CS 1674: Intro to Computer Vision

Geometric Transformations and Multiple Views

Prof. Adriana Kovashka
University of Pittsburgh
October 9, 2018

Why multiple views?

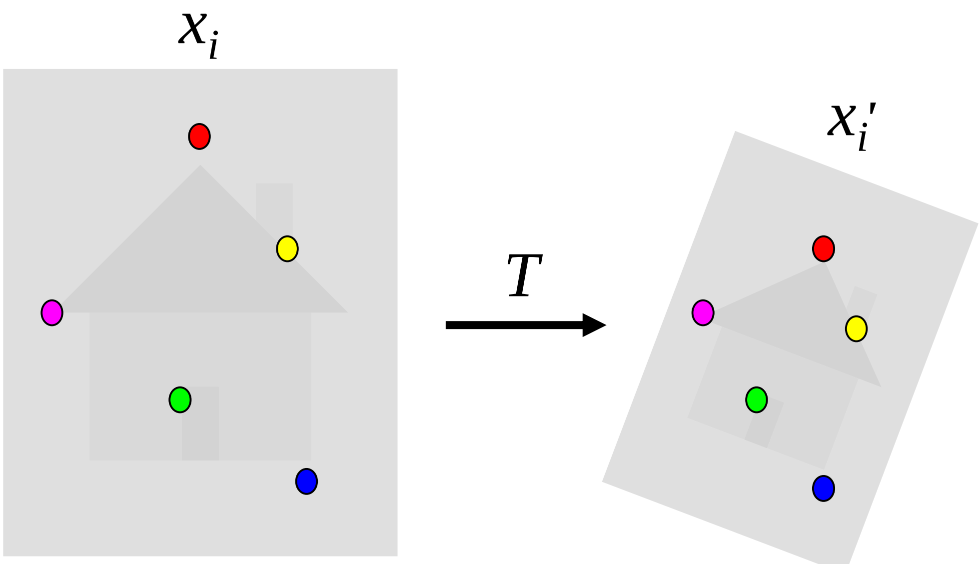
- Structure and depth are inherently ambiguous from single views.

- Multiple views help us to perceive 3d shape and depth.

Kristen Grauman, images from Svetlana Lazebnik

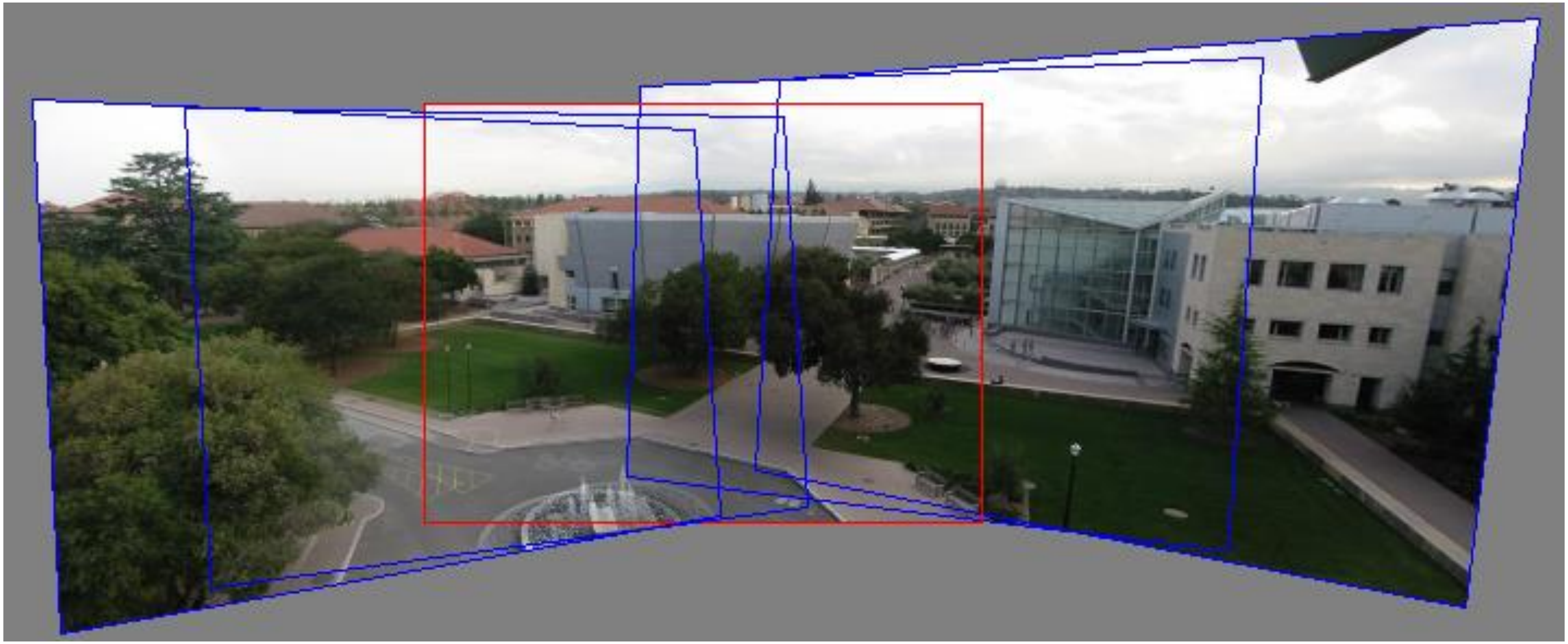
Alignment problem

- We previously discussed how to match features across images, of the same or different objects
- Now let's focus on the case of "two images of the same object"(e.g. x_i and x_i')
- What transformation relates x_i and x_i'?
- In *alignment*, we will **fit the parameters of some transformation** according to a set of matching feature pairs ("correspondences").



Adapted from Kristen Grauman and Derek Hoiem

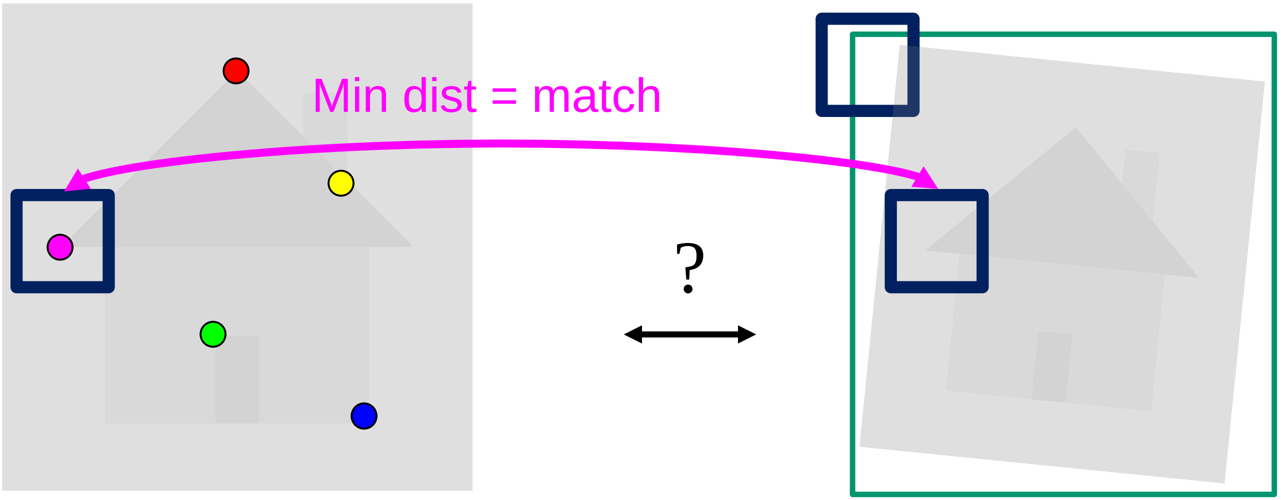
Motivation: Image mosaics



Kristen Grauman

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

First, what are the correspondences?



- Compare content in **local** patches, find best matches.
  - Scan x_i' with template formed from a point in x_i, and compute e.g. Euclidean distance between pixel intensities in the patch
  - Or compare SIFT features

Adapted from Kristen Grauman

Second, what are the transformations?

Examples of transformations:


translate


rotate

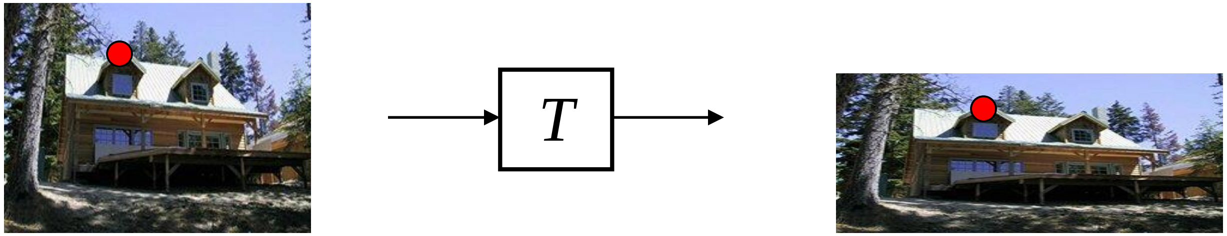

change aspect ratio


squish


change perspective

Adapted from Alyosha Efros

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**?

- It is the same for any point p
- It can be described by just a few numbers (parameters)

Let's represent T as a matrix:

$$p' = \mathbf{M}p$$

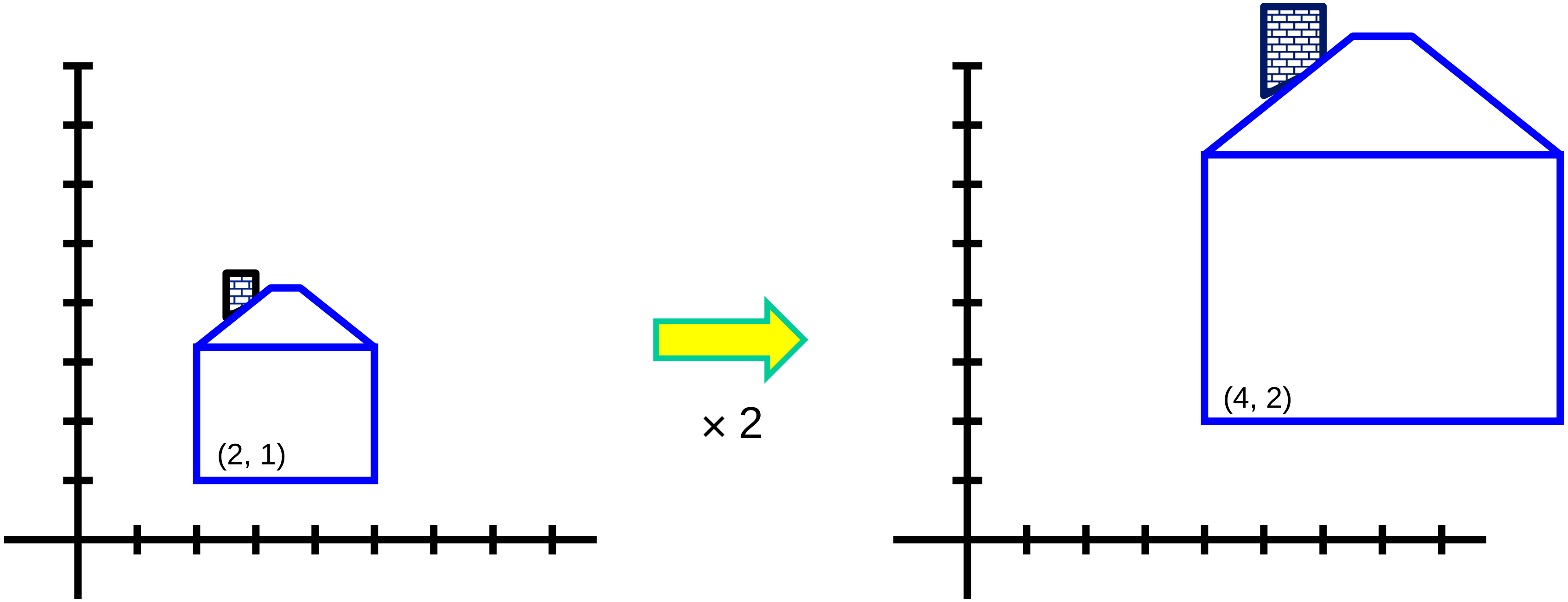
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Alyosha Efros

Scaling

Scaling a coordinate means multiplying each of its components by a scalar

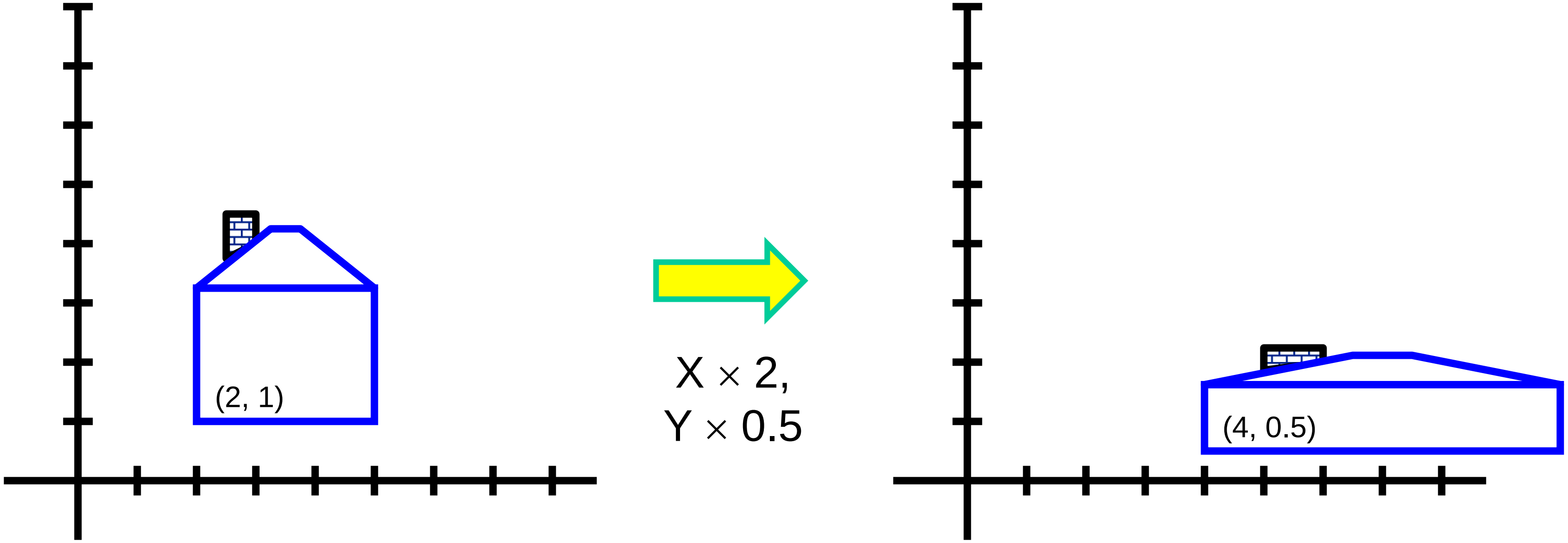
Uniform scaling means this scalar is the same for all components:



Adapted from Alyosha Efros

Scaling

Non-uniform scaling: different scalars per component



Adapted from Alyosha Efros

Scaling

Scaling operation:

$$x' = ax$$

$$y' = by$$

Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

x’ = mx + ny
y’ = px + qy

Adapted from Alyosha Efros

2D Linear transformations

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

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

Linear transformations are combinations of …

- Scale,
- Rotation,
- Shear, and
- Mirror

Alyosha Efros

What transforms can we write w/ 2x2 matrix?

2D Scaling?

$$\boldsymbol{x' = s_x * x}$$

$$\boldsymbol{y' = s_y * y}$$

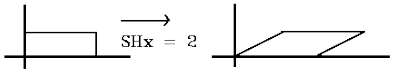
$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_x} & 0 \\ 0 & \boldsymbol{s_y} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

2D Rotate around (0,0)? (see hidden slide)

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

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

2D Shear?



$$\boldsymbol{x' = x + sh_x * y}$$

$$\boldsymbol{y' = sh_y * x + y}$$

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

Modified from Alyosha Efros

Fig. from https://www.siggraph.org/education/materials/HyperGraph/modeling/mod_tran/2dshear.htm

What transforms can we write w/ 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)?

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

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

2D Translation?

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}_x$$
$$\boldsymbol{y}' = \boldsymbol{y} + \boldsymbol{t}_y$$

CAN’T DO!

Alyosha Efros

Homogeneous coordinates

To convert to homogeneous coordinates:

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

homogeneous image
coordinates

Converting *from* homogeneous coordinates

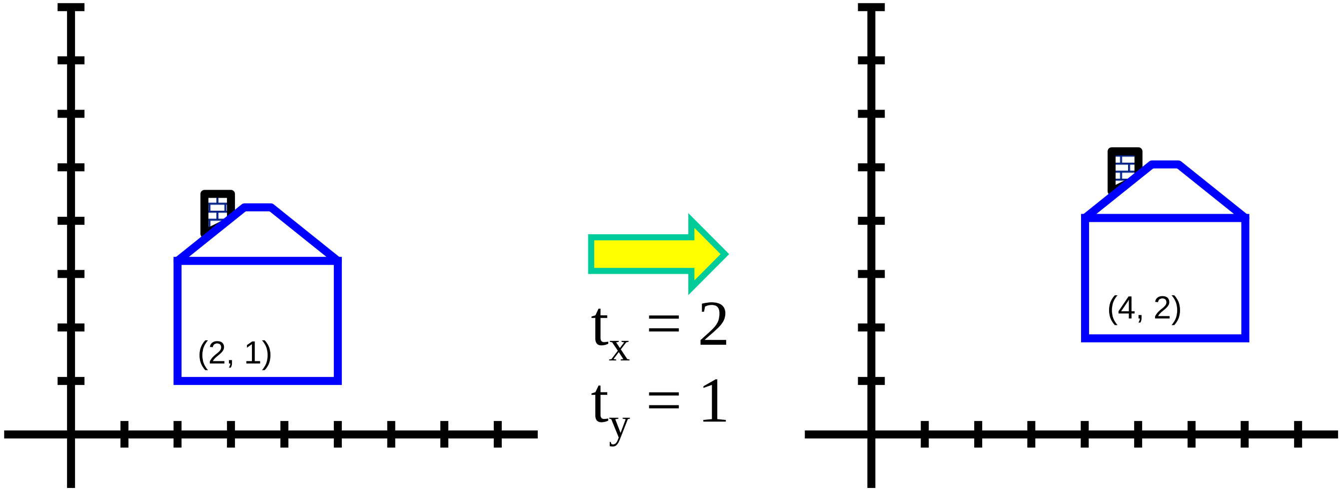
$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Kristen Grauman

Translation

Homogeneous Coordinates

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



$t_x = 2$

$t_y = 1$

Adapted from Alyosha Efros

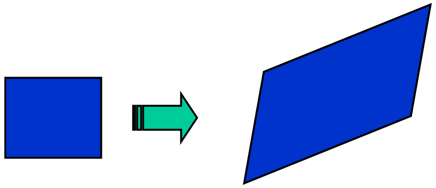
2D affine transformations

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

Affine transformations are combinations of …

- Linear transformations, and
- Translations

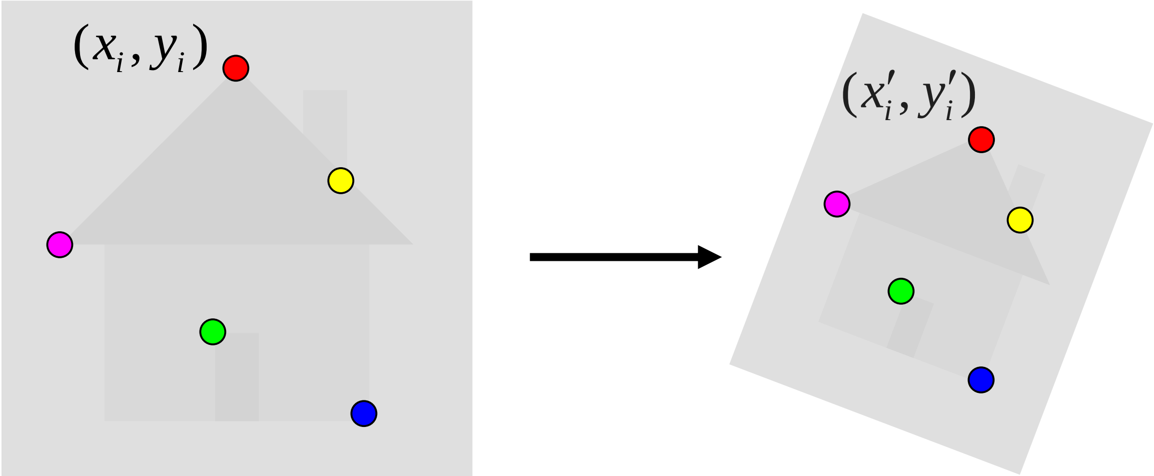
Maps lines to lines, parallel lines remain parallel



Adapted from Alyosha Efros

Fitting an affine transformation

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



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

$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

Alyosha Efros

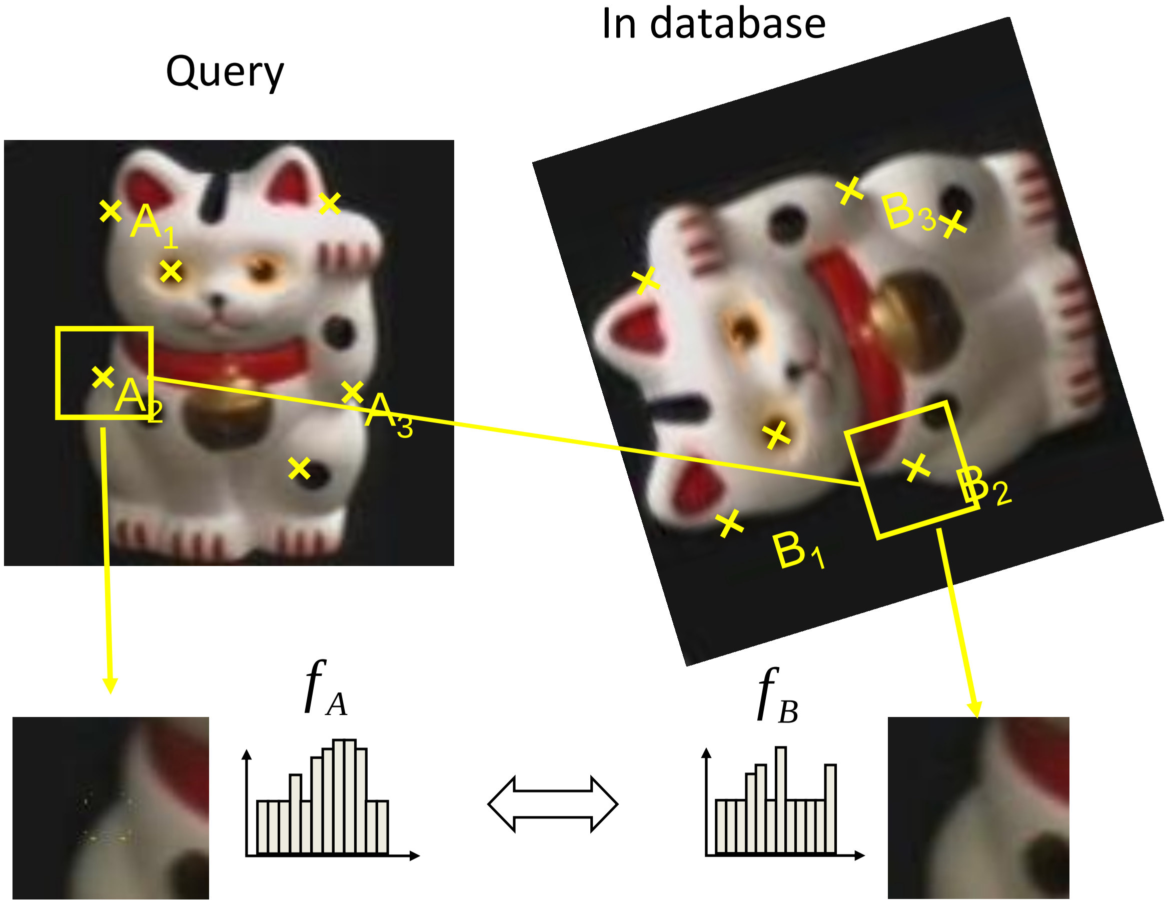
Fitting an affine transformation

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

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?
- Once we have solved for the parameters, how do we compute (x'_{new}, y'_{new}) given (x_{new}, y_{new}) ?

Adapted from Kristen Grauman

Detour: Keypoint matching for search

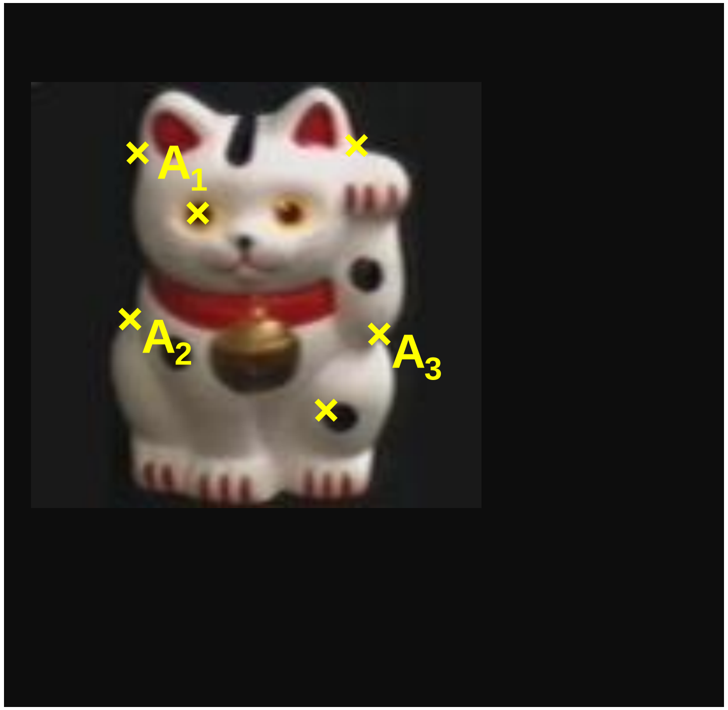


$$d(f_A, f_B) < T$$

1. Find a set of distinctive key-points
2. Define a region around each keypoint (window)
3. Compute a local descriptor from the region
4. Match descriptors

Adapted from K. Grauman, B. Leibe

Detour: solving for translation with outliers

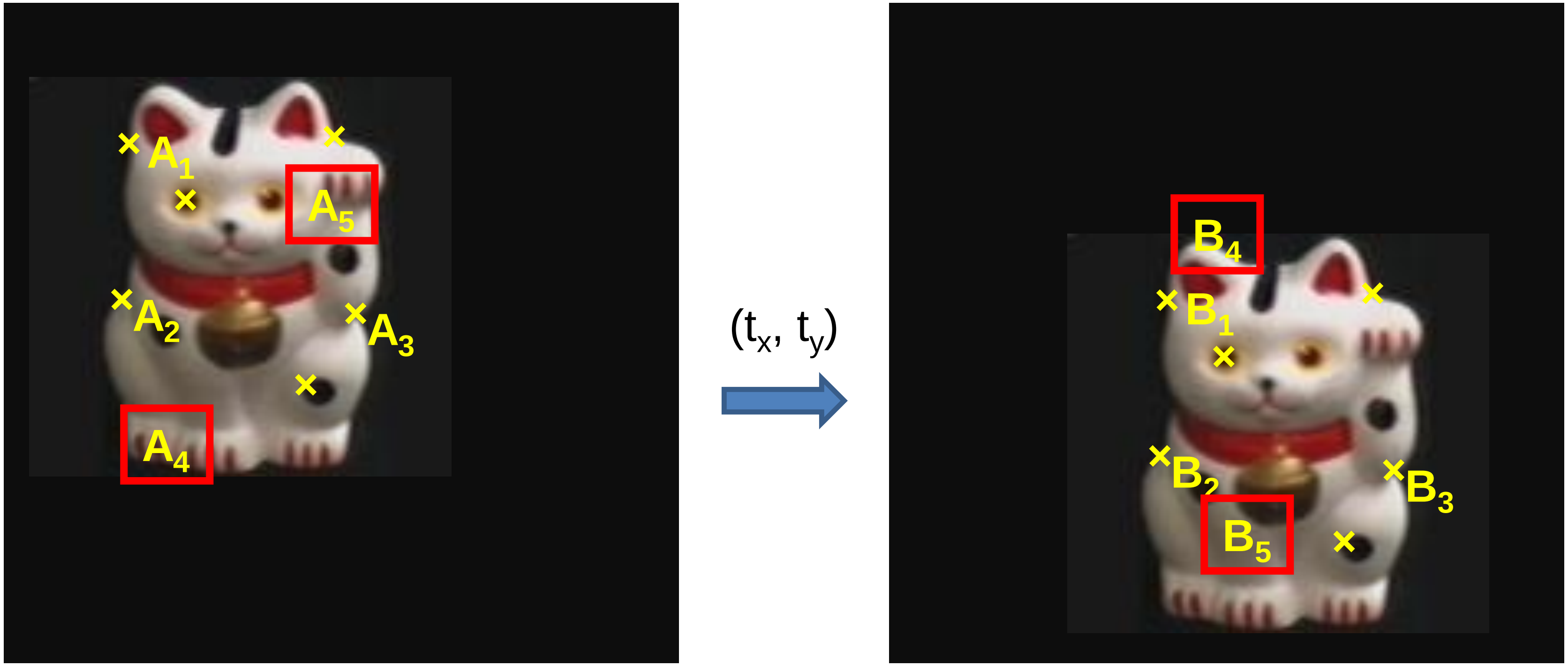


Given matched points in {A} and {B}, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Derek Hoiem

Detour: solving for translation with outliers



Problem: outliers

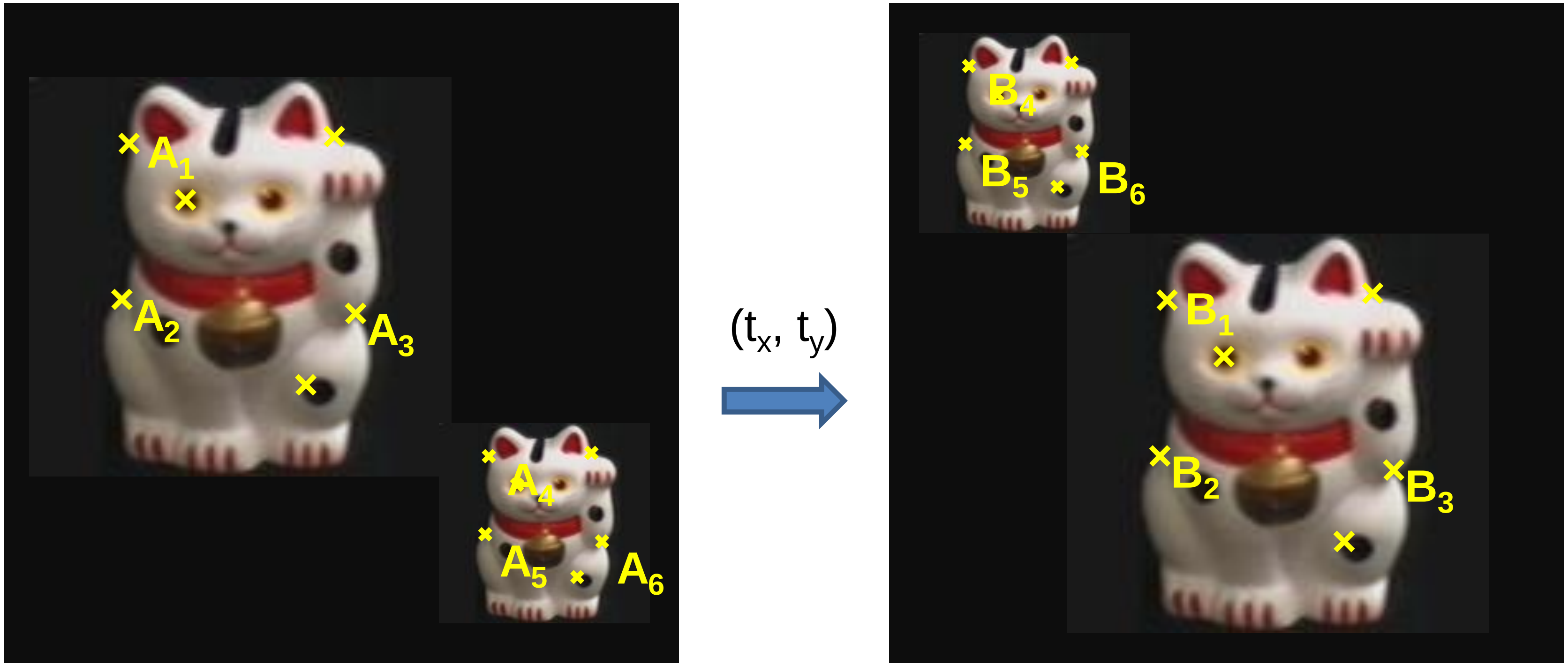
Hough transform solution

1. Initialize a grid of parameter values
2. Each matched pair casts a vote for consistent values
3. Find the parameters with the most votes

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Adapted from Derek Hoiem

Detour: solving for translation with outliers



Problem: multiple objects

Hough transform solution

1. Initialize a grid of parameter values
2. Each matched pair casts a vote for consistent values
3. Find the parameters with the most votes

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Adapted from Derek Hoiem

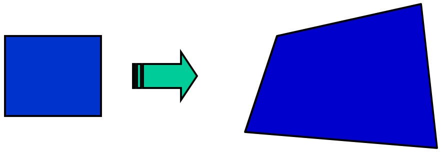
2D projective transformations

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

Projective transformations:

- Affine transformations, and
- Projective warps

Parallel lines do not necessarily remain parallel



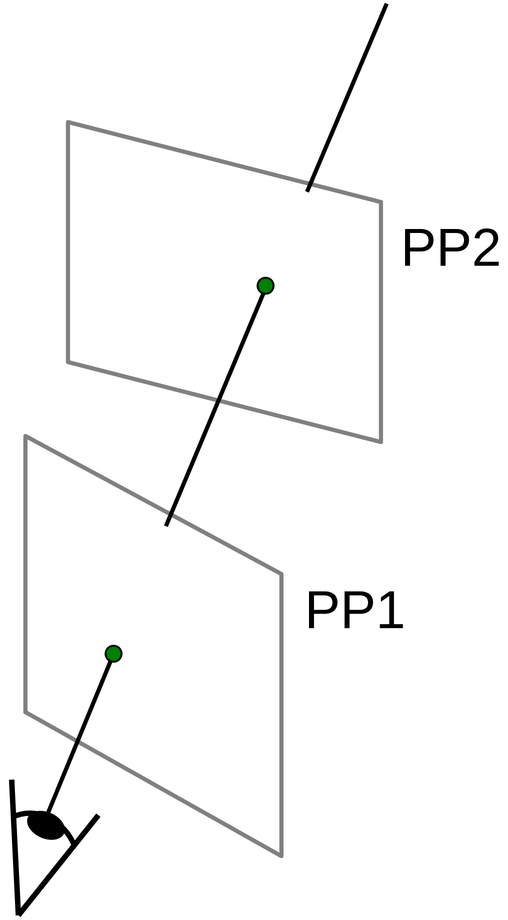
Kristen Grauman

Projective transformations

A projective transformation is a mapping between any two projective planes with *the same center of projection*

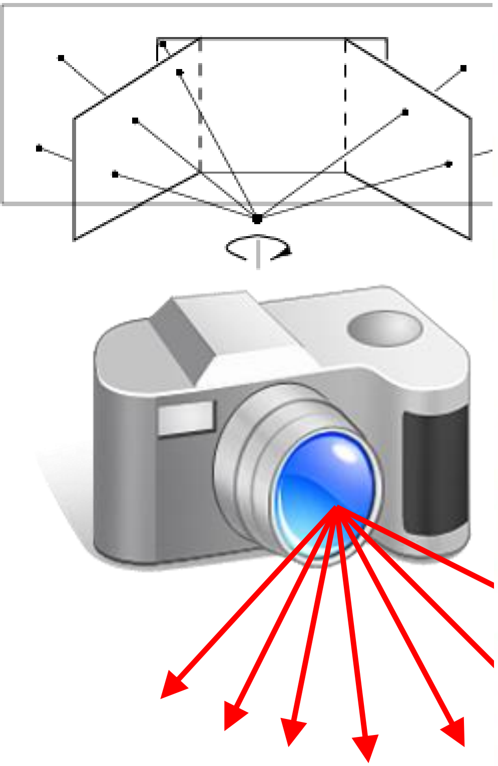
Also called **Homography**

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



Adapted from Alyosha Efros

Image mosaics: Goals



· · ·

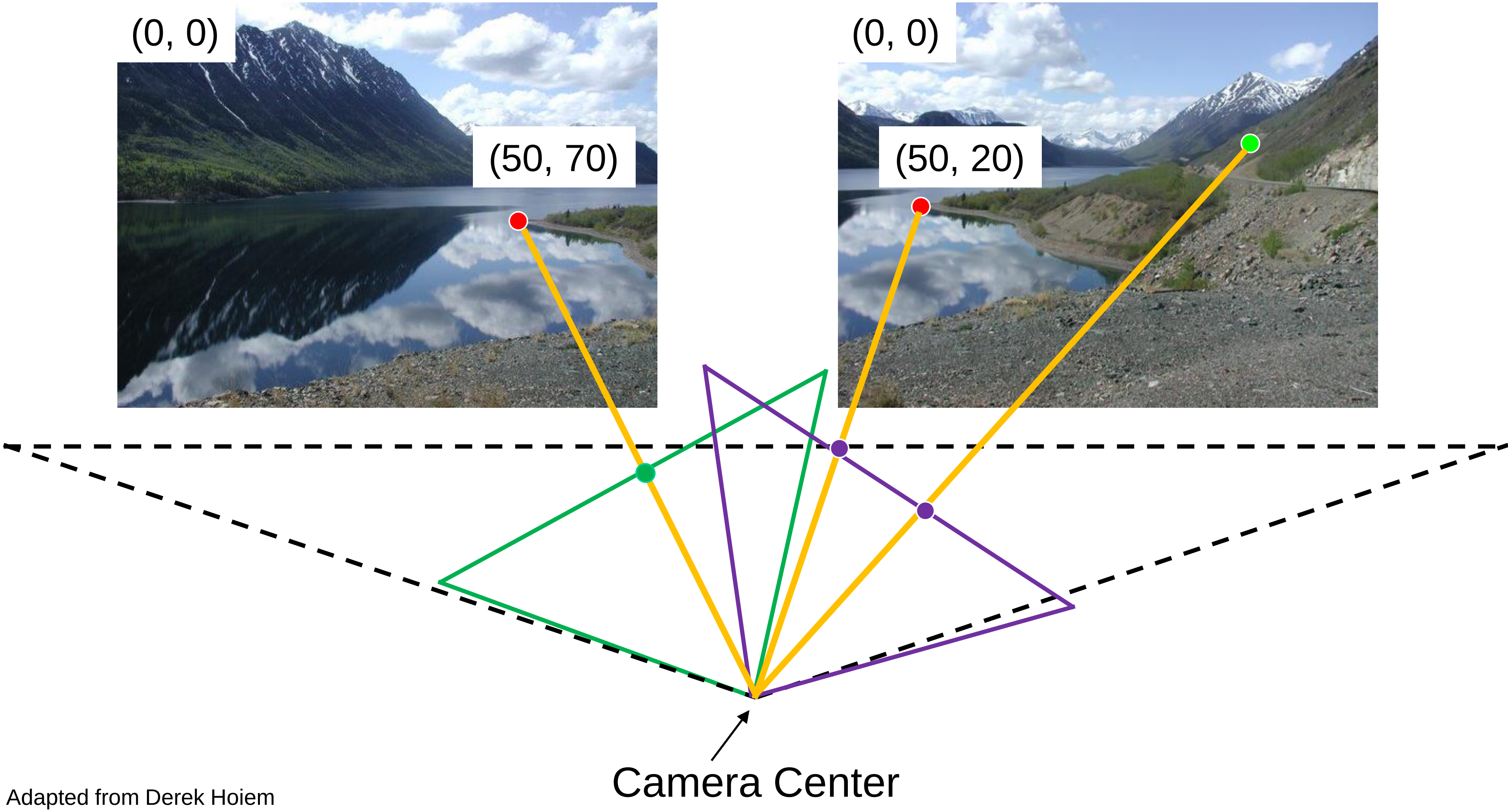
image from S. Seitz

Obtain a wider angle view by combining multiple images.

Kristen Grauman

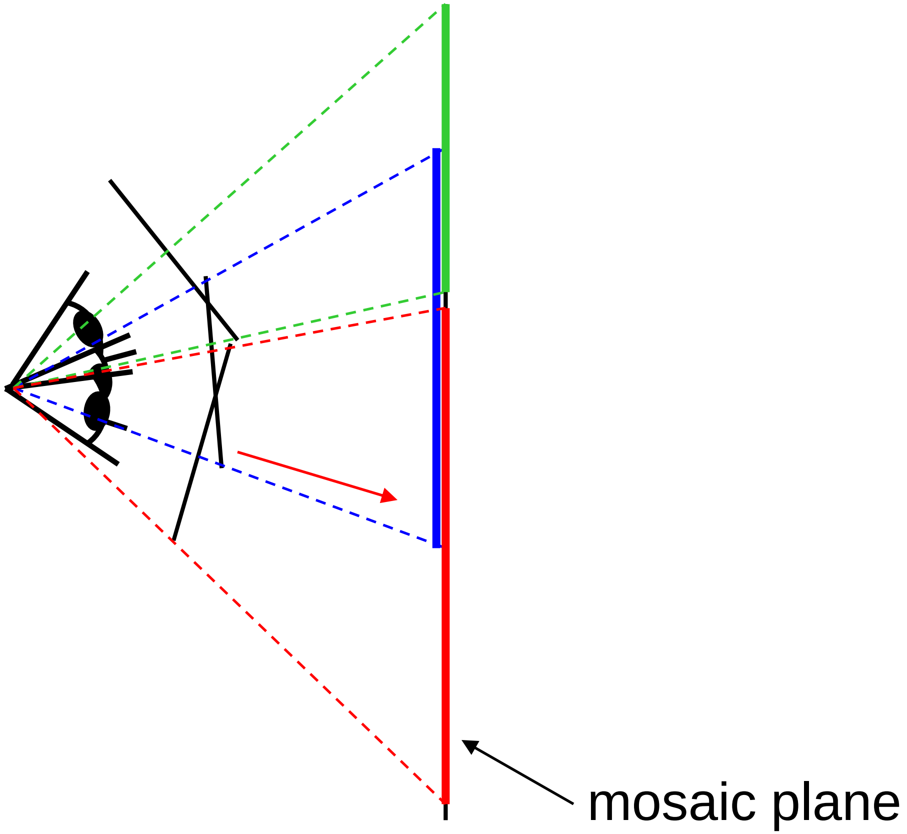
Image mosaics: Camera setup

Two images with camera rotation but no translation



Adapted from Derek Hoiem

Image mosaics: Many 2D views, one 3D object



The mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*

Steve Seitz

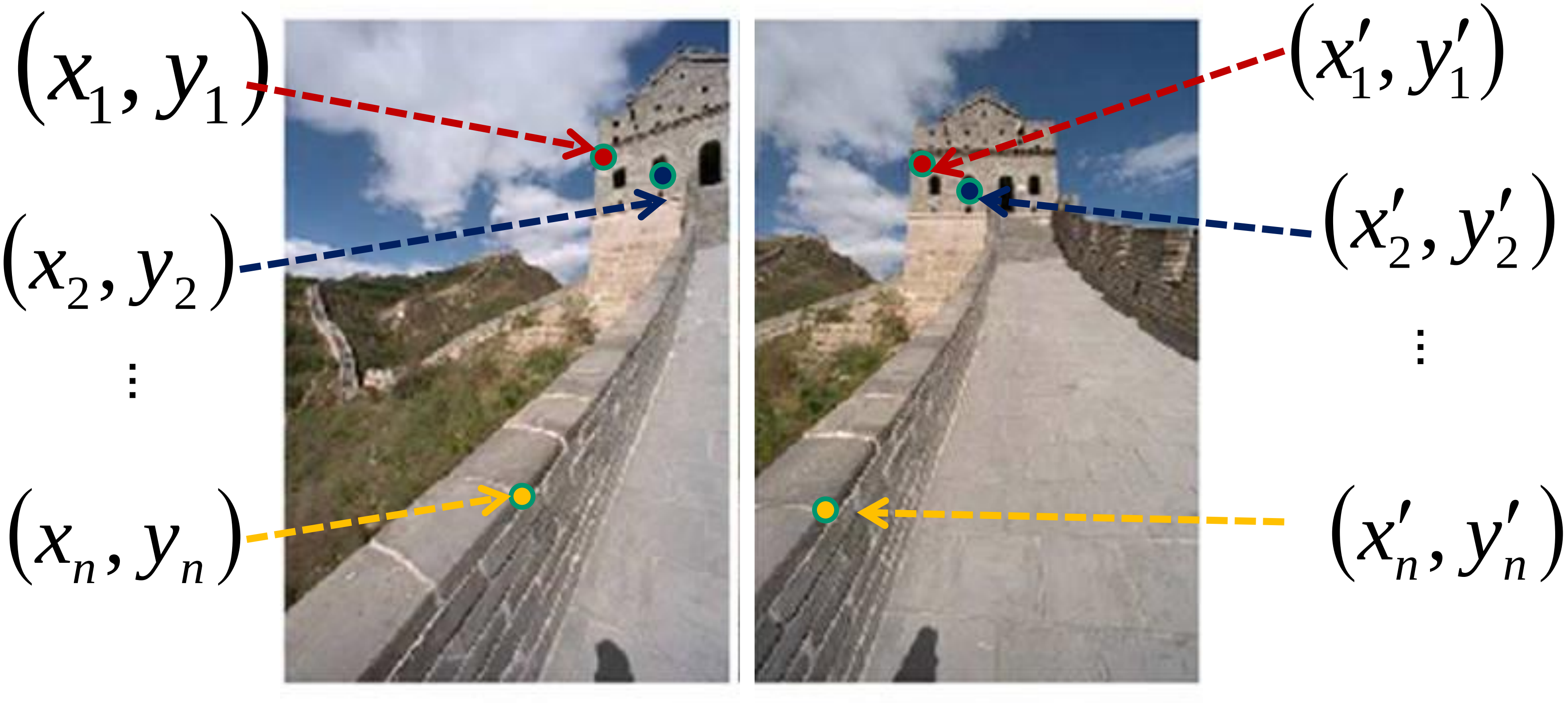
How to stitch together panorama (mosaic)?

Basic Procedure

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

Adapted from Steve Seitz

Computing the 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…

Kristen Grauman

Computing the homography

- Assume we have four matched points: How do we compute homography **H**?

p’=Hp

$$\mathbf{p}' = \begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad \mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}$$

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Can set scale factor $h_9 = 1$.
So, there are 8 unknowns.
Need at least 8 eqs, but the more the better…

A

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \mathbf{h} = \mathbf{0}$$

DEMO

Adapted from Derek Hoiem, Kristen Grauman

Derivation: http://www.cse.psu.edu/~rtc12/CSE486/lecture16.pdf

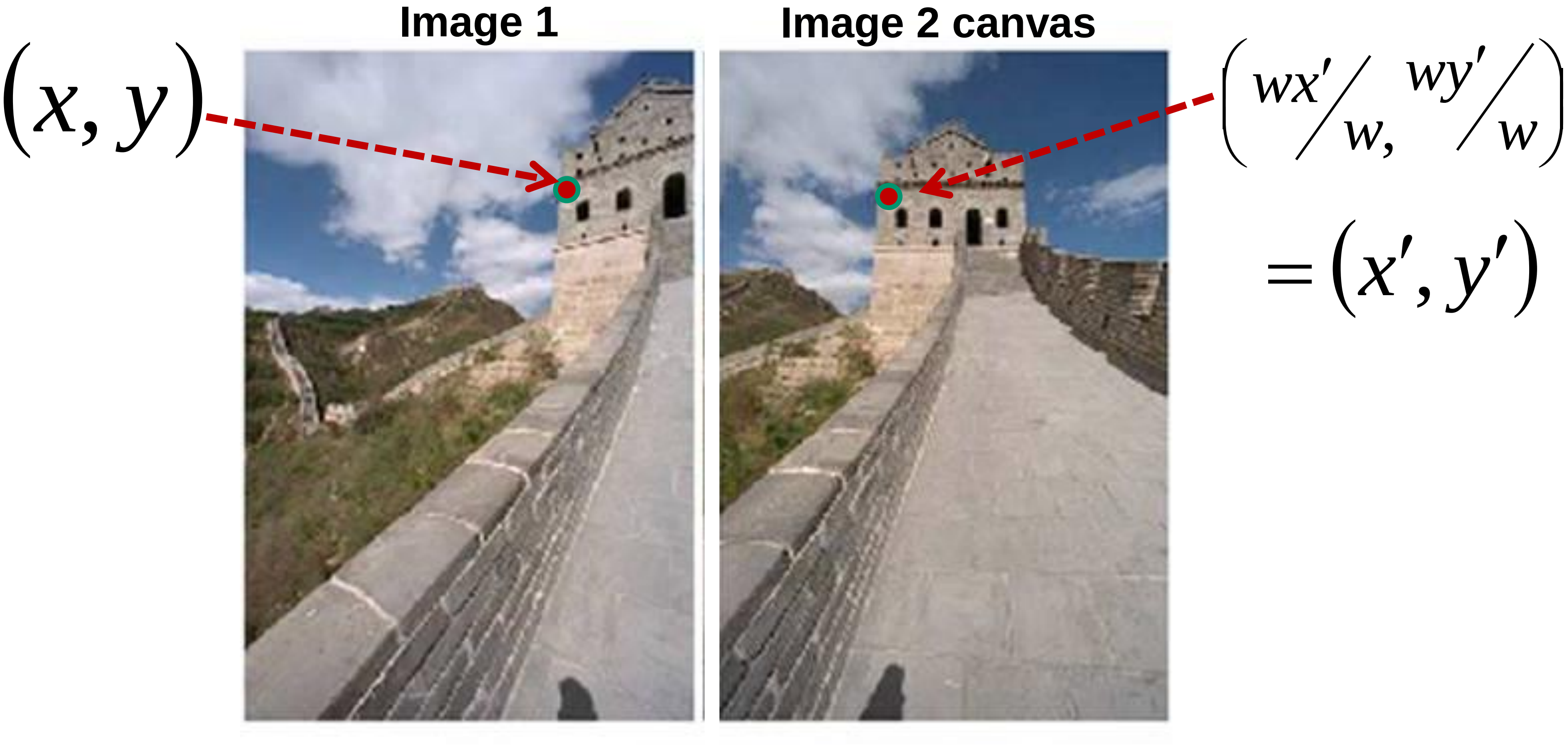
How to stitch together panorama (mosaic)?

Basic Procedure

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

Adapted from Steve Seitz

Transforming the second image



$$(x, y)$$

$$\left({}^{wx'}\!/_{w}, {}^{wy'}\!/_{w} \right) = (x', y')$$

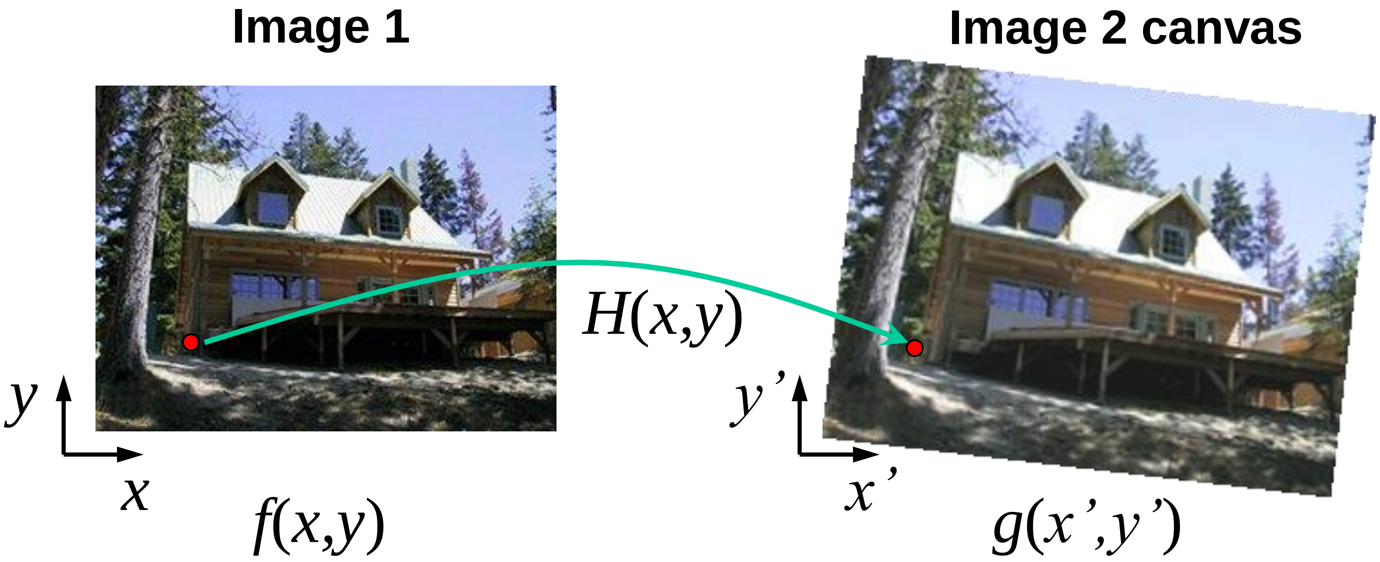
To **apply** a given homography **H**

- Compute **p’** = **Hp** (regular matrix multiply)
- Convert **p’** from homogeneous to image coordinates

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

Modified from Kristen Grauman

Transforming the second image



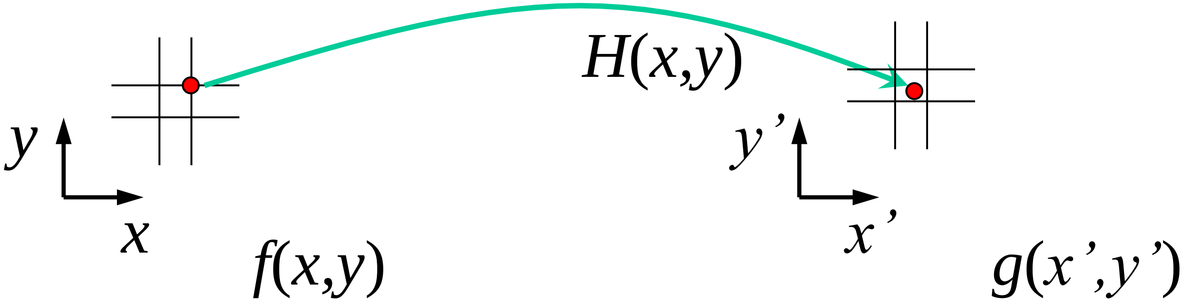
Forward warping:

Send each pixel $f(x,y)$ to its corresponding location

$(x',y') = H(x,y)$ in the right image

Modified from Alyosha Efros

Transforming the second image



Forward warping:

Send each pixel *f*(*x*,*y*) to its corresponding location

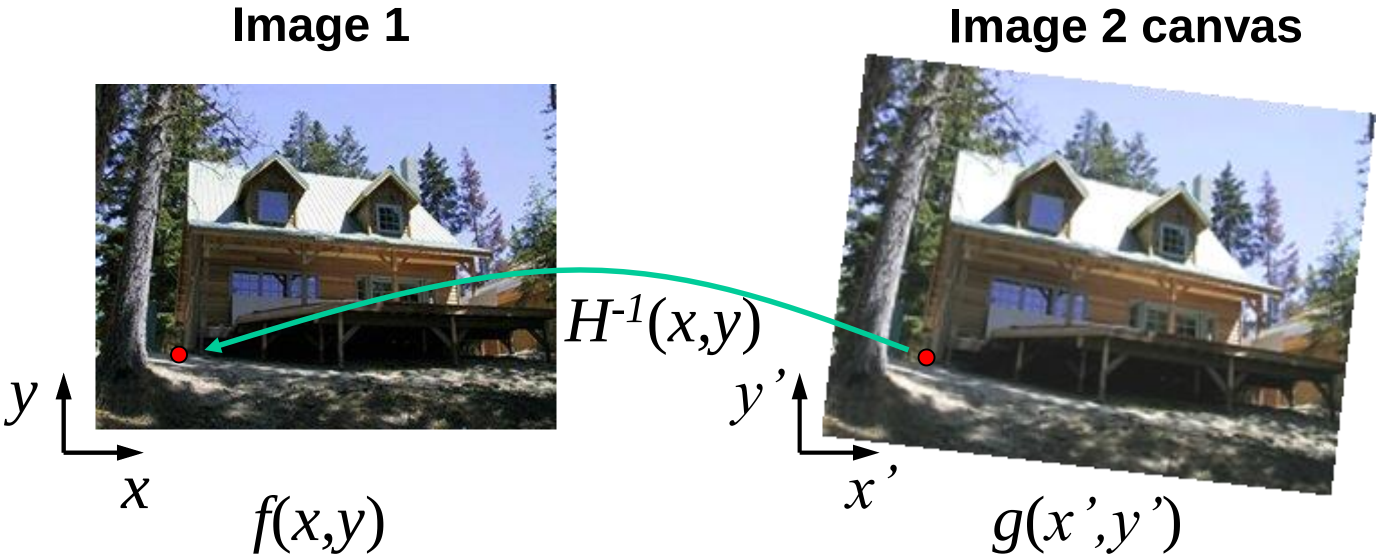
$(x',y') = H(x,y)$ in the right image

Q: what if pixel lands “between” two pixels?

A: distribute color among neighboring pixels (x’,y’)

Alyosha Efros

Transforming the second image



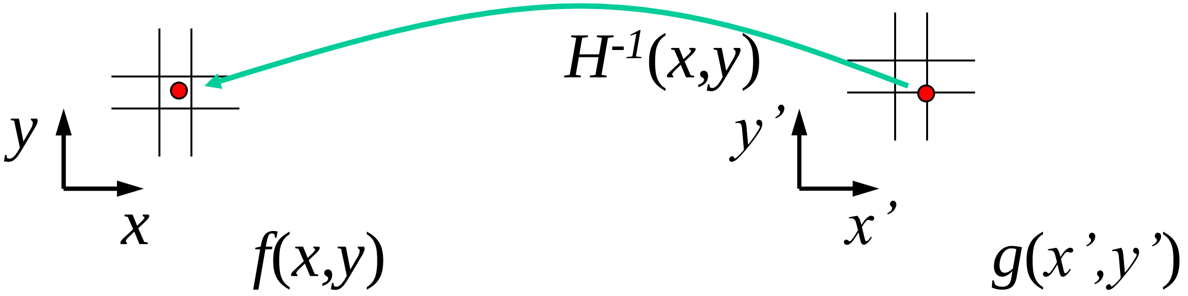
Inverse warping:

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

$(x,y) = H^{-1}(x',y')$ in the left image

Modified from Alyosha Efros

Transforming the second image



Inverse warping:

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

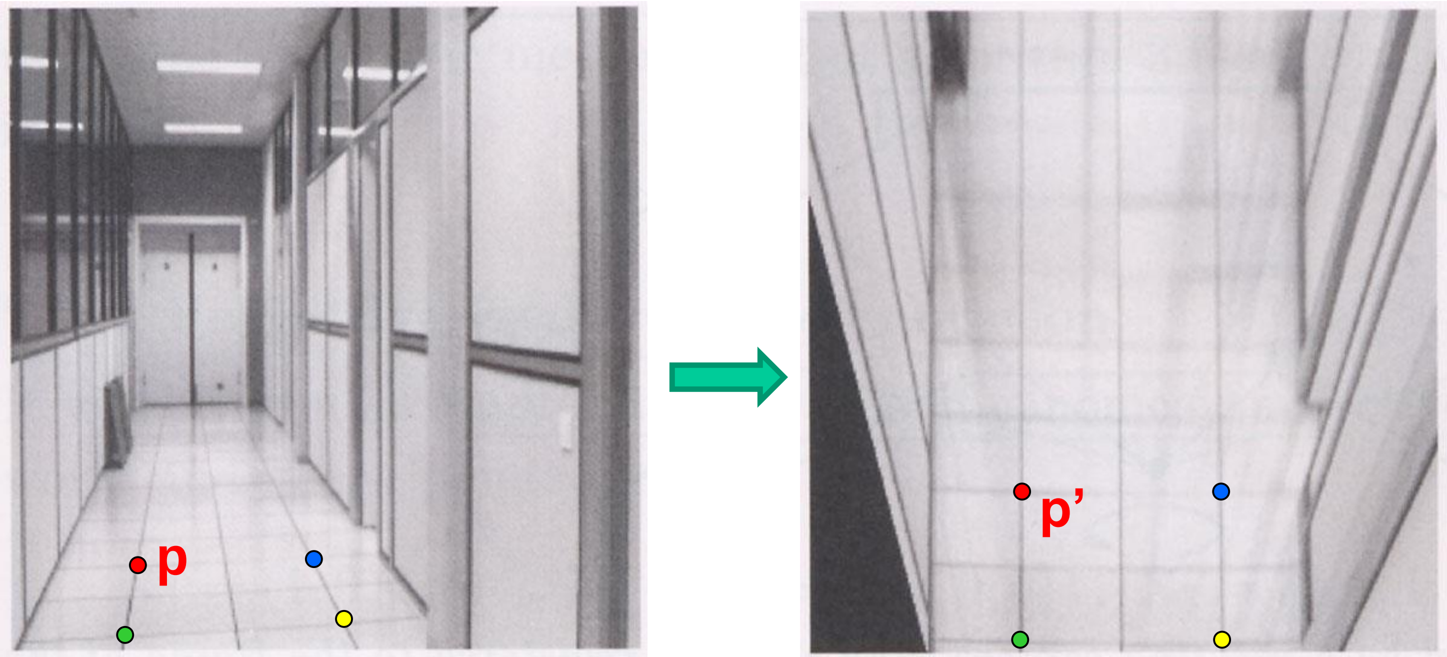
$(x,y) = H^{-1}(x',y')$ in the left image

Q: what if pixel comes from “between” two pixels?

A: *interpolate* color value from neighbors

Alyosha Efros

Homography example: Image rectification



To unwarp (rectify) an image solve for homography **H** given **p** and **p’**: **p’=Hp**

Derek Hoiem

Summary of affine/projective transforms

- Write **2d transformations** as matrix-vector multiplication (including translation when we use homogeneous coordinates)
- **Fitting transformations**: solve for unknown parameters given corresponding points from two views – linear, affine, projective (homography)
- **Mosaics**: uses homography and image warping to merge views taken from same center of projection
  - Perform **image warping** (forward, inverse)

Adapted from Kristen Grauman

Next: Stereo vision

- Homography: Same camera center, but camera rotates
- Stereo vision: Camera center is not the same (we have multiple cameras)

- Epipolar geometry
  - Relates cameras from two positions/cameras

- Stereo depth estimation
  - Recover depth from disparities between two images

Adapted from Derek Hoiem

Stereo photography and stereo viewers

Take two pictures of the same subject from two slightly different viewpoints and display so that each eye sees only one of the images.

Invented by Sir Charles Wheatstone, 1838

Image from fisher-price.com

© Copyright 2001 Johnson-Shaw Stereoscopic Museum

Kristen Grauman

Depth from stereo for computers

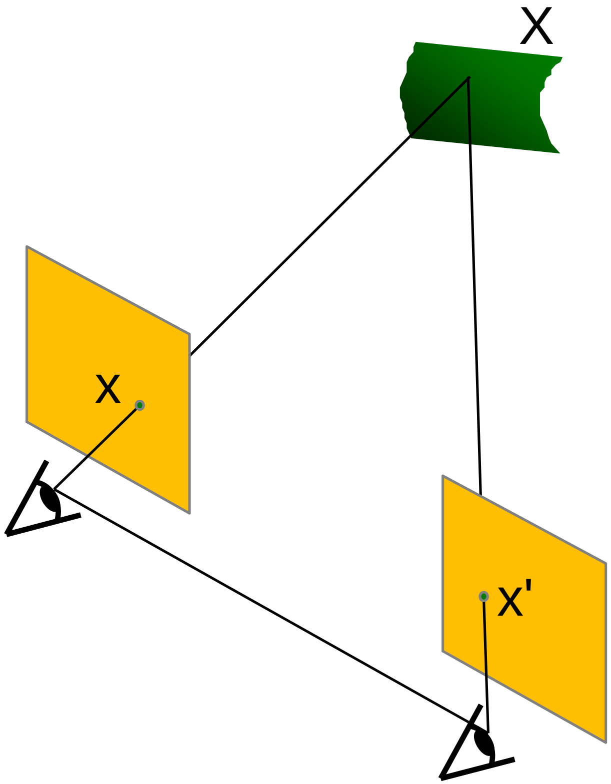
Two cameras, simultaneous views

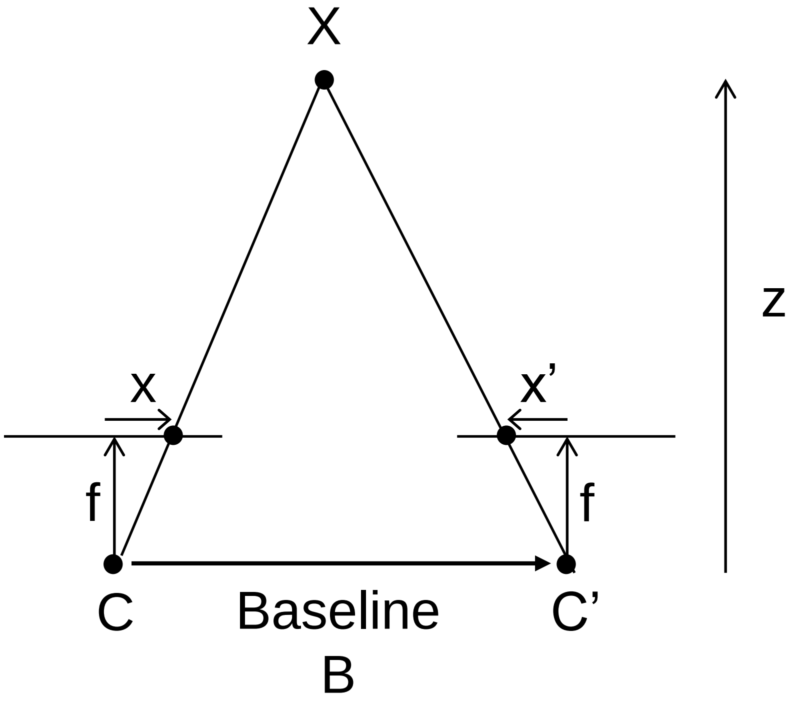
Single moving camera and static scene

Kristen Grauman

Depth from stereo

- Goal: recover depth by finding image coordinate x' that corresponds to x

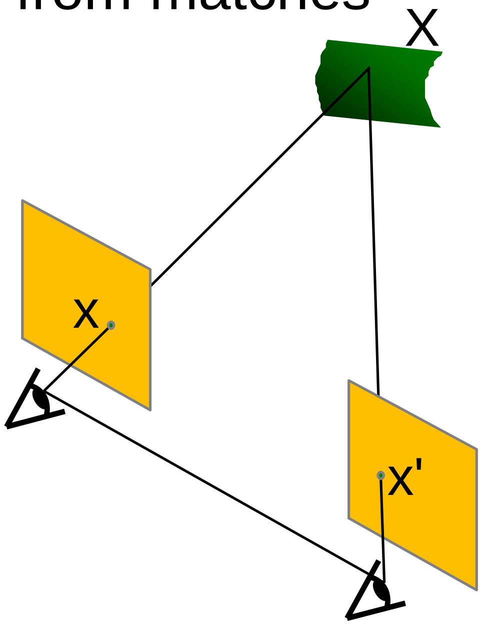




Derek Hoiem

Depth from stereo

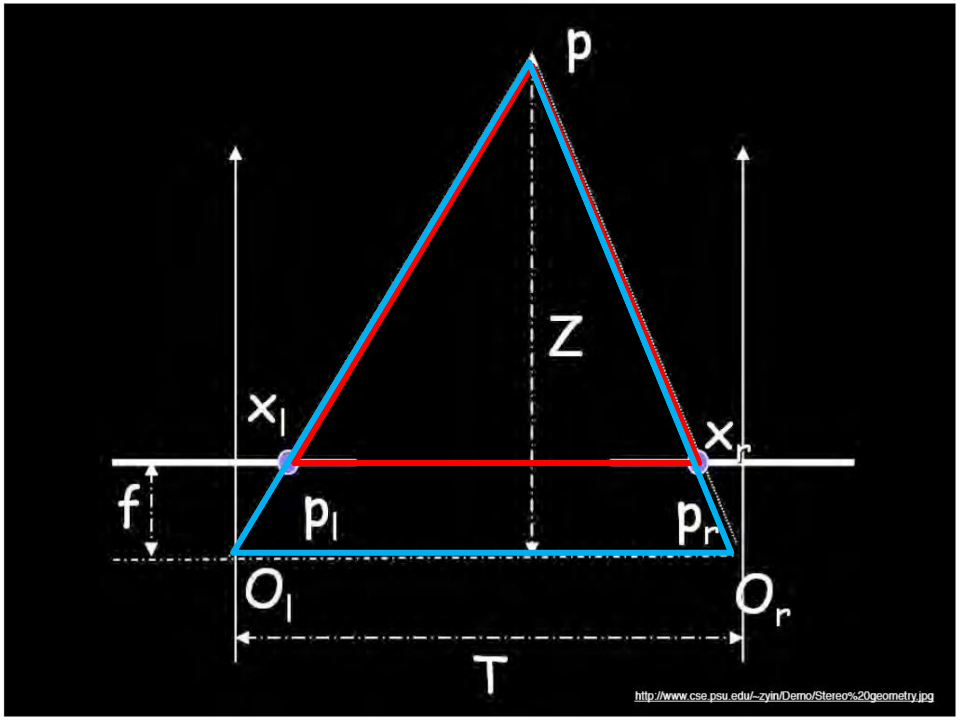
- Goal: recover depth by finding image coordinate x' that corresponds to x
- Sub-Problems
    1. Calibration: How do we recover the relation of the cameras (if not already known)?
    2. Correspondence: How do we search for the matching point x'?
    3. Estimate depth from matches



Derek Hoiem

Geometry for a simple stereo system

- Assume parallel optical axes, known camera parameters (i.e., calibrated cameras). **What is expression for Z?**



Similar triangles (p_l, P, p_r) and (O_l, P, O_r):

$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

depth

$$Z = f\frac{T}{x_r - x_l}$$

disparity

Depth is inversely proportional to disparity.

Adapted from Kristen Grauman

Depth from disparity

- We have two images from different cameras.
- If we could find the **corresponding points** in two images, we could **estimate relative depth**…
- How do we match a point in the first image to a point in the second **efficiently**?

image I(x,y)

Disparity map D(x,y)

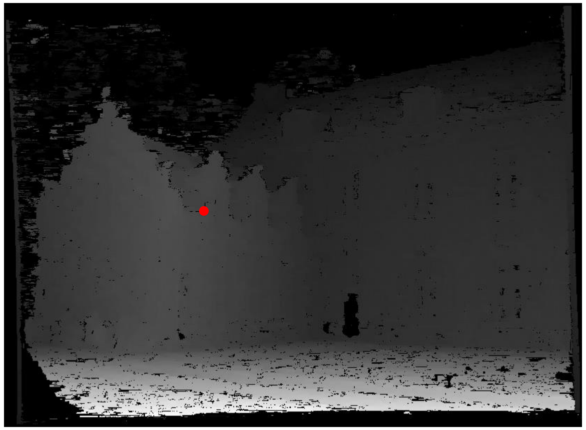
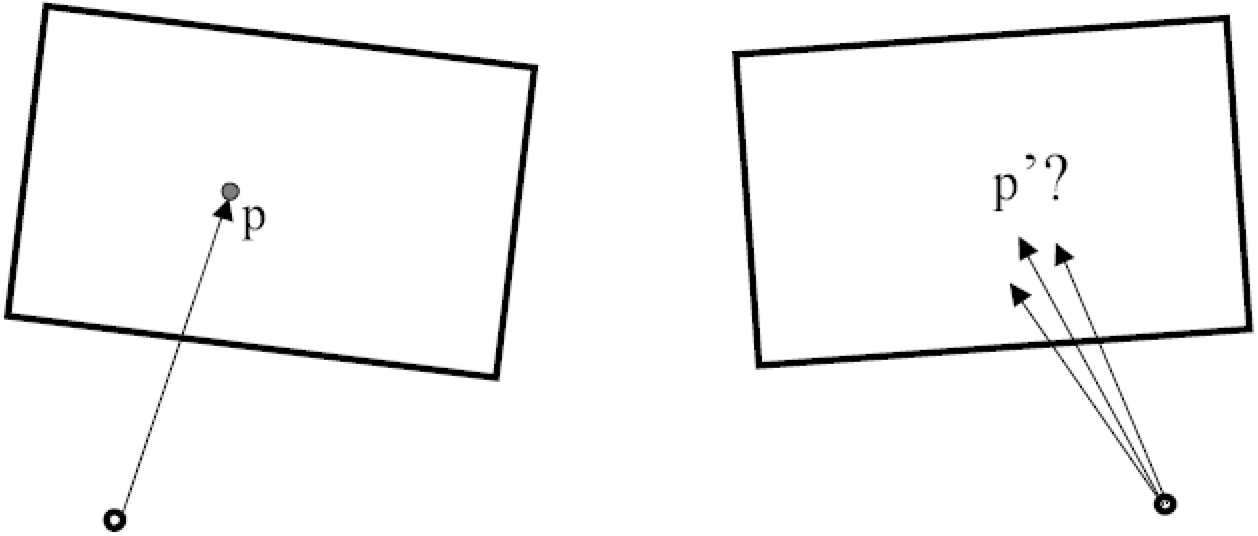


image I´(x´,y´)

Kristen Grauman

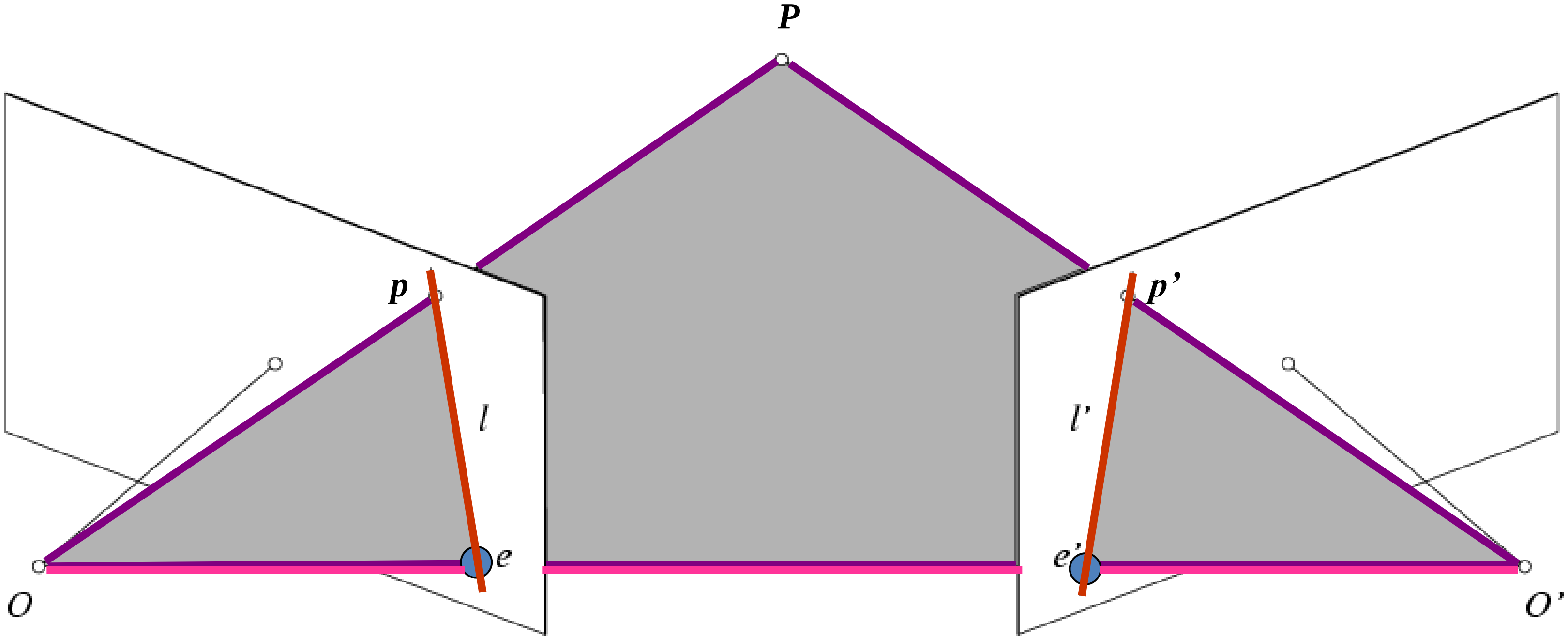
Stereo correspondence constraints



- Given p in left image, where can corresponding point p' be?

Kristen Grauman

Epipolar geometry: notation



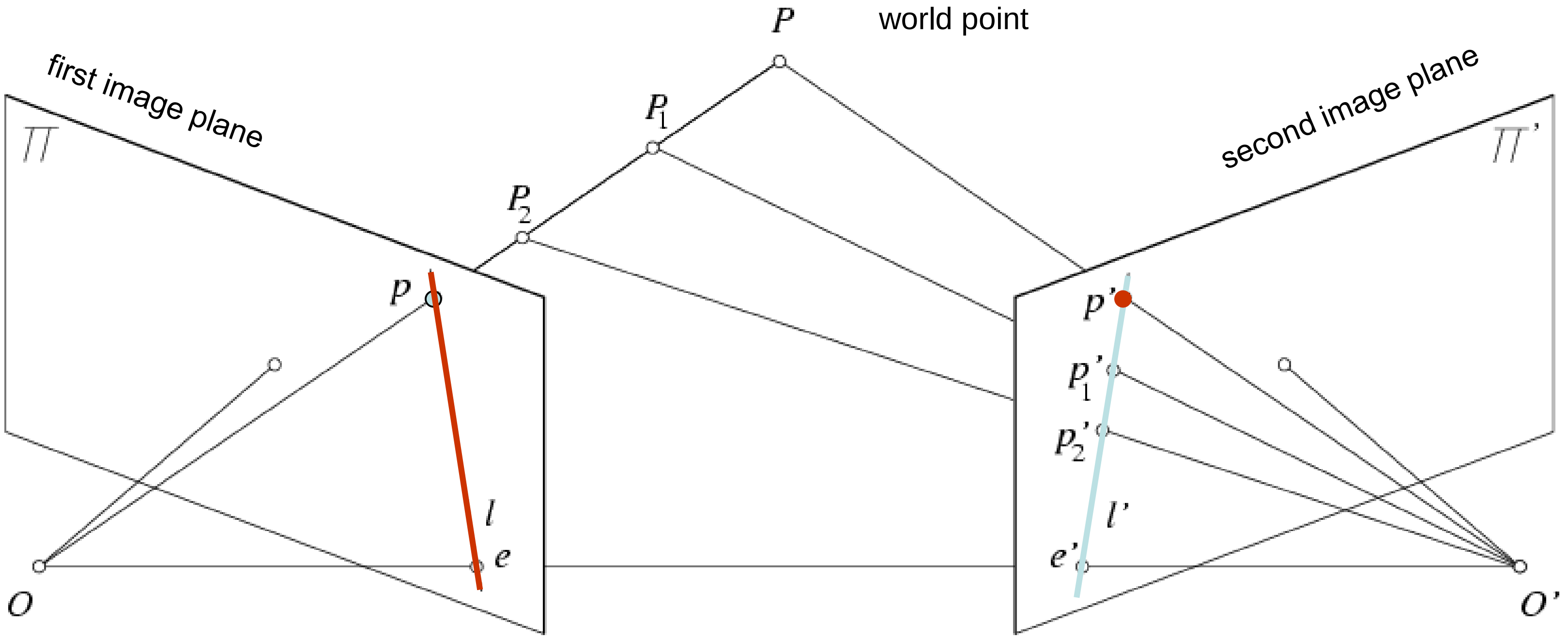
- **Baseline** – line connecting the two camera centers

- **Epipoles**
= intersections of baseline with image planes
= projections of the other camera center

- **Epipolar Plane** – plane containing baseline
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)

Adapted from Derek Hoiem

Epipolar constraint



Geometry of two views constrains where the corresponding pixel for some image point in the first view must occur in the second view.

- It must be on the line where (1) the plane connecting the world point and optical centers, and (2) the image plane, intersect.
- Potential matches for *p* have to lie on the corresponding line *l'*.
- Potential matches for *p'* have to lie on the corresponding line *l*.

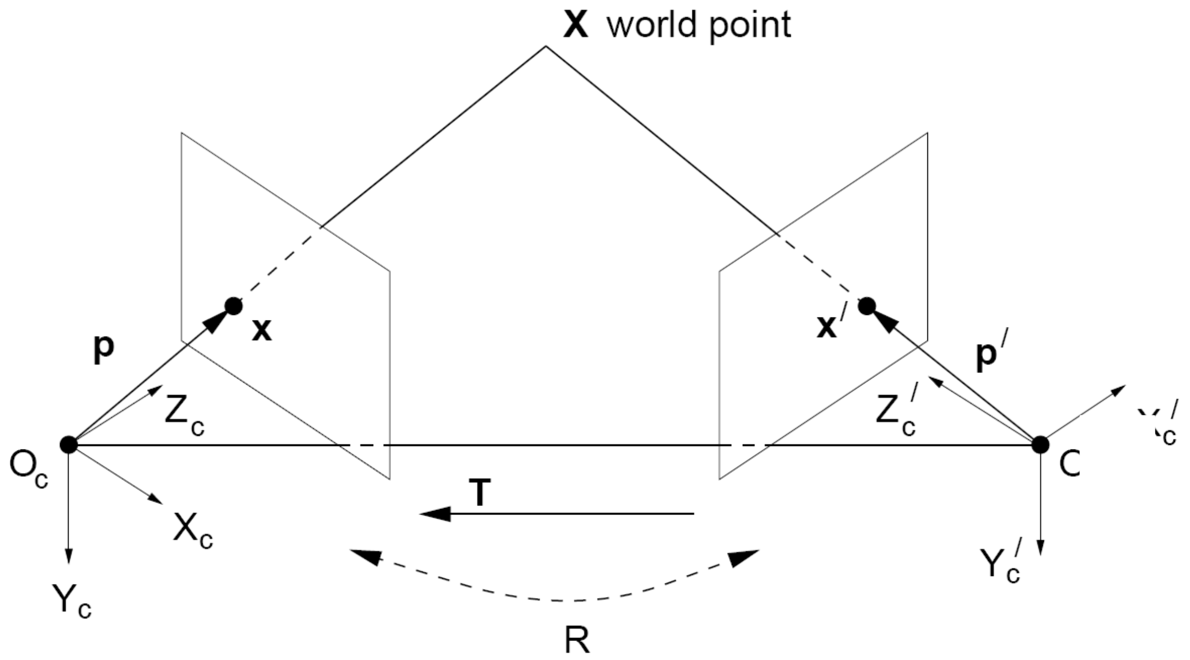
Adapted from Kristen Grauman, Derek Hoiem

Epipolar constraint

The epipolar constraint is useful because it reduces the correspondence problem to a 1D search along an epipolar line.

Kristen Grauman, image from Andrew Zisserman

Stereo geometry, with calibrated cameras



- If the stereo rig is calibrated, we know how to **rotate** and **translate** camera reference frame 1 to get to camera reference frame 2
  - Rotation: 3x3 matrix **R**; translation: 3x1 vector **T**.

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}$$

(See hidden slides for how we get to the next slide.)

Adapted from Kristen Grauman

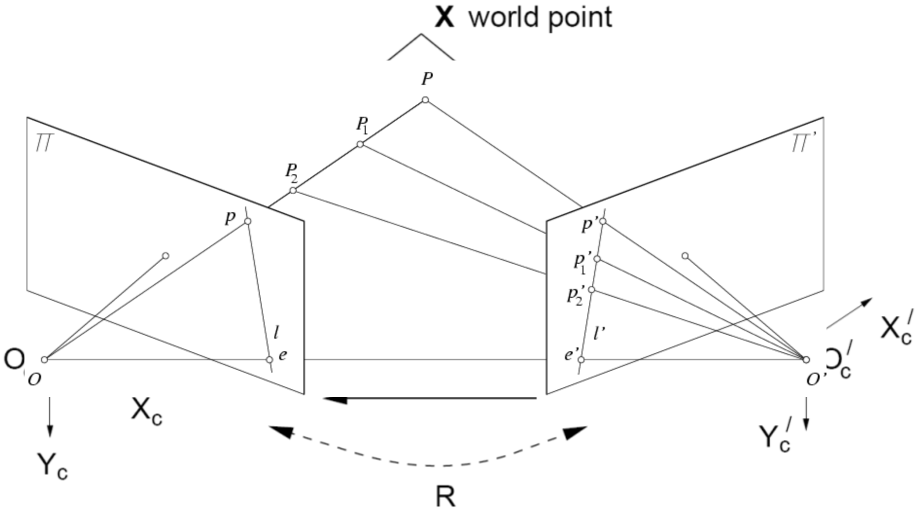
Essential matrix

$$\boxed{\mathbf{X}'\cdot\left(\mathbf{T}\times\mathbf{RX}\right)=0}$$

$$\mathbf{X}'\cdot\left([\mathrm{T}_x]\mathbf{RX}\right)=0$$

Let $\mathbf{E}=[\mathrm{T}_x]\mathbf{R}$

$$\mathbf{X}'\cdot\mathbf{EX}=\mathbf{X}'^T\mathbf{EX}=0$$



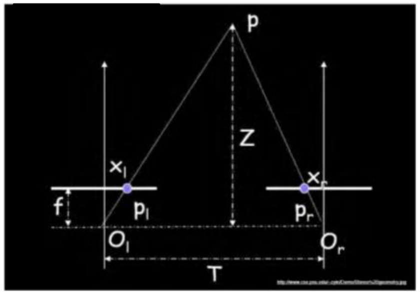
E is called the **essential matrix**, and it relates corresponding image points between both cameras, given the rotation and translation.

Before we said: If we observe a point in one image, its position in other image is constrained to lie on line defined by above. It turns out that:

- $E^T x$ is the epipolar line l' through x' in the second image, corresponding to x.
- Ex' is the epipolar line l through x in the first image, corresponding to x'.

Adapted from Kristen Grauman, Derek Hoiem

Essential matrix example: parallel cameras



$$\mathbf{R} =$$

$$\mathbf{T} =$$

$$\mathbf{E} = [\mathrm{T}_{\mathbf{x}}]\mathbf{R} =$$

$$\mathbf{p} = [x, y, f]$$

$$\mathbf{p'} = [x', y', f]$$

$$\mathbf{p}'^{\mathrm{T}}\mathbf{E}\mathbf{p} = 0$$

For the parallel cameras, image of any point must lie on same horizontal line in each image plane.

Kristen Grauman

image I(x,y)

Disparity map D(x,y)

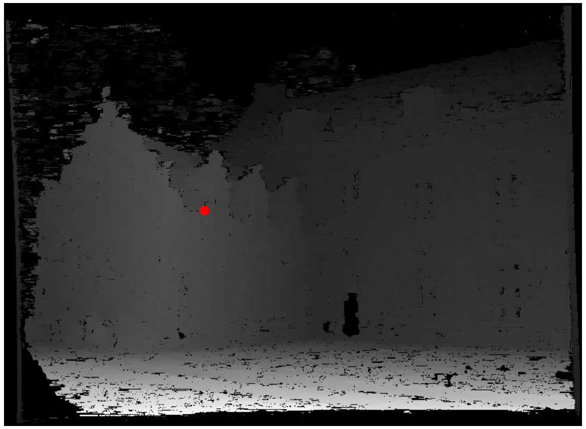
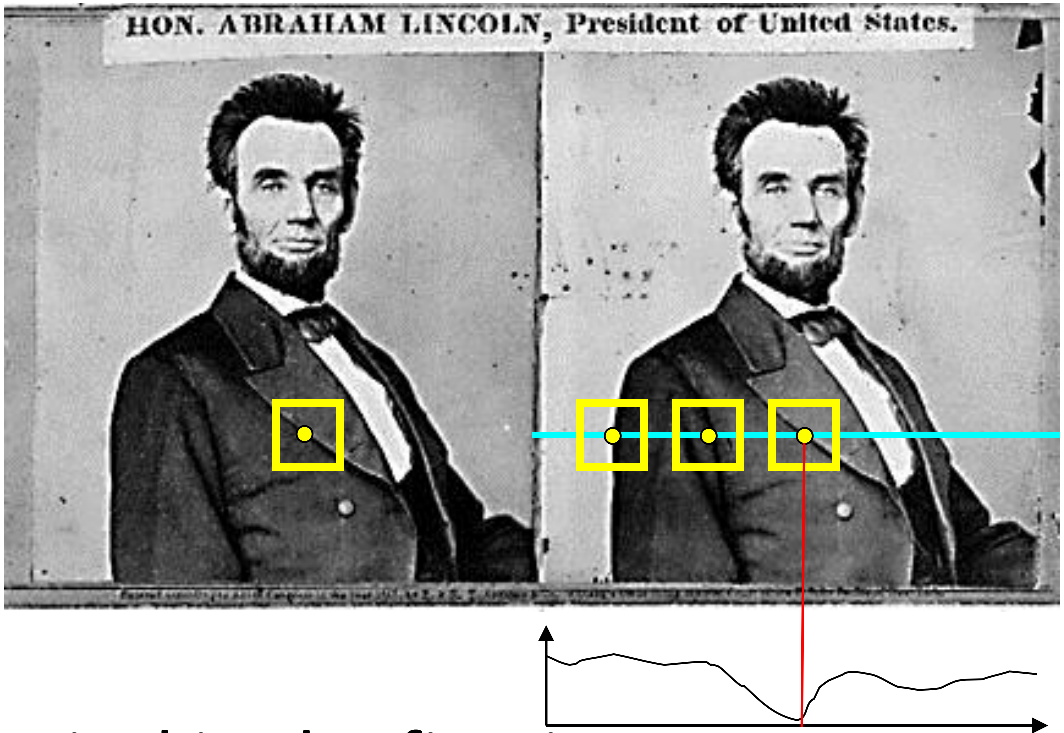


image I´(x´,y´)

$$(x'\boxed{,y')}=(x+D(x,y\boxed{),y)}$$

Adapted from Kristen Grauman

Basic stereo matching algorithm

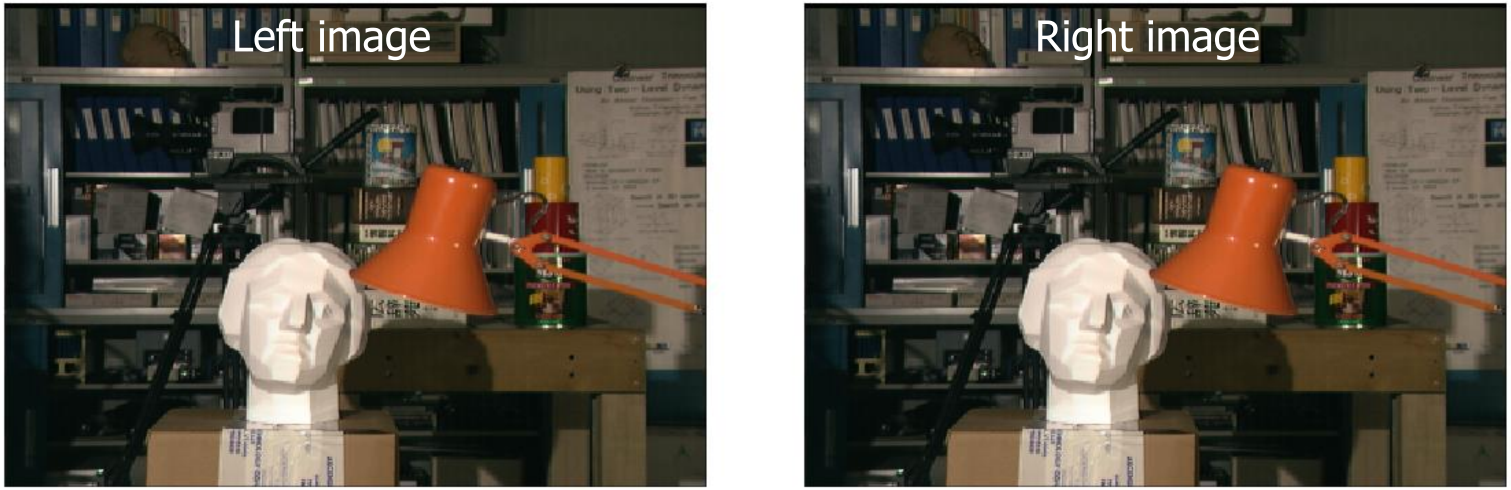


- For each pixel in the first image
  - Find corresponding epipolar scanline in the right image
  - Search along epipolar line and pick the best match x’: slide a window along the right scanline and compute Euclidean distance between contents of that window with the reference window in the left image; take the window corresponding to the minimum as the match
  - Compute disparity x-x’ and set depth(x) = f*T/(x-x’)

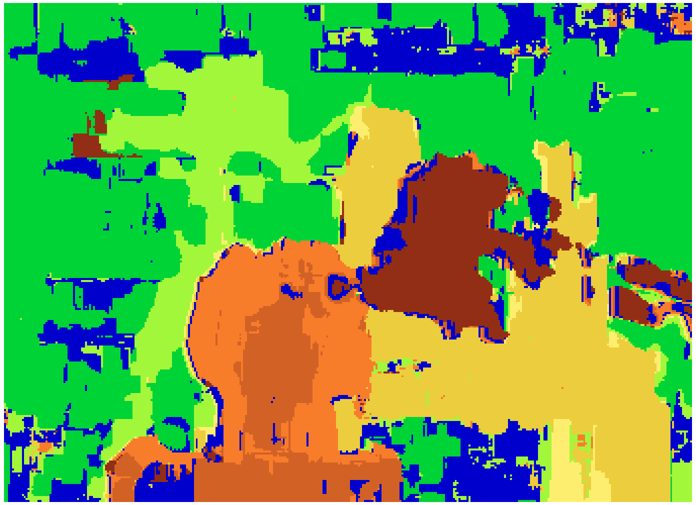
Adapted from Derek Hoiem

Results with window search

Data



Predicted depth



Ground truth

Derek Hoiem

Summary of stereo vision

- **Epipolar geometry**
  - Epipoles are intersection of baseline with image planes
  - Matching point in second image is on a line passing through its epipole
  - Epipolar constraint limits where points from one view will be imaged in the other, which makes search for correspondences quicker
  - Essential matrix E maps from a point in one image to a line (its epipolar line) in the other

- **Stereo depth estimation**
  - Find corresponding points along epipolar scanline
  - Estimate disparity (depth is inverse to disparity)

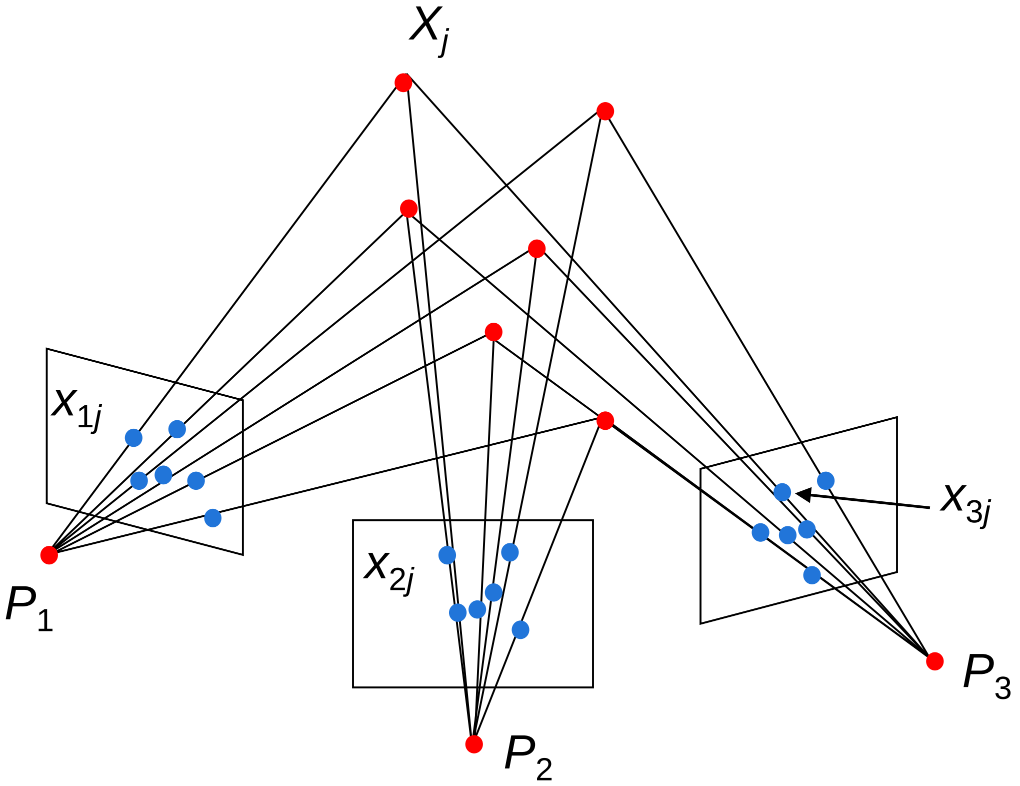
Adapted from Kristen Grauman and Derek Hoiem

Projective structure from motion

- Given: *m* images of *n* fixed 3D points

$$\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n$$

- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* corresponding 2D points $\mathbf{x}_{ij}$



Svetlana Lazebnik

Photo tourism

Noah Snavely, Steven M. Seitz, Richard Szeliski, "Photo tourism: Exploring photo collections in 3D," SIGGRAPH 2006

http://phototour.cs.washington.edu/

3D from multiple images

Sameer Agarwala, Noah Snavely, Ian Simon, Steven M. Seitz, Richard Szeliski, "Building Rome in a Day," ICCV 2009