Image alignment

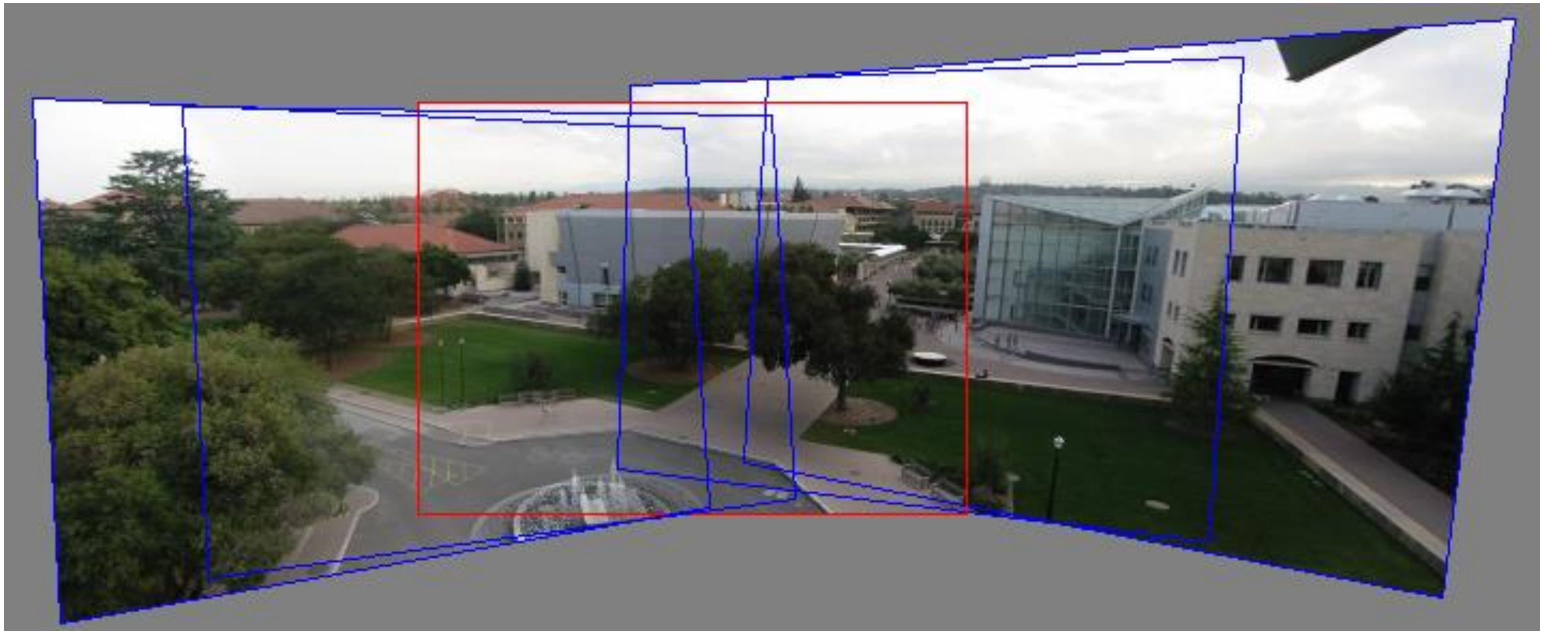
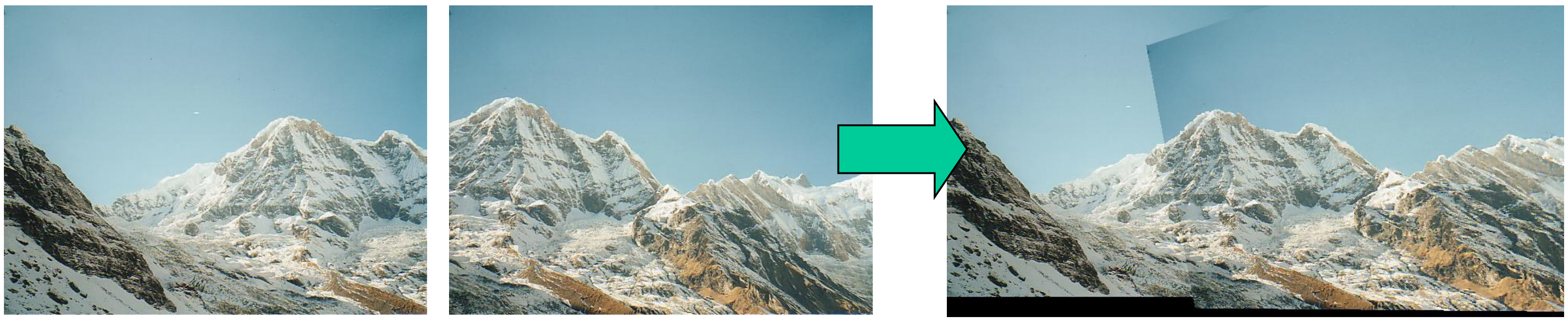
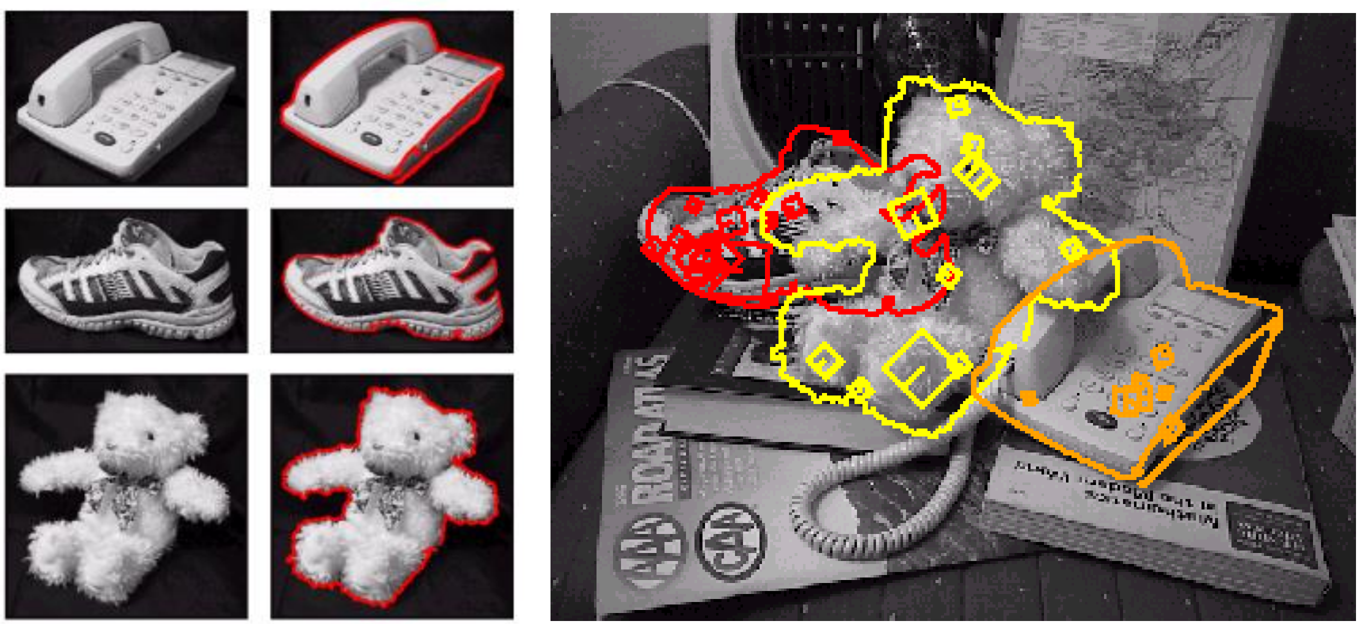


Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fall/

Image alignment: Applications

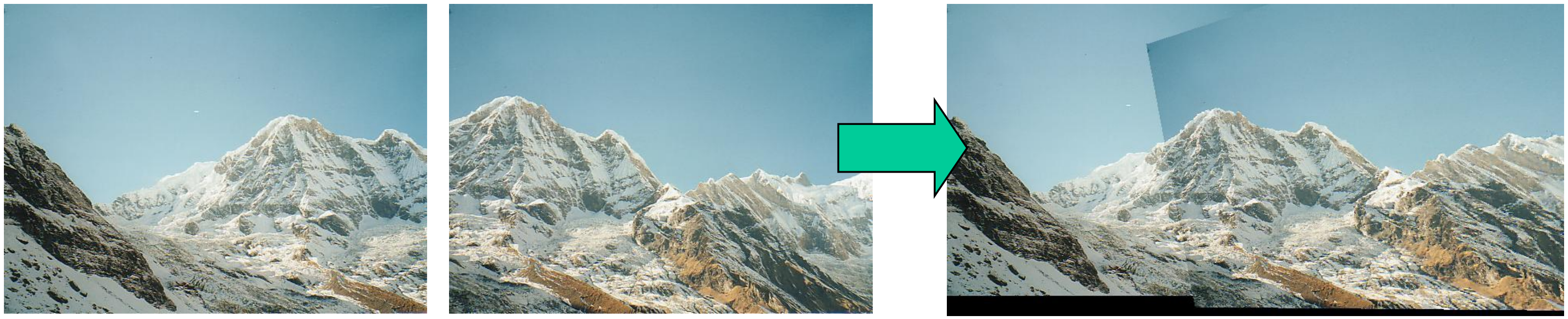


Panorama stitching



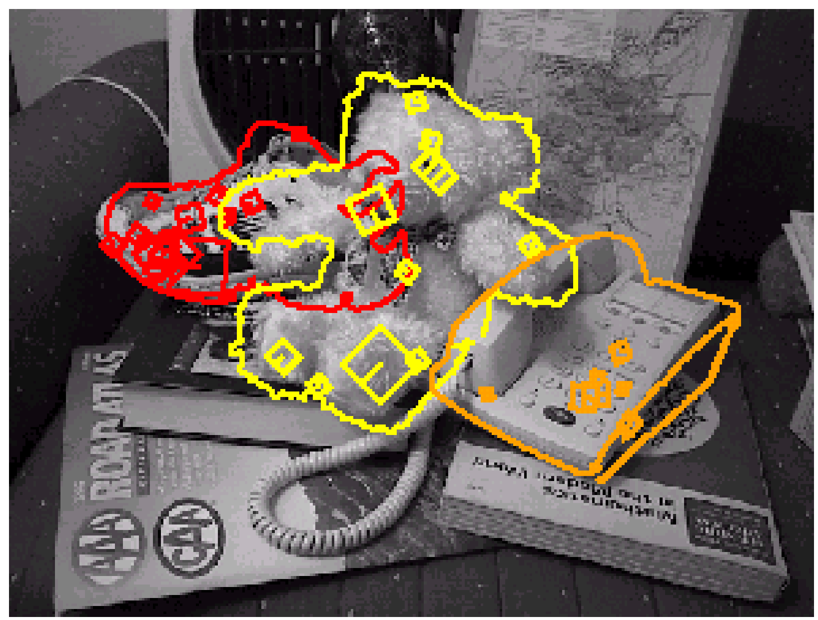
Recognition of object instances

Image alignment: Challenges



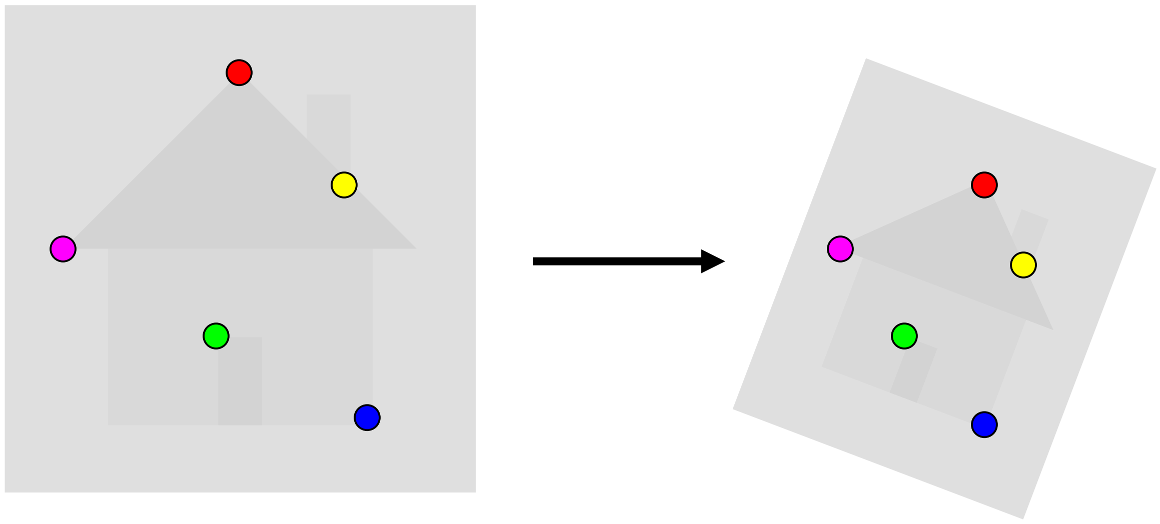
Small degree of overlap

Intensity changes



Occlusion, clutter

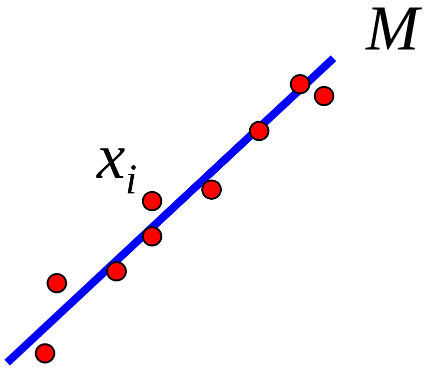
Image alignment



- Two families of approaches:
  - **Direct (pixel-based) alignment**
    - Search for alignment where most pixels agree
  - **Feature-based alignment**
    - Search for alignment where *extracted features* agree
    - Can be verified using pixel-based alignment

Alignment as fitting

- Previous lectures: fitting a model to features in one image

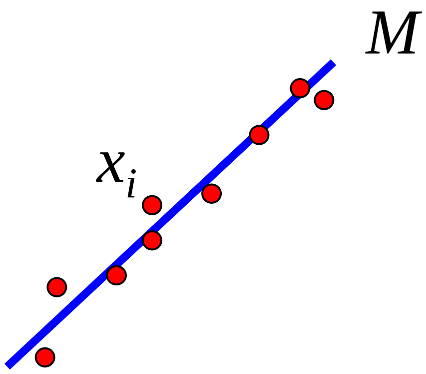


Find model *M* that minimizes

$$\sum_i \mathrm{residual}(x_i, M)$$

Alignment as fitting

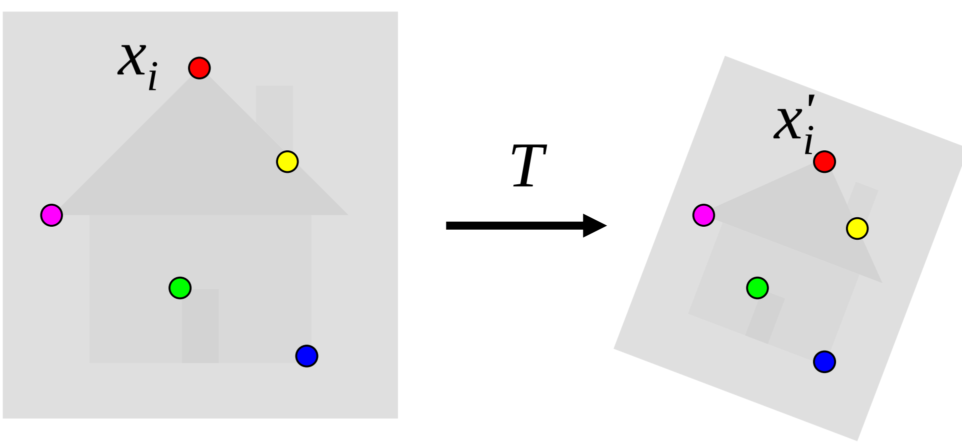
- Previous lectures: fitting a model to features in one image



Find model *M* that minimizes

$$\sum_i \text{residual}(x_i, M)$$

- Alignment: fitting a model to a transformation between pairs of features (*matches*) in two images

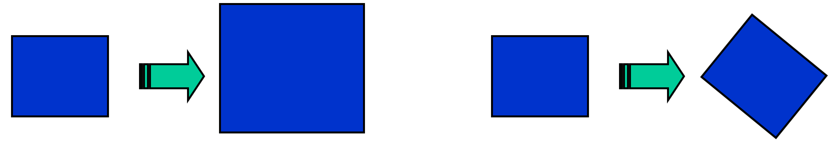


Find transformation *T* that minimizes

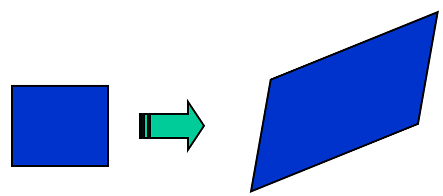
$$\sum_i \text{residual}(T(x_i), x'_i)$$

2D transformation models

- Similarity (translation, scale, rotation)



- Affine



- Projective (homography)

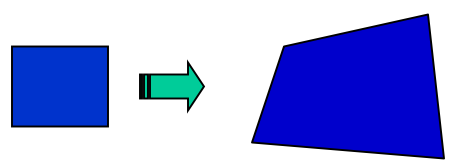
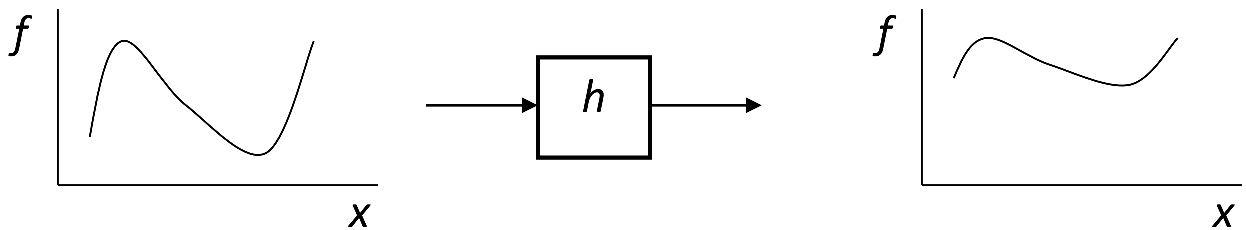


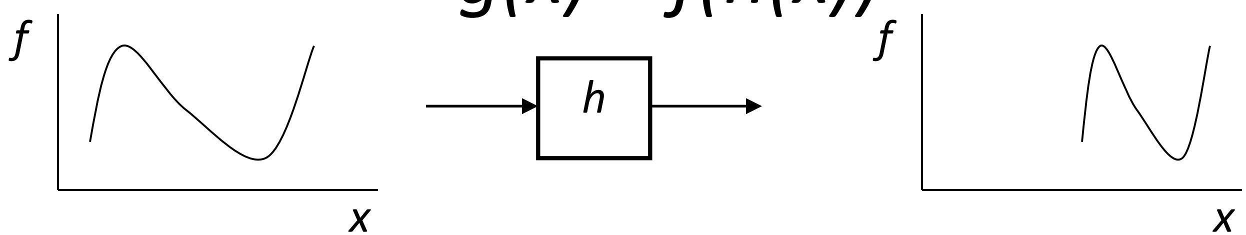
Image Warping

Image Warping

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



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

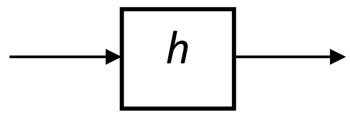


Szeliski

Image Warping

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

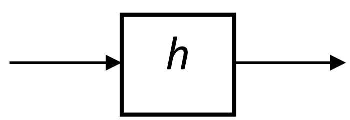
f



g

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

f



g

Parametric (global) warping

- Examples of parametric warps:

translation

rotation

aspect

affine

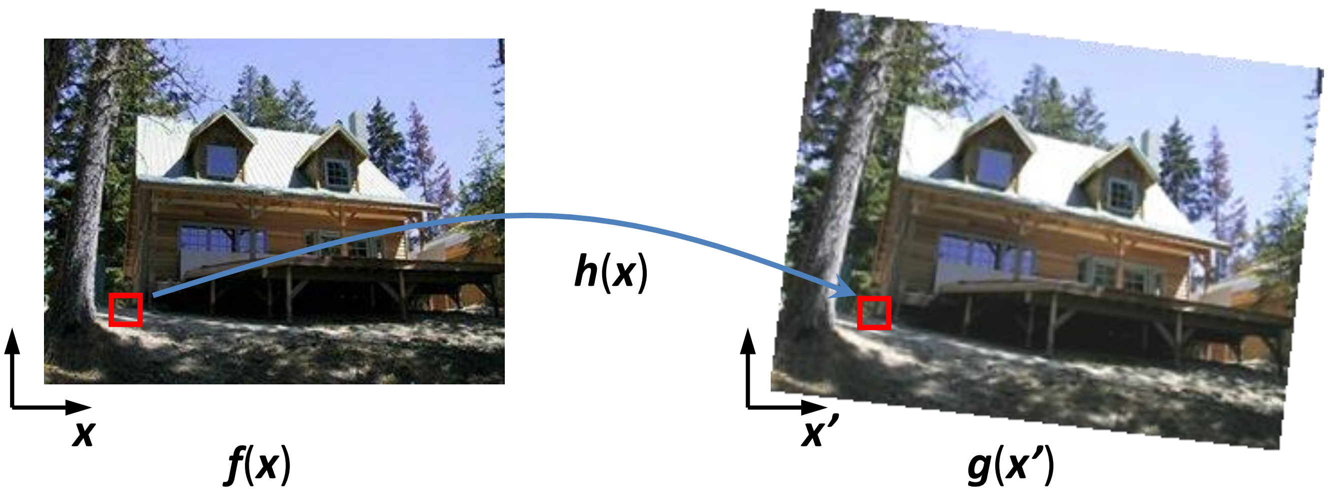
perspective

cylindrical

Szeliski

Image Warping

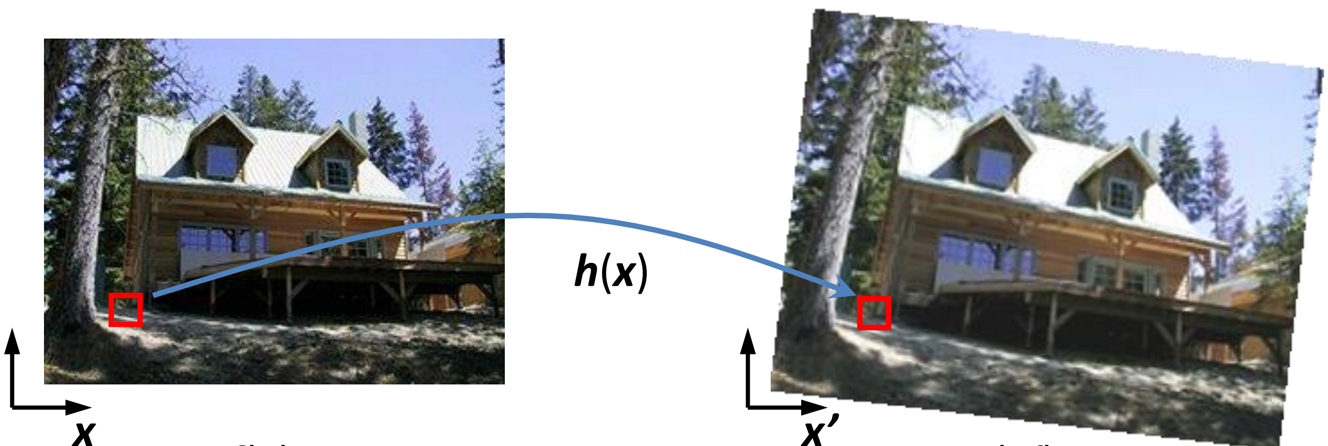
- Given a coordinate transform $\mathbf{x}' = \mathbf{h}(\mathbf{x})$ and a source image $\mathbf{f}(\mathbf{x})$, how do we compute a transformed image $\mathbf{g}(\mathbf{x}') = \mathbf{f}(\mathbf{h}(\mathbf{x}))$?



Szeliski

Forward Warping

- Send each pixel ***f***(***x***) to its corresponding location ***x'*** = ***h***(***x***) in ***g***(***x'***)
  - What if pixel lands "between" two pixels?

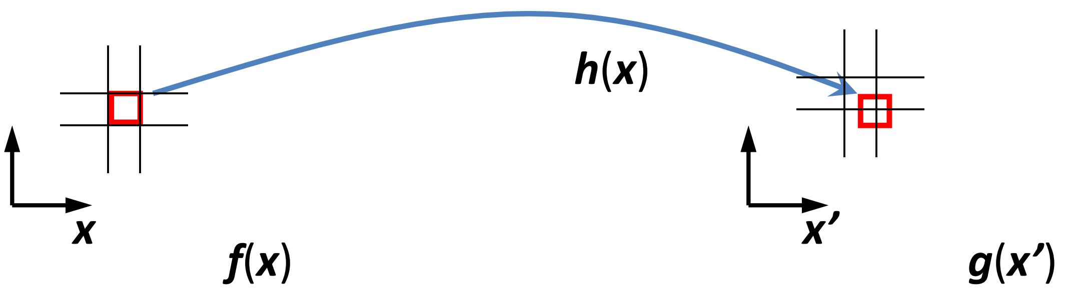


f(***x***) ***g***(***x'***)

Szeliski

Forward Warping

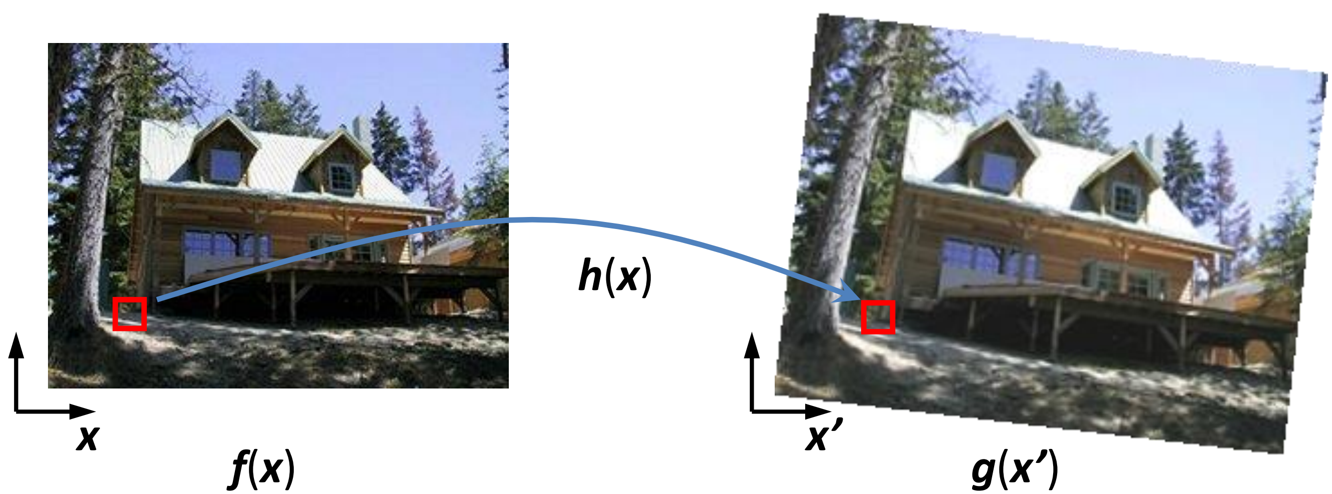
- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x'})$
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)



Szeliski

Inverse Warping

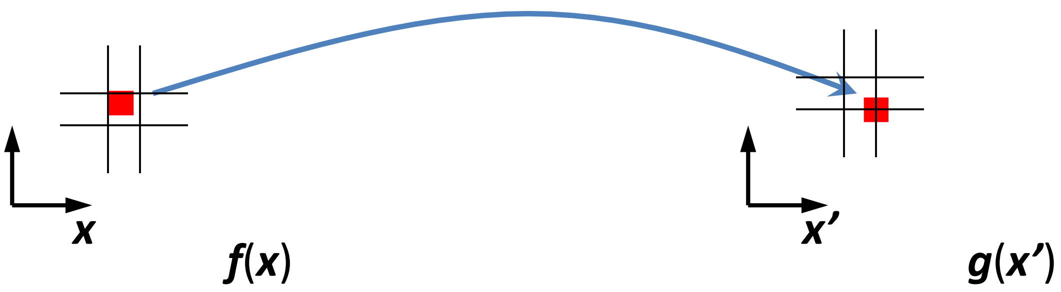
- Get each pixel $\boldsymbol{g}(\boldsymbol{x'})$ from its corresponding location $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{f}(\boldsymbol{x})$
  - What if pixel comes from "between" two pixels?



Szeliski

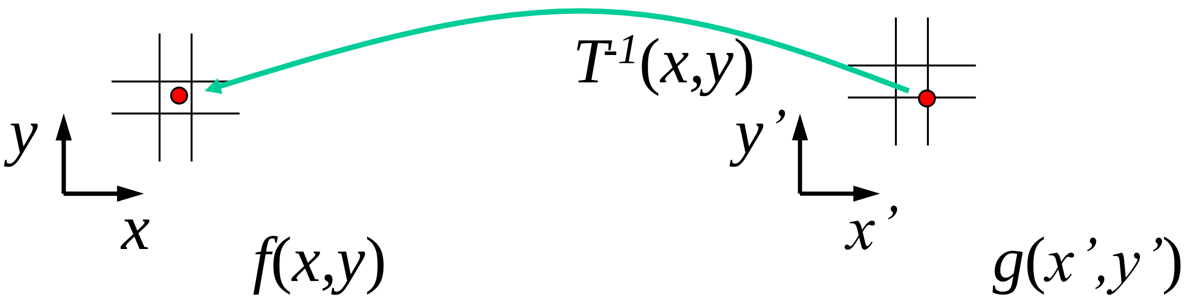
Inverse Warping

- 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



Szeliski

Inverse warping



Get each pixel $g(x',y')$ from its corresponding location

$(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

A: *Interpolate* color value from neighbors

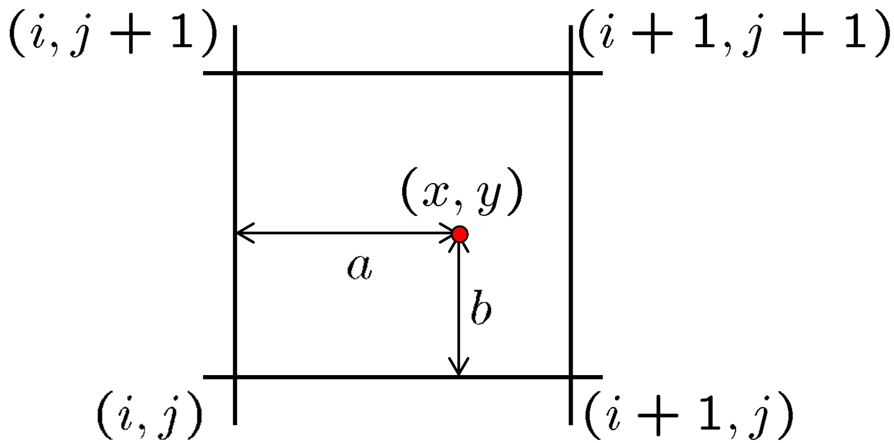
– nearest neighbor, bilinear…

```
>> help interp2
```

Slide from Alyosha Efros, CMU

Bilinear interpolation

Sampling at *f(x,y):*



$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) \quad && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

Interpolation

- Possible interpolation filters
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)
  - sinc / FIR
- Needed to prevent "jaggies" and "texture crawl"

Szeliski

2D coordinate transformations

- translation: $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$ $\boldsymbol{x} = (x,y)$
- rotation: $\boldsymbol{x}' = \boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- similarity: $\boldsymbol{x}' = s\,\boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- affine: $\boldsymbol{x}' = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{t}$
- perspective: $\underline{\boldsymbol{x}}' \cong \boldsymbol{H}\,\underline{\boldsymbol{x}}$ $\underline{\boldsymbol{x}} = (x,y,1)$
  ($\underline{\boldsymbol{x}}$ is a *homogeneous* coordinate)
- These all form a nested *group* (closed w/ inv.)

Szeliski

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

Source: Alyosha Efros

2D Affine Transformations

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

Affine transformations are combinations of …

- Linear transformations, and
- Translations

Parallel lines remain parallel

Grauman

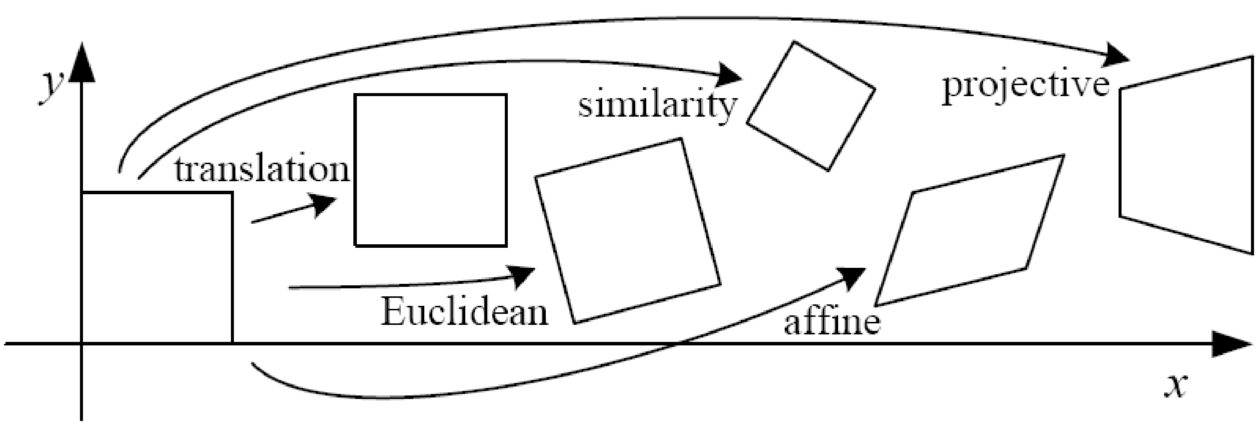
Projective Transformations

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

Projective transformations:

- Affine transformations, and
- Projective warps

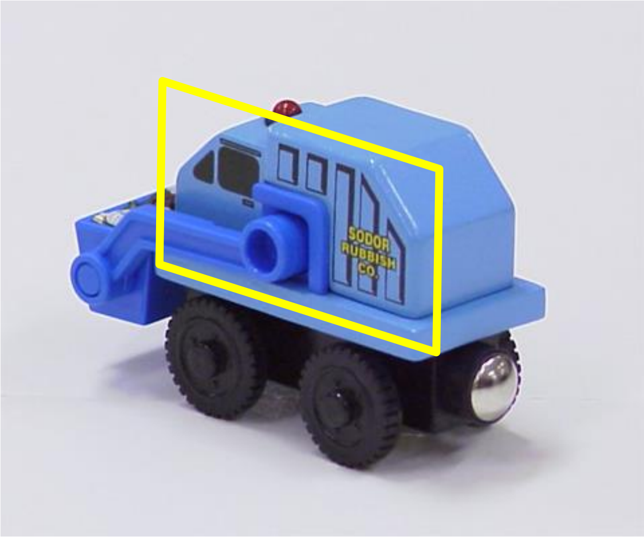
Parallel lines do not necessarily remain parallel



Grauman

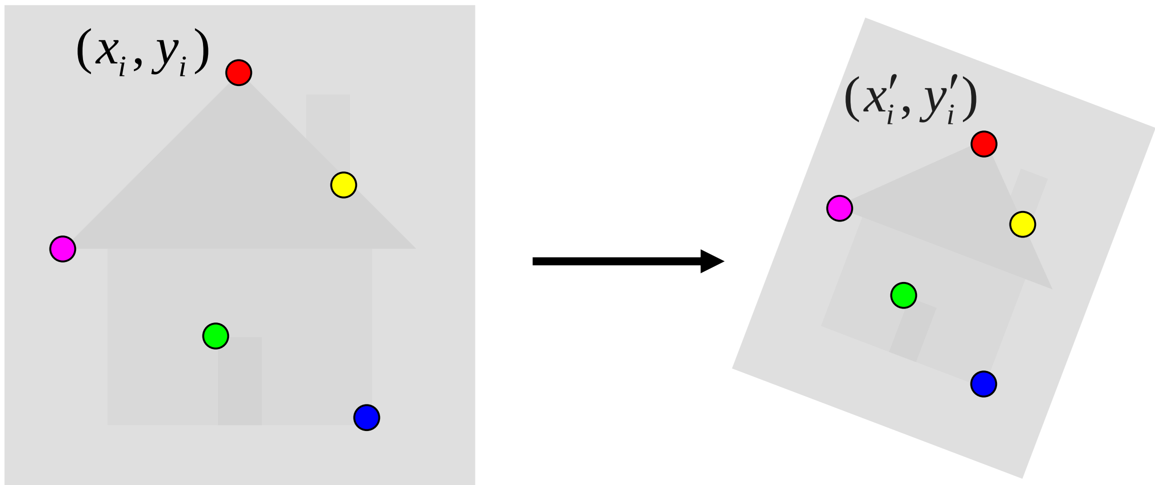
Let's start with affine transformations

- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used to initialize fitting for more complex models



Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

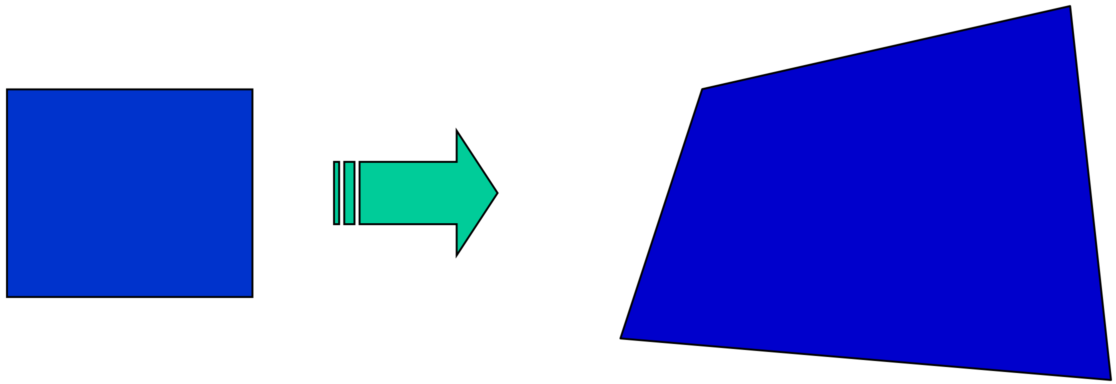
Fitting an affine transformation

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters

Fitting a plane projective transformation

- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)

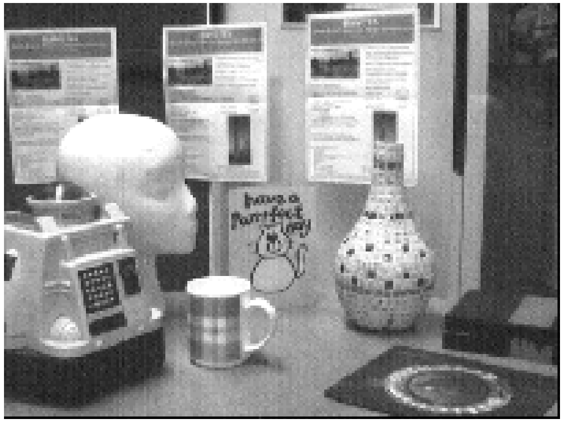
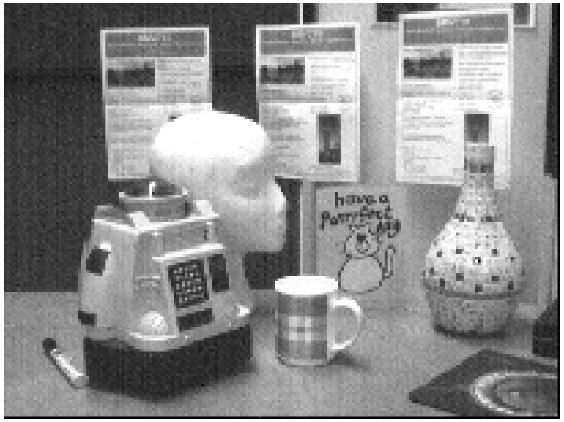


Homography

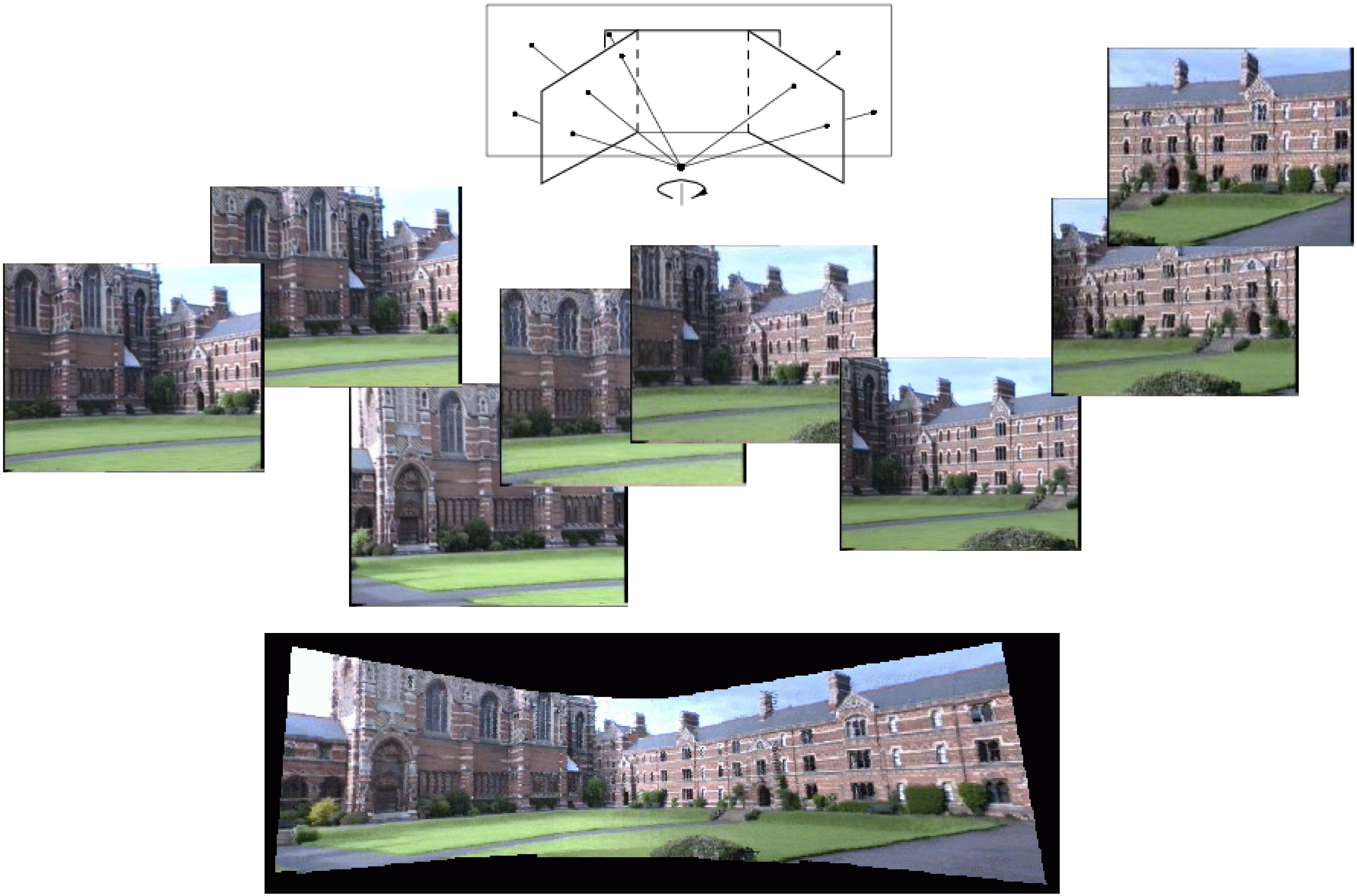
- The transformation between two views of a planar surface

- The transformation between images from two cameras that share the same center

Application: Panorama stitching



Source: Hartley & Zisserman

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \left[\begin{array}{c} x \\ y \\ 1 \end{array}\right]$$

Converting *to* homogeneous image coordinates

$$\left[\begin{array}{c} x \\ y \\ w \end{array}\right] \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

- Equation for homography:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Solving for homographies

p' = Hp

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Can set scale factor $h_{33}=1$. So, there are 8 unknowns.

Set up a system of linear equations:

Ah = b

where vector of unknowns $h = [h_{11}, h_{12}, \ldots, h_{32}]^T$

Need at least 8 eqs, but the more the better…

Solve for h. If overconstrained, solve using least-squares:

$$\min \|Ah - b\|^2$$

Graum

Fitting a homography

- Equation for homography:

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\lambda \mathbf{x}_i' = \mathbf{H}\,\mathbf{x}_i$$

$$\mathbf{x}_i' \times \mathbf{H}\,\mathbf{x}_i = 0$$

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h}_{\mathbf{1}}^T \mathbf{x}_i \\ \mathbf{h}_{\mathbf{2}}^T \mathbf{x}_i \\ \mathbf{h}_{\mathbf{3}}^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y_i' \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x_i' \mathbf{h}_3^T \mathbf{x}_i \\ x_i' \mathbf{h}_2^T \mathbf{x}_i - y_i' \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

$$\begin{bmatrix} 0^T & -\mathbf{x}_i^T & y_i' \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x_i' \mathbf{x}_i^T \\ -y_i' \mathbf{x}_i^T & x_i' \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

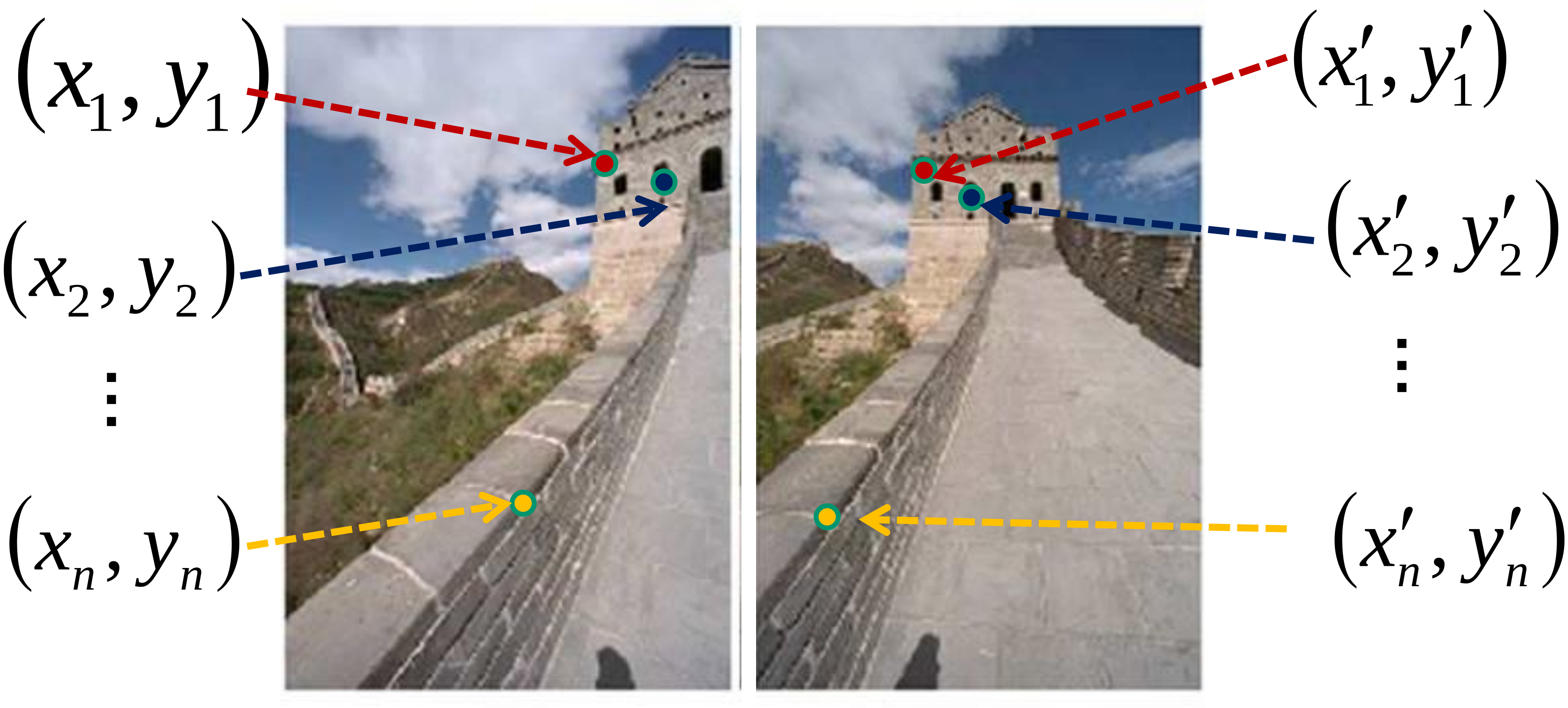
3 equations, only 2 linearly independent

Direct linear transform

$$\begin{bmatrix} 0^T & \mathbf{x}_1^T & -y_1'\,\mathbf{x}_1^T \\ \mathbf{x}_1^T & 0^T & -x_1'\,\mathbf{x}_1^T \\ \cdots & \cdots & \cdots \\ 0^T & \mathbf{x}_n^T & -y_n'\,\mathbf{x}_n^T \\ \mathbf{x}_n^T & 0^T & -x_n'\,\mathbf{x}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \qquad \mathbf{A}\,\mathbf{h} = 0$$

- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Four matches needed for a minimal solution
- More than four: homogeneous least squares

Homography



To compute the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of H are the unknowns…

Grauman