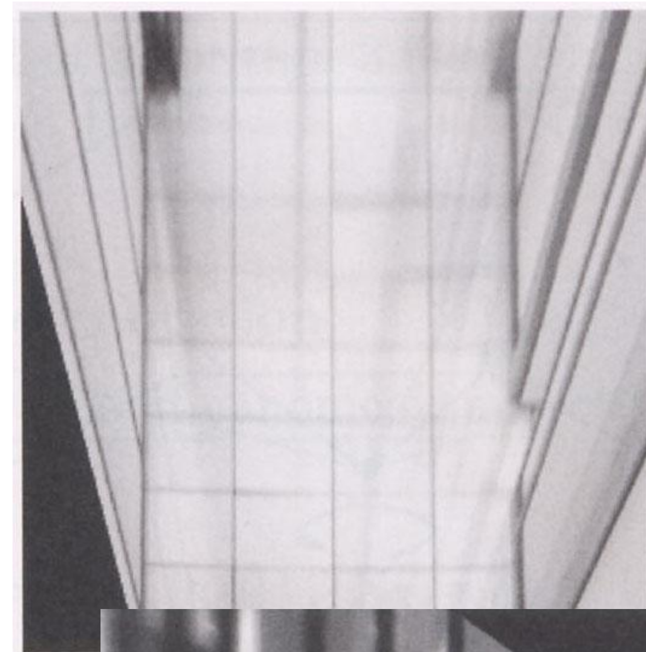


Image warping with homographies



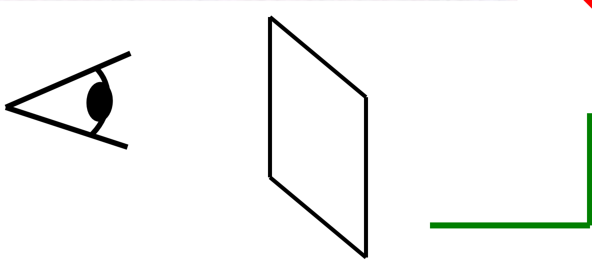
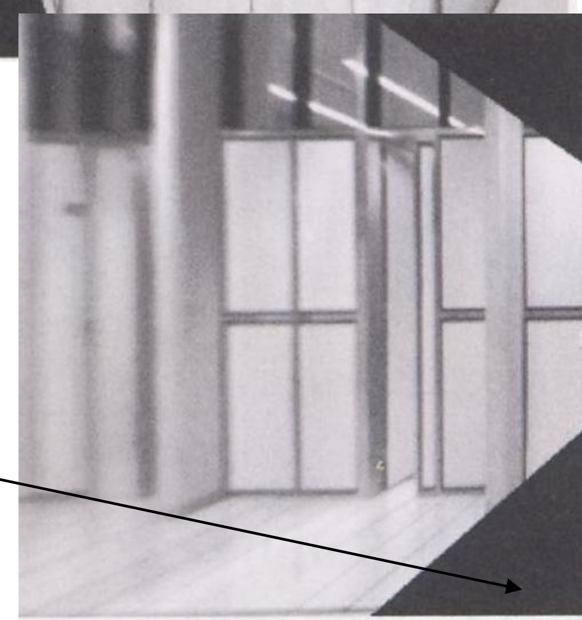


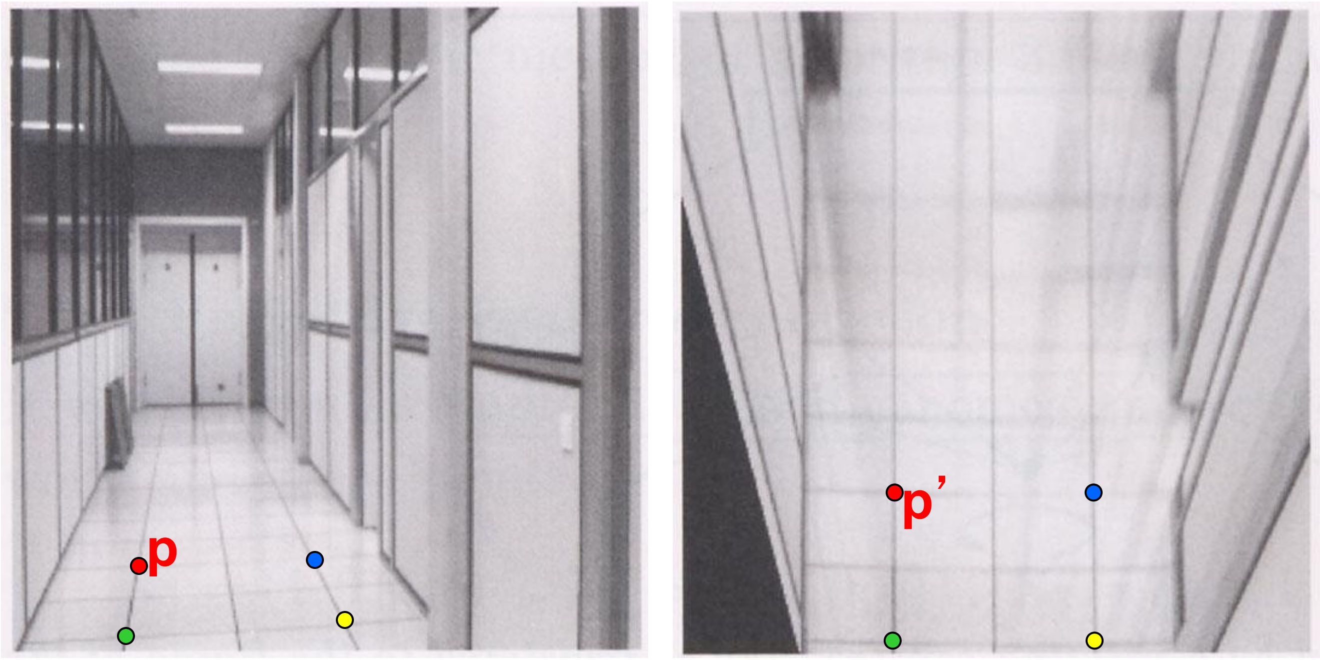
image plane in front



black area
where no pixel
maps to

Source: Steve Seitz

Image rectification



To unwarp (rectify) an image

- solve for homography **H** given **p** and **p**'
- solve equations of the form: w**p**' = **Hp**

– linear in unknowns: w and coefficients of **H**

– H is defined up to an arbitrary scale factor

Solving for homographies

$$\begin{bmatrix} w x_i' \\ w y_i' \\ w \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{aligned} x_i' &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y_i' &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

$$
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\
 & & & & \vdots & & & & \\
x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\
0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n
\end{bmatrix}
\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}
$$

A **h** **0**

2n × 9 **9** **2n**

Defines a least squares problem:

$$\text{minimize } \|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$$

- Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

Fun with homographies

Original image

St.Petersburg
photo by A. Tikhonov

Virtual camera rotations

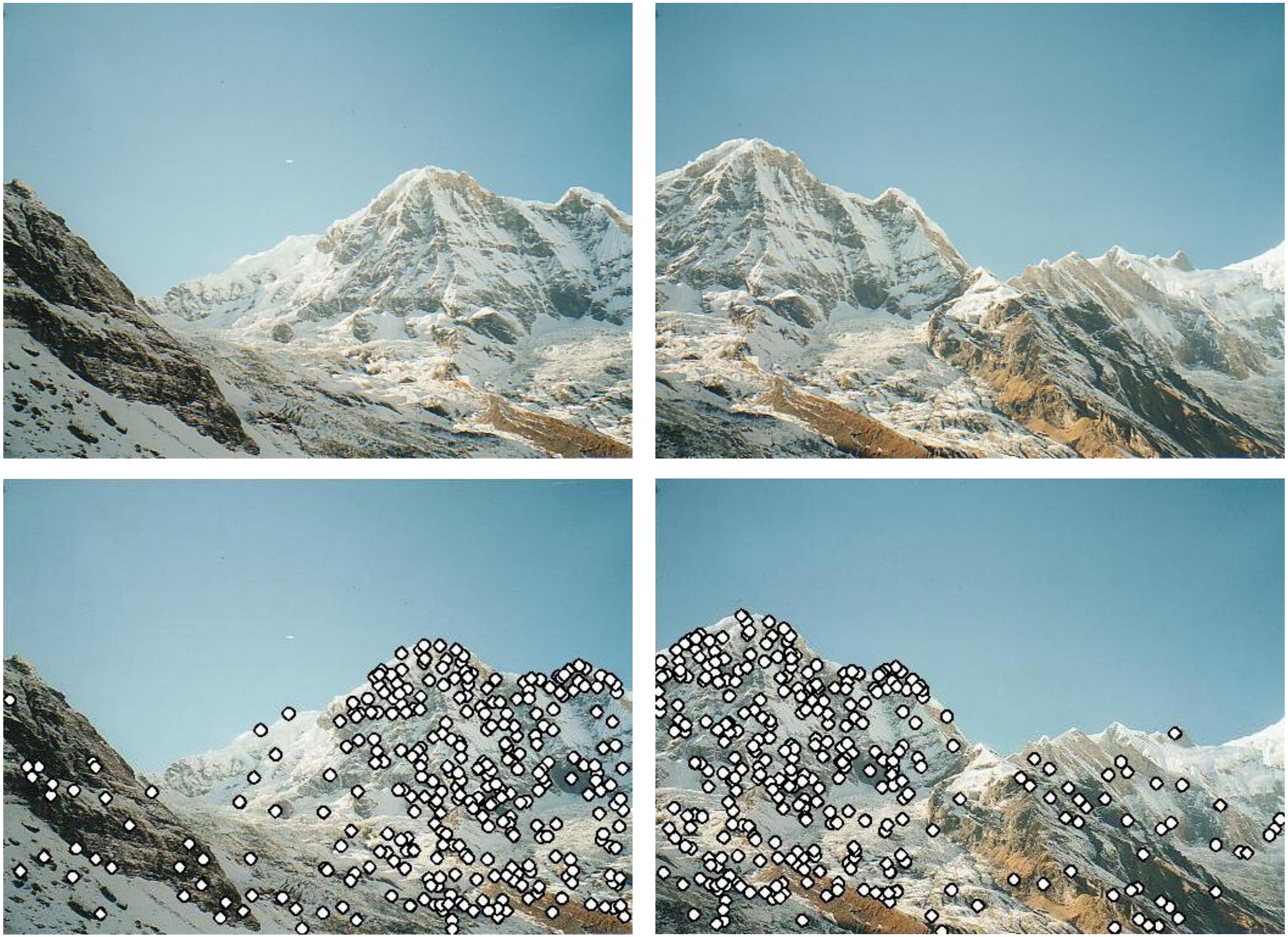
Image Alignment Algorithm

Given images A and B

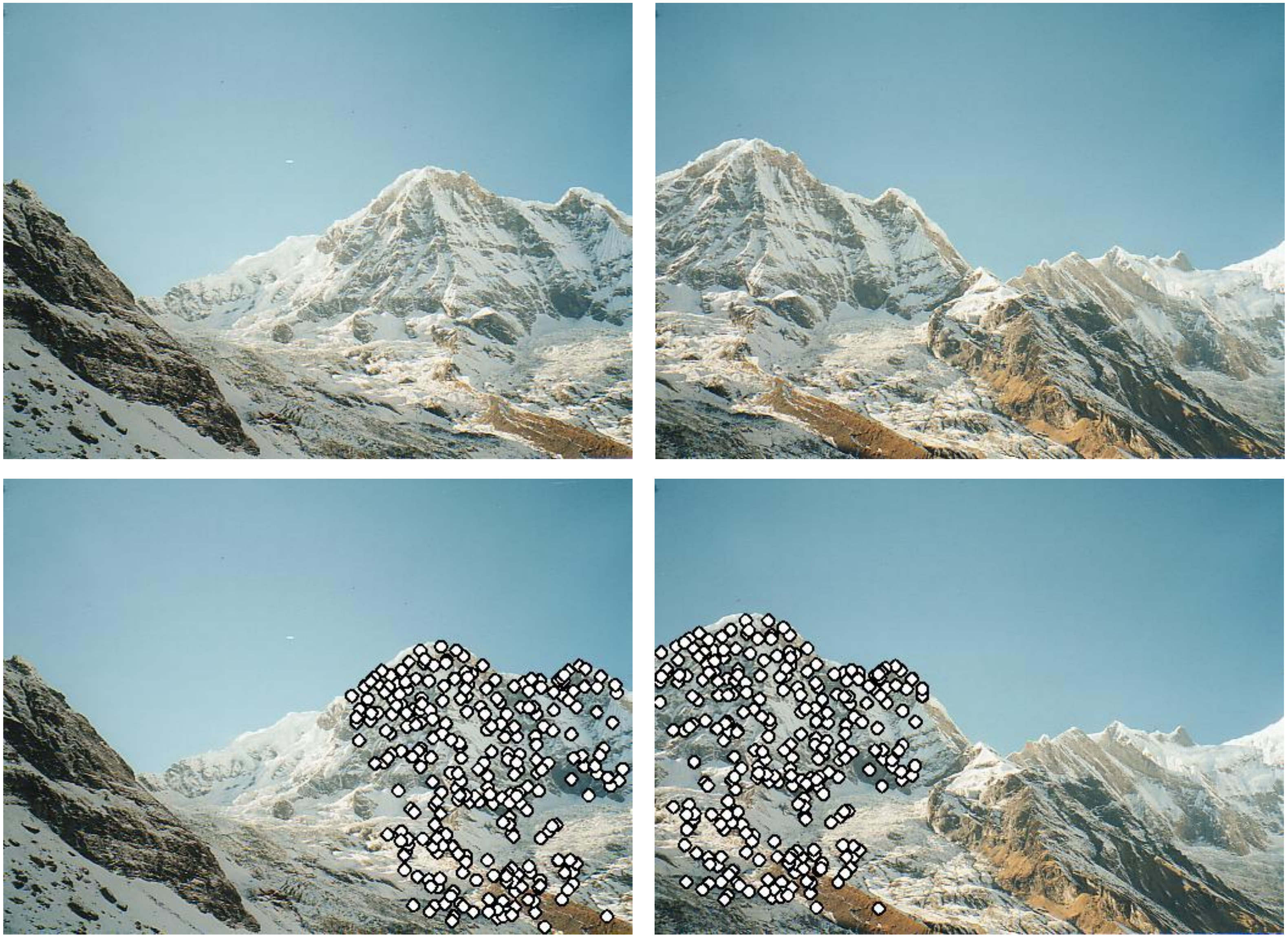
1. **Compute image features for A and B**
2. **Match features between A and B**
3. **Compute homography between A and B using least squares on set of matches**

What could go wrong?

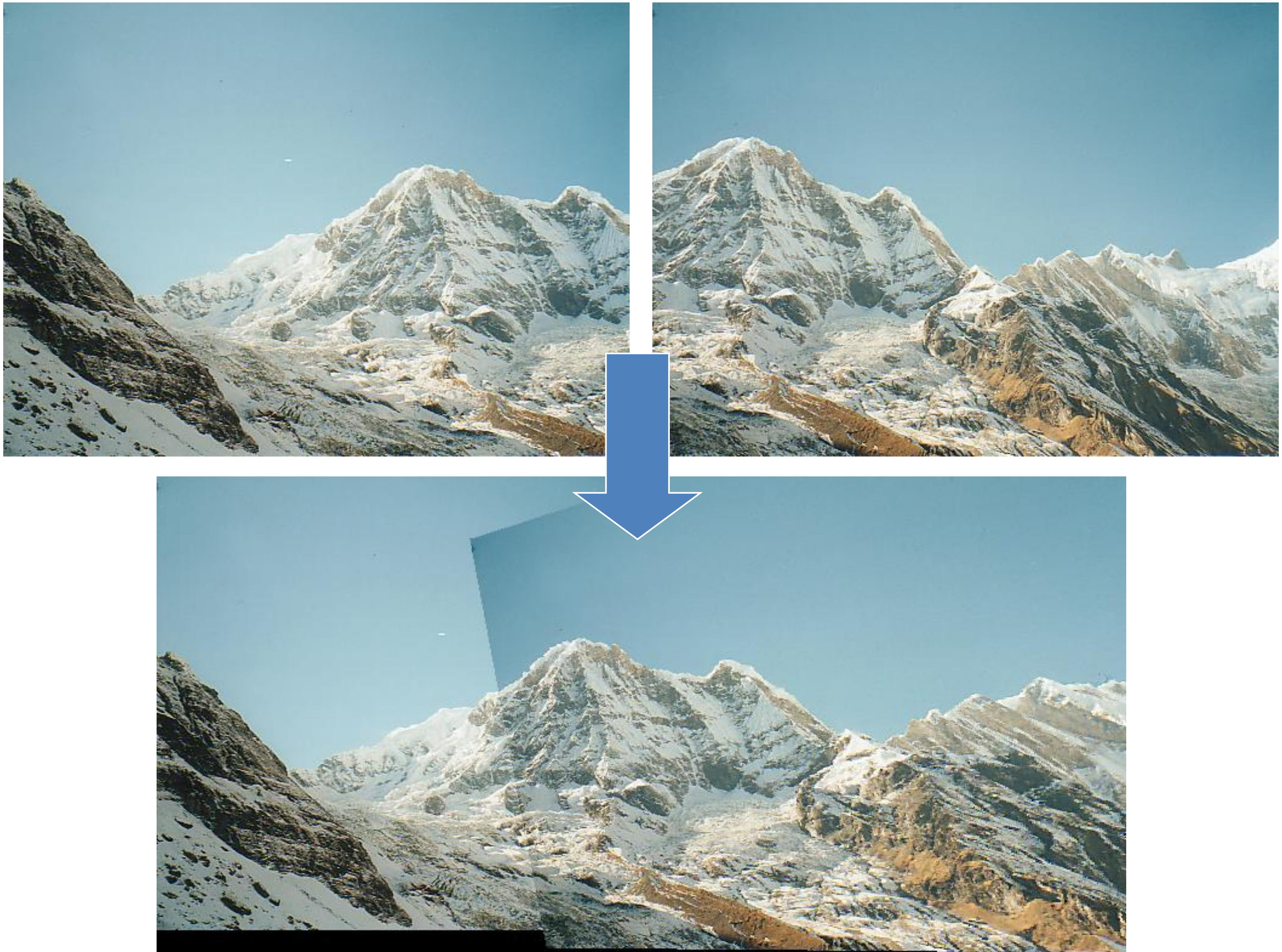
RANSAC for Homography



RANSAC for Homography



RANSAC for Homography



Probabilistic model for verification

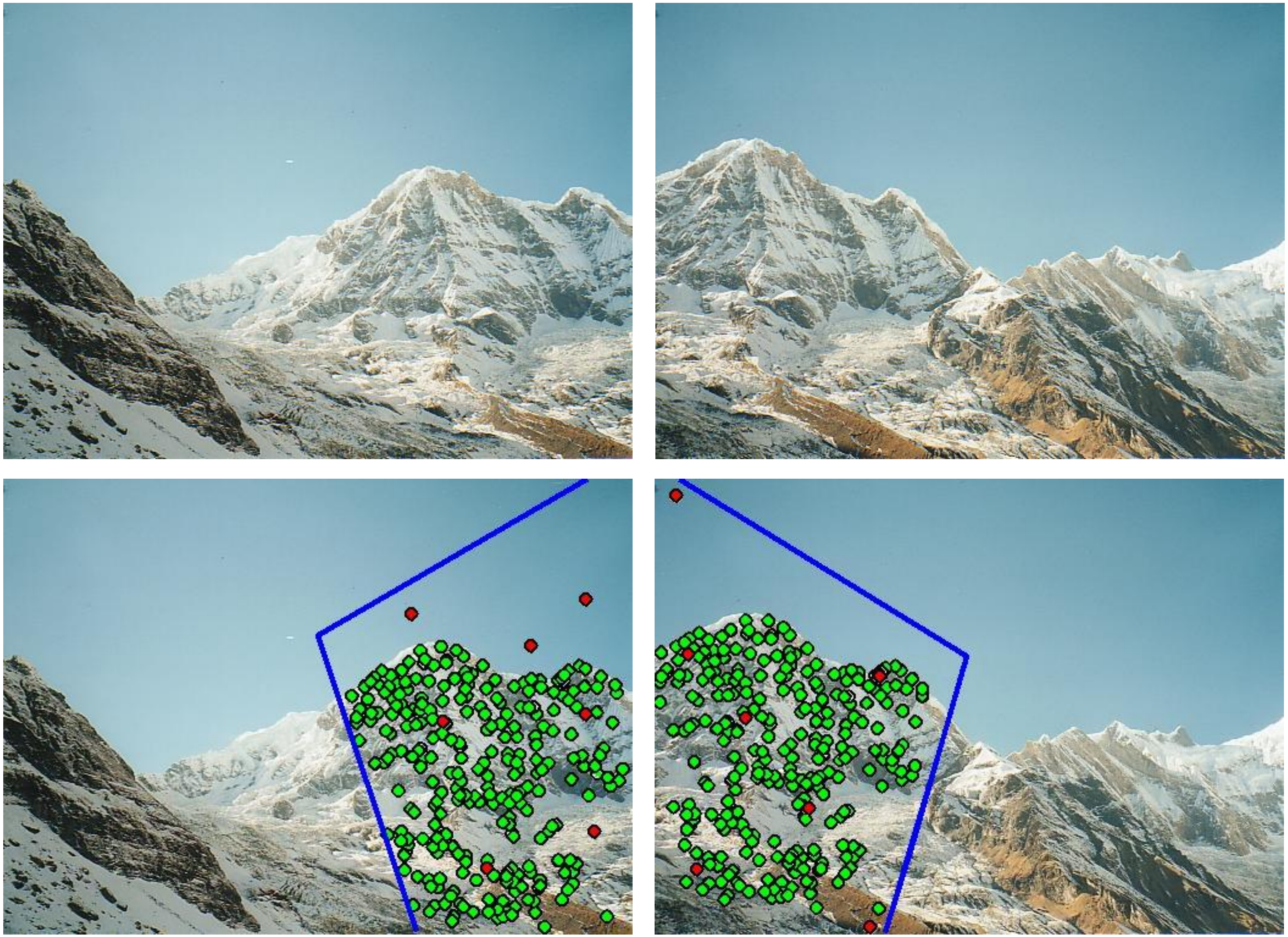
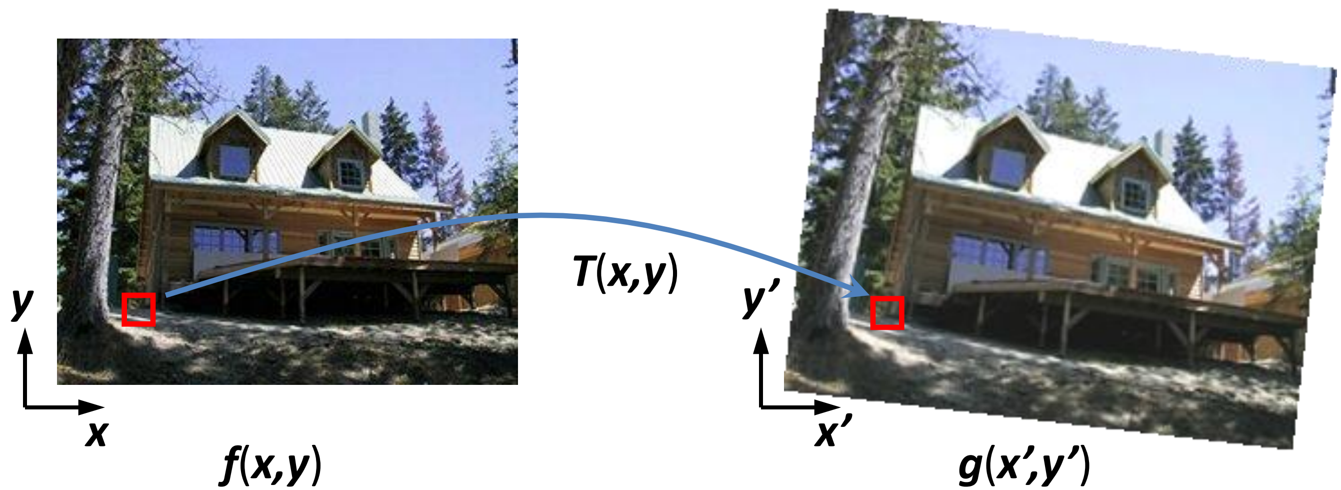


Image Warp

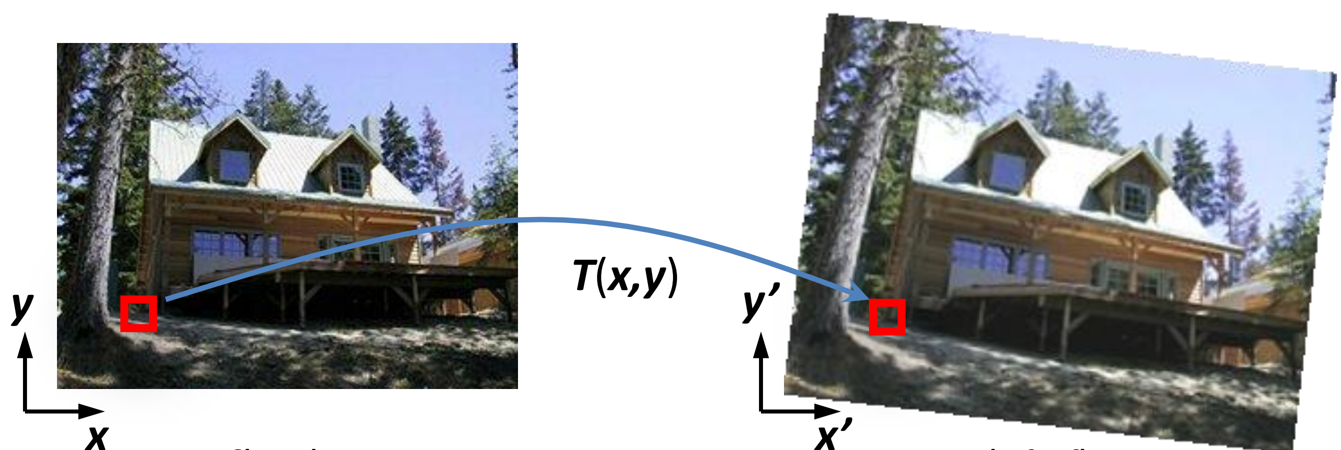
Image Warping

- **Given a coordinate xform $(x', y') = T(x, y)$ and a source image $f(x, y)$, how do we compute an xformed image $g(x', y') = f(T(x, y))$?**



Forward Warping

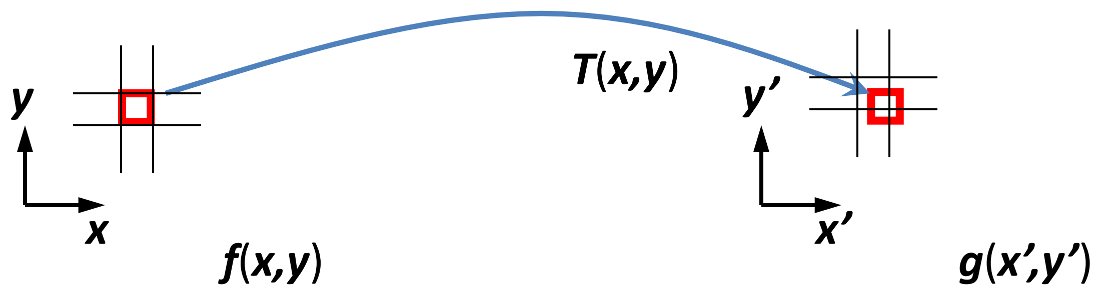
- **Send each pixel *f*(*x*) to its corresponding location (*x'*,*y'*) = *T*(*x*,*y*) in *g*(*x'*,*y'*)**
  - What if pixel lands "between" two pixels?



f(*x*,*y*)   *g*(*x'*,*y'*)

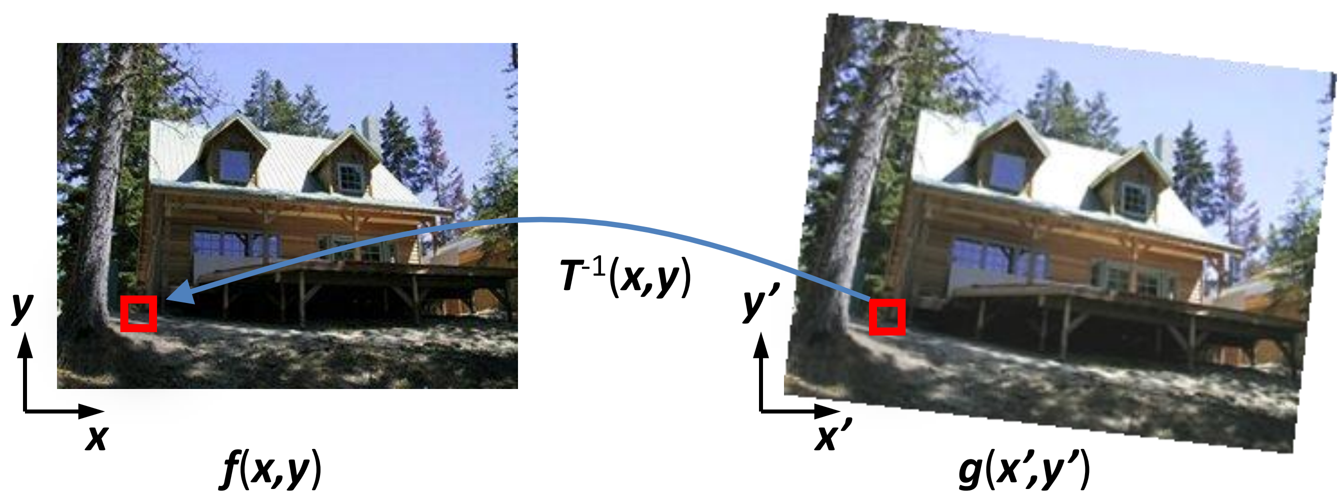
Forward Warping

- **Send each pixel *f*(*x*,*y*) to its corresponding location *x'* = *h*(*x*,*y*) in *g*(*x'*,*y'*)**
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)
  - Can still result in holes



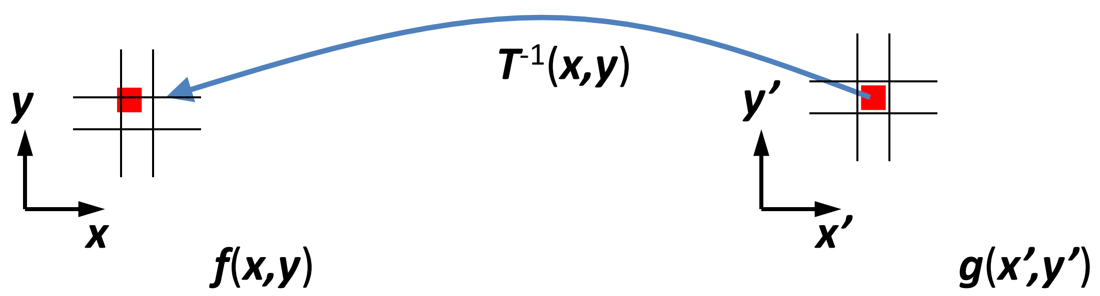
Inverse Warping

- **Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x,y)$ in $f(x,y)$**
  - Requires taking the inverse of the transform
  - What if pixel comes from “between” two pixels?



Inverse Warping

- **Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$**
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image



Interpolation

- **Possible interpolation filters:**
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)
  - sinc

- **Needed to prevent "jaggies" and "texture crawl"**

  (with prefiltering)

Blending

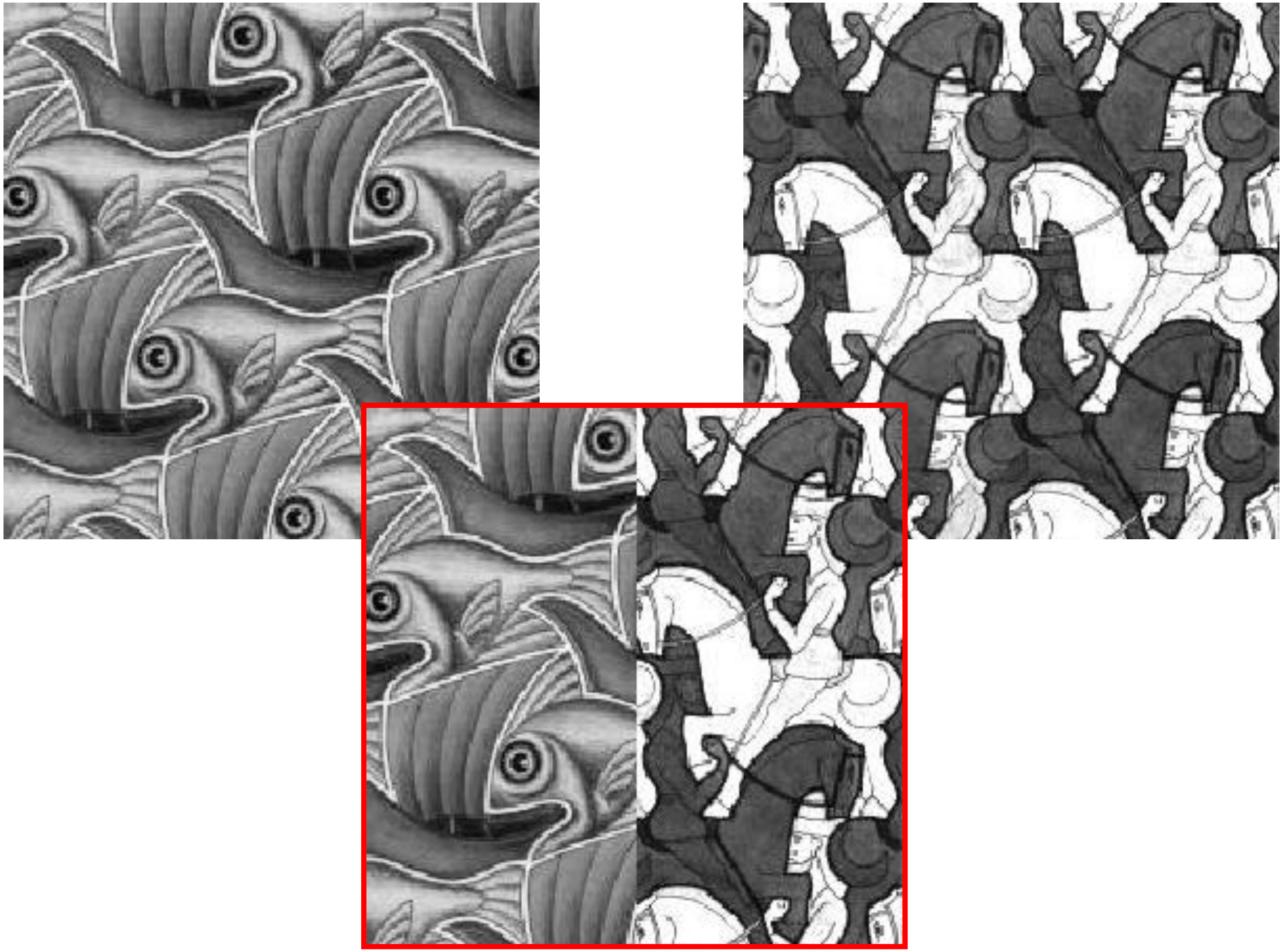
Blending

- **We've aligned the images – now what?**

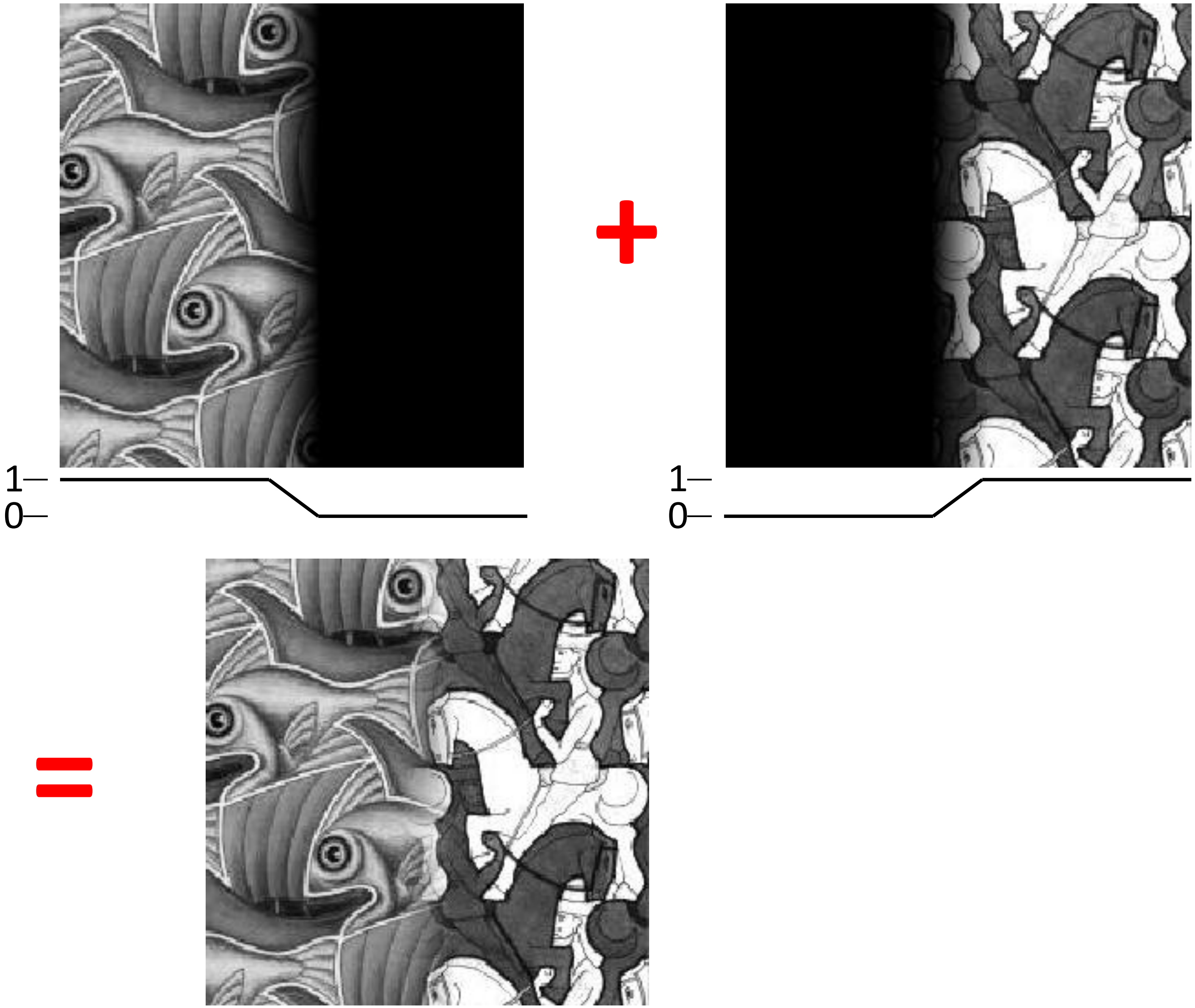
Blending

- **Want to seamlessly blend them together**

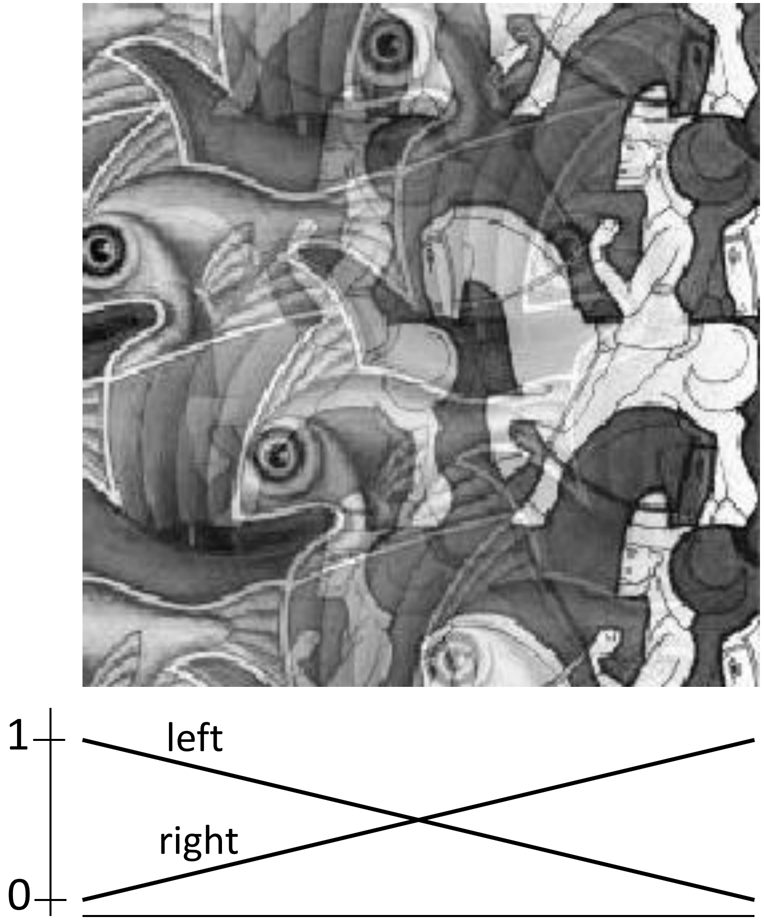
Image Blending

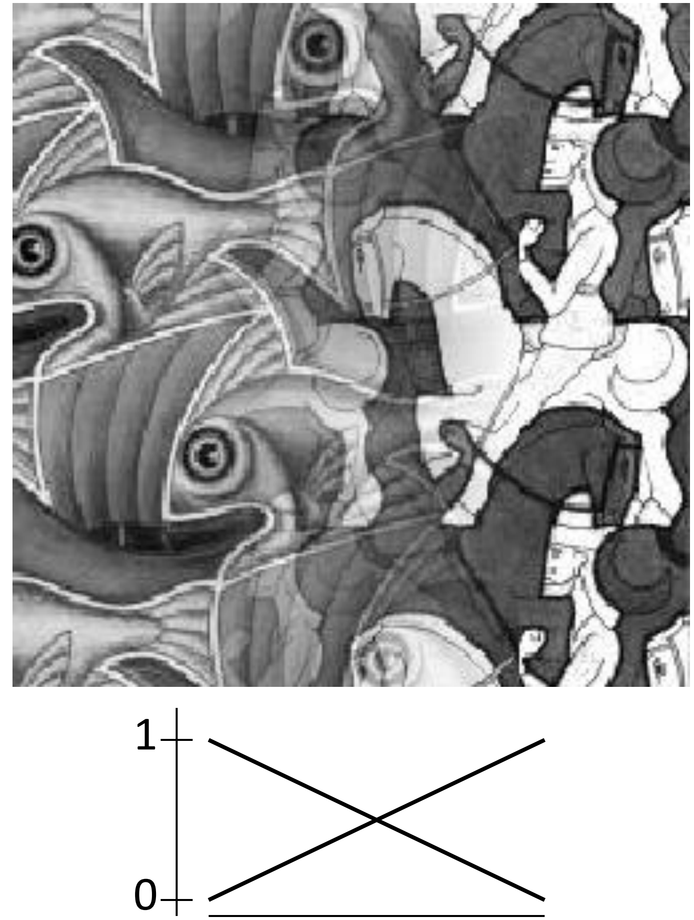


Feathering

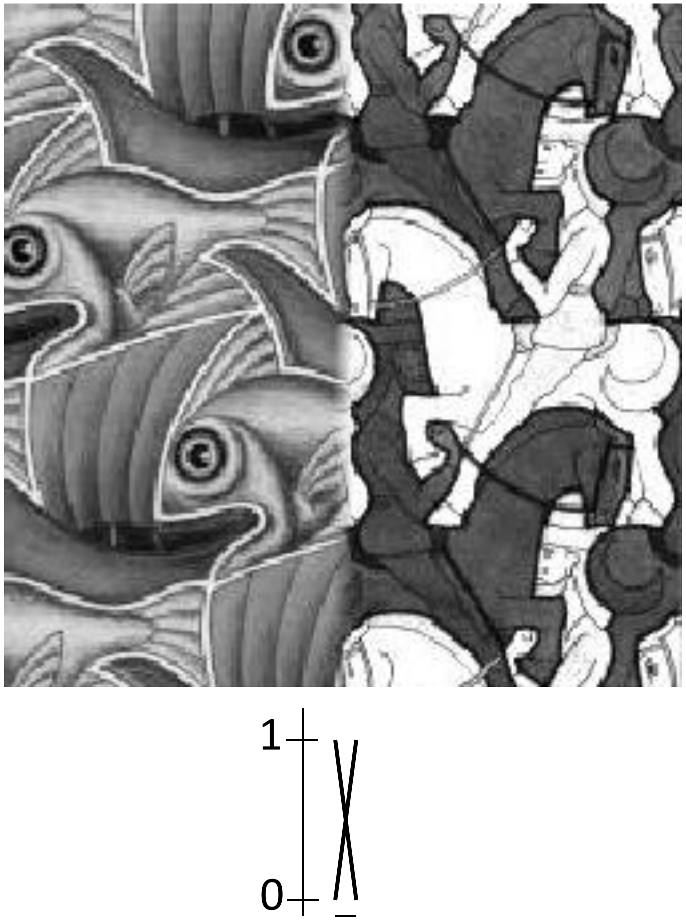


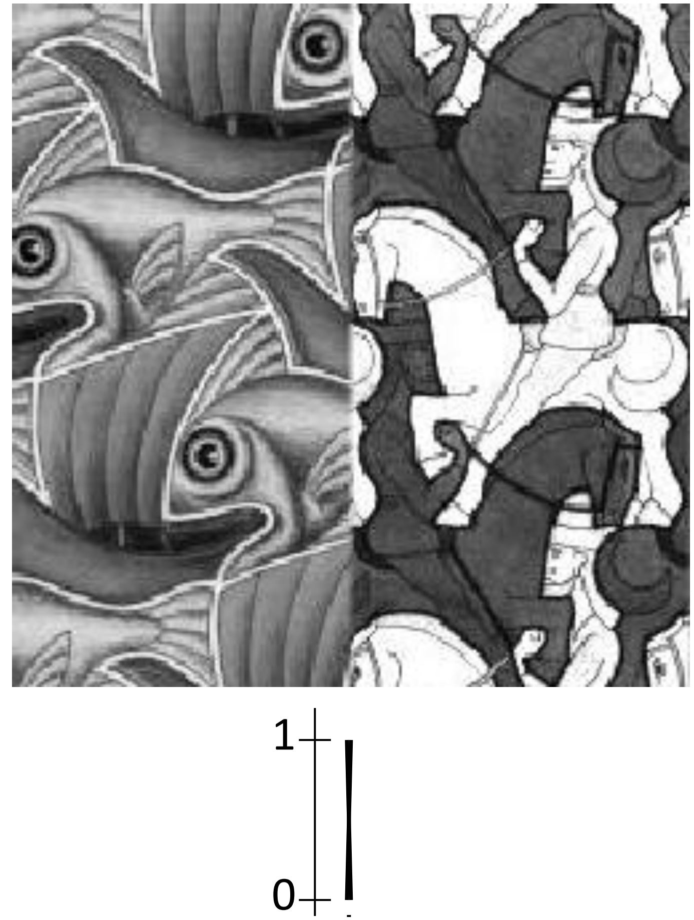
Effect of window size



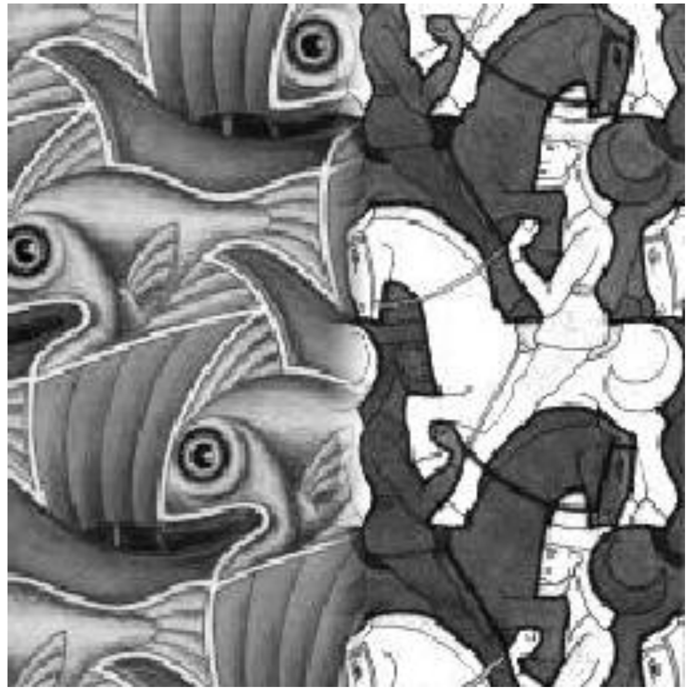


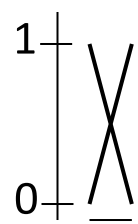
Effect of window size





Good window size

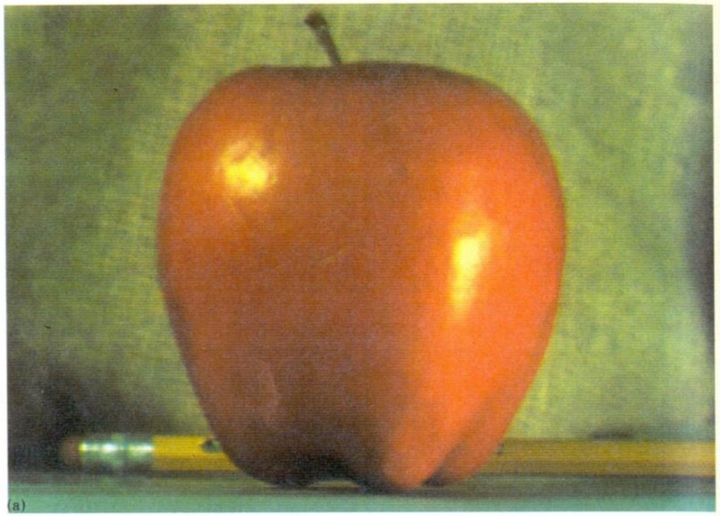


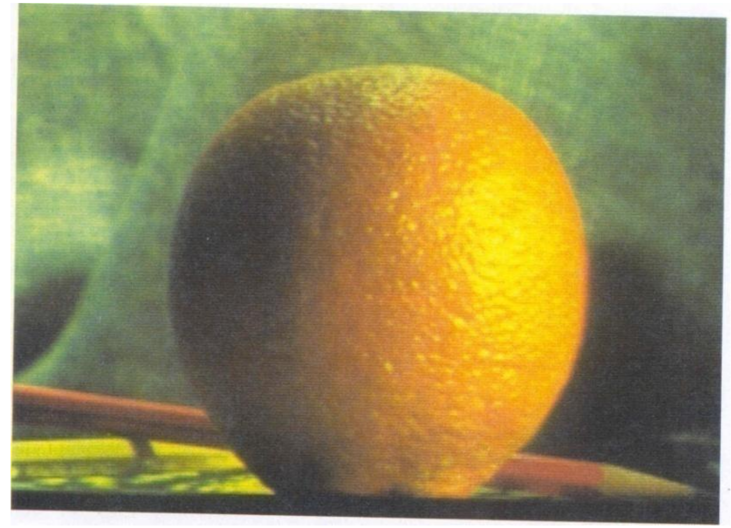


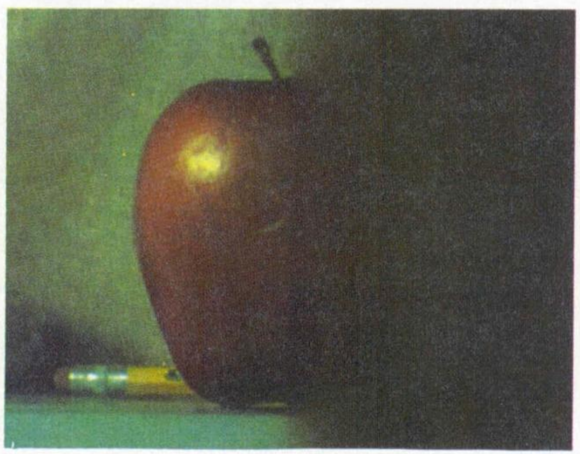
"Optimal" window: smooth but not ghosted

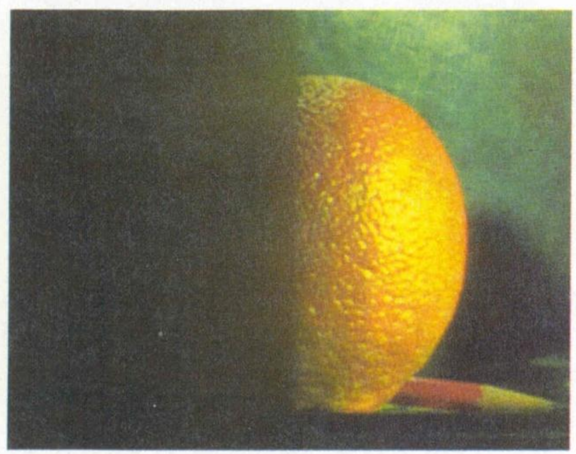
- Doesn't always work...

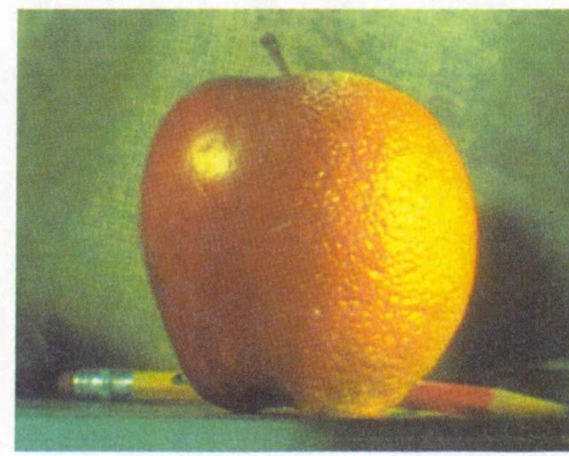
Pyramid blending






(d)

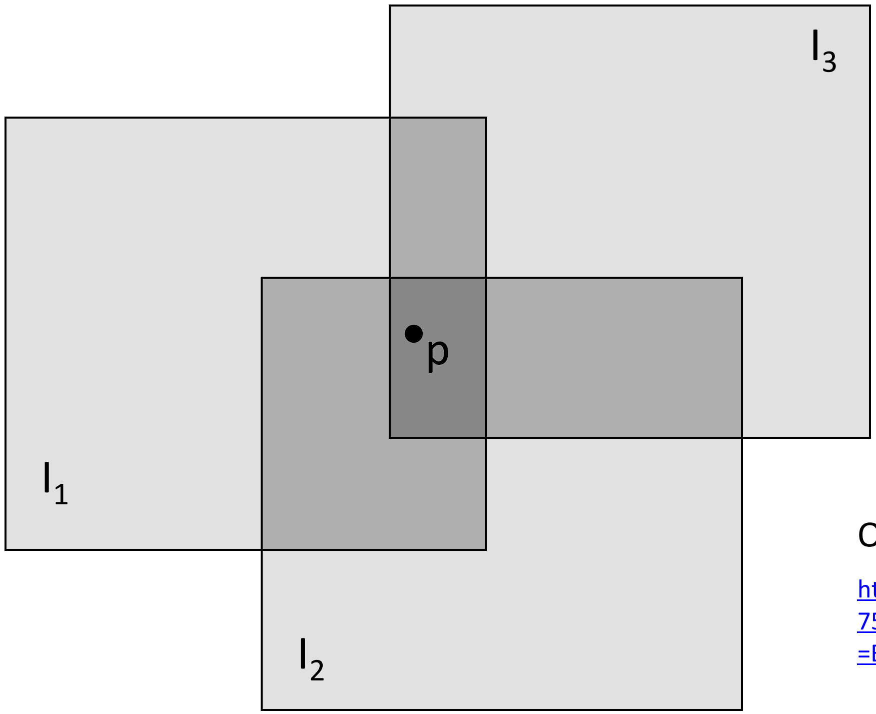

(h)


(l)

Create a Laplacian pyramid, blend each level

- Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

Alpha Blending



Optional: see Blinn (CGA, 1994) for details:

http://ieeexplore.ieee.org/iel1/38/7531/00310740.pdf?isNumber=7531&prod=JNL&arnumber=310740&arSt=83&ared=87&arAuthor=Blinn%2C+J.F.

Encoding blend weights: $I(x,y) = (\alpha R, \alpha G, \alpha B, \alpha)$

color at p = $$\frac{(\alpha_1 R_1,\ \alpha_1 G_1,\ \alpha_1 B_1) + (\alpha_2 R_2,\ \alpha_2 G_2,\ \alpha_2 B_2) + (\alpha_3 R_3,\ \alpha_3 G_3,\ \alpha_3 B_3)}{\alpha_1 + \alpha_2 + \alpha_3}$$

Implement this in two steps:

1. accumulate: add up the (α premultiplied) RGBα values at each pixel
2. normalize: divide each pixel's accumulated RGB by its α value

   Q: what if $\alpha = 0$?

Poisson Image Editing

- **For more info: Perez et al, SIGGRAPH 2003**
  - http://research.microsoft.com/vision/cambridge/papers/perez_siggraph03.pdf

Readings

- F & P 12.1.2 12.1.3
- Szeliski, *CVAA*:
  - Chapter 3.5: Image warping
  - Chapter 6.1: 2D and 3D Feature-based alignment
  - Chapter 9.1: Motion models
  - Chapter 9.2: Global alignment
  - Chapter 9.3: Compositing

- Recognizing Panoramas, Brown & Lowe, ICCV’2003
- Szeliski & Shum, SIGGRAPH'97