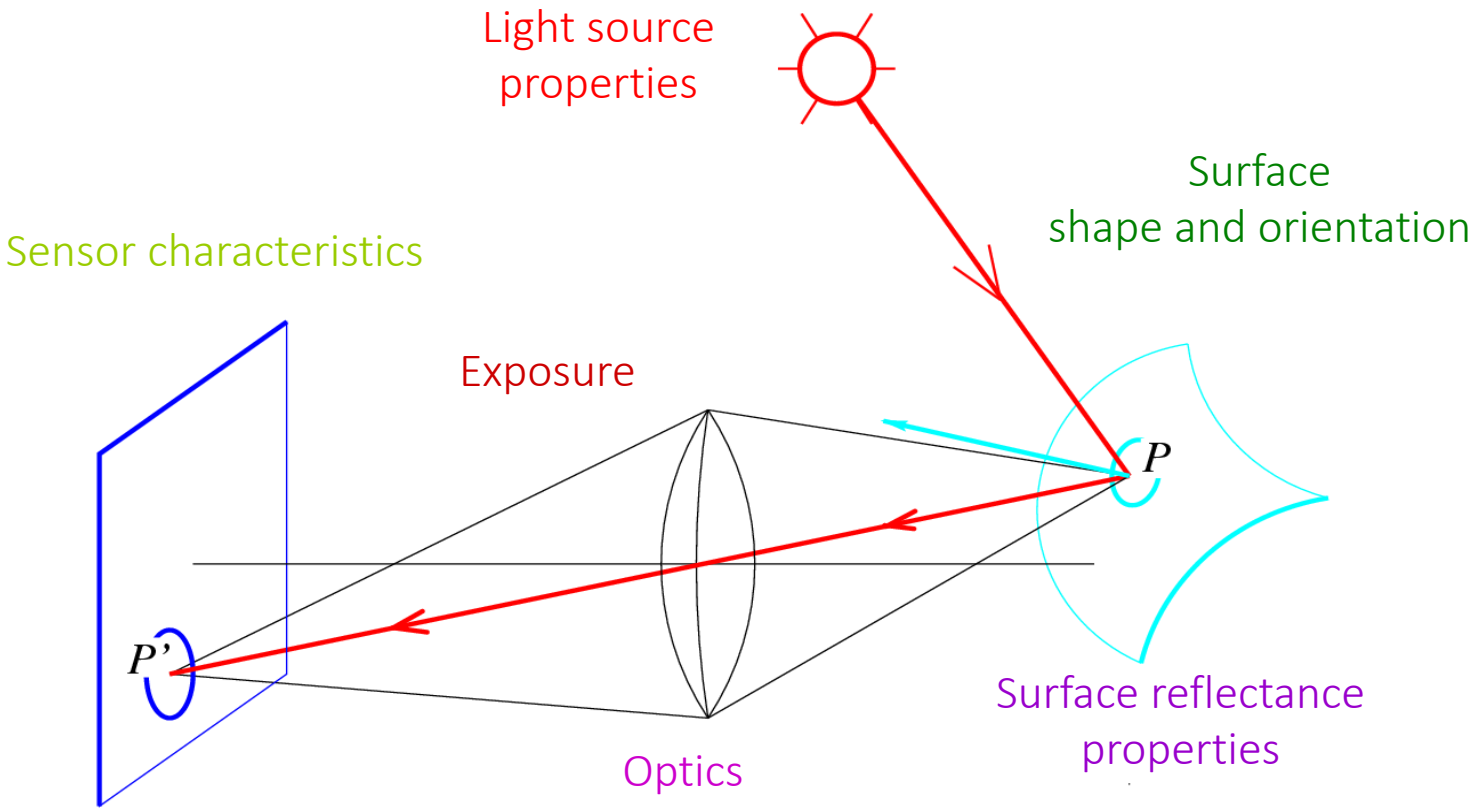
Projective Geometry

Kuk-Jin Yoon

Visual Intelligence Lab.
Department of Mechanical Engineering

KAIST

Recall: Image Formation



Applications of Projective Geometry

Vermeer's Music Lesson

Reconstructions by Criminisi et al.

Perspective Distortion

- What does a sphere project to?

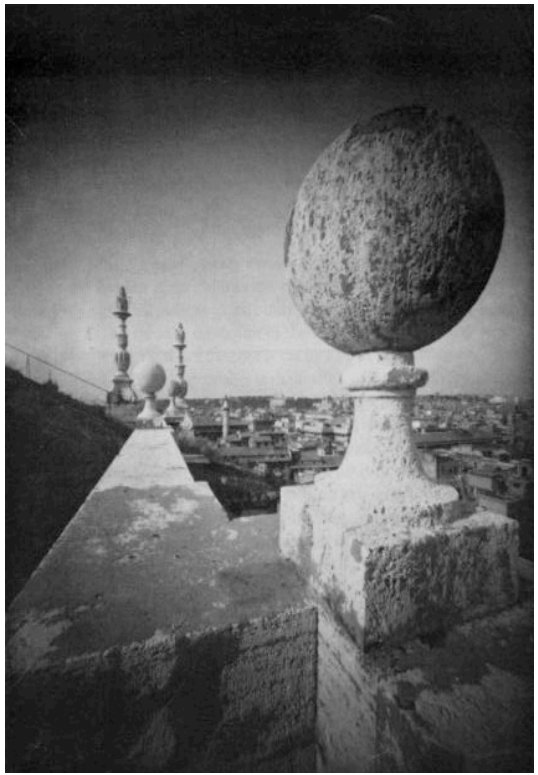
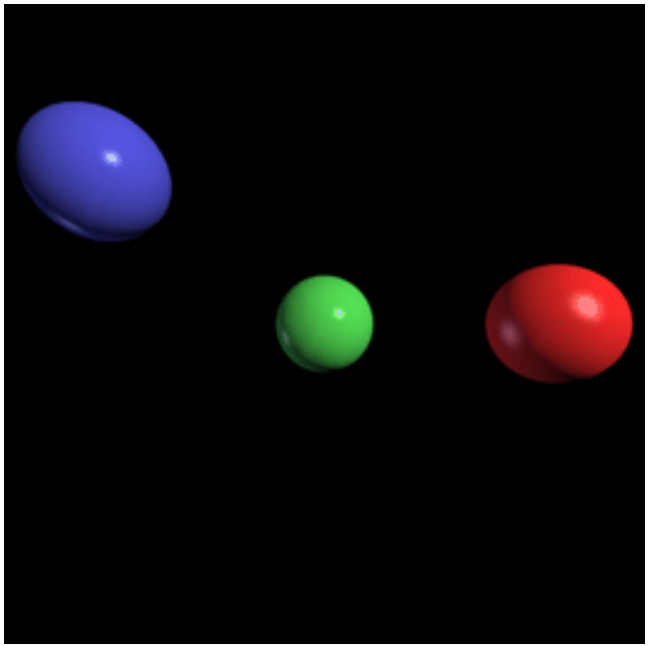


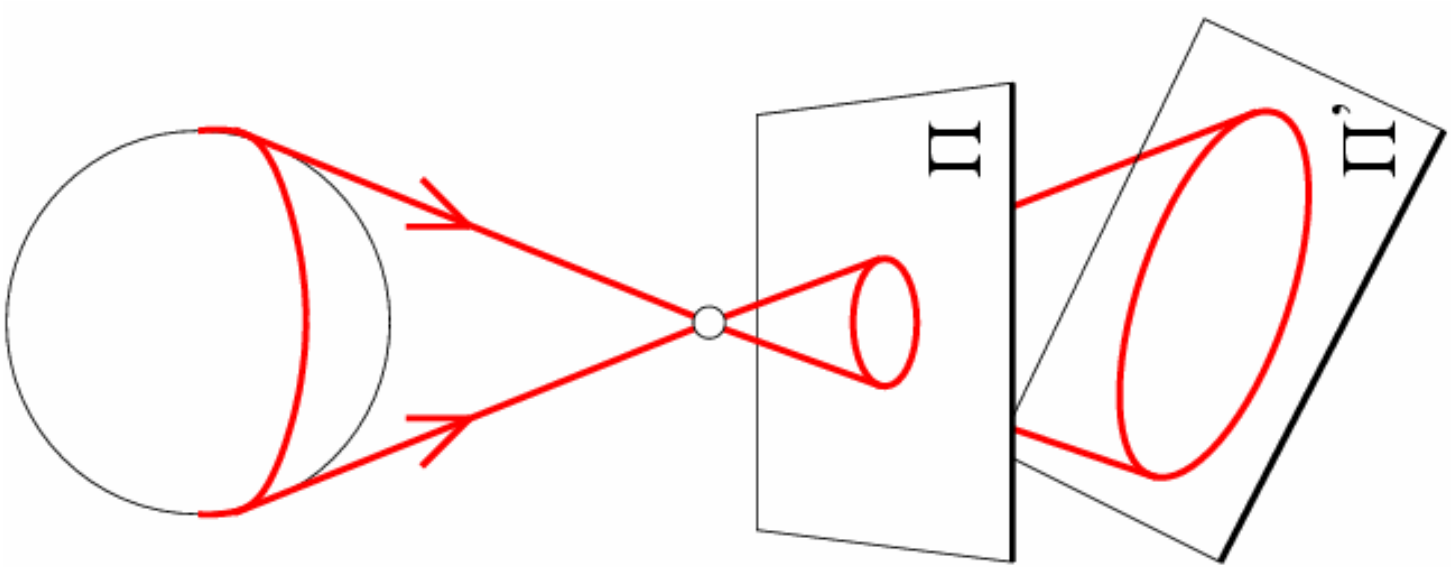
Image source: F. Durand



This is view of 3 spheres, rendered with a 90° field of view. All three spheres are the same distance from the camera.

Perspective Distortion

- What does a sphere project to?



Projective Plane

- Definition of real projective plane $\mathbf{P}^2(\mathbf{R})$
  - The set of all lines in $\mathbf{R}^3$ passing through the origin (0, 0, 0) or
  - The set of equivalence classes of $\mathbf{R}^3\backslash(0, 0, 0)$, i.e. 3D-space without the origin, where two points $P = (x, y, z)$ and $P^* = (x^*, y^*, z^*)$ are equivalent iff there is a nonzero real number λ such that $P = \lambda P^*$.
  - More generally,

$$\mathbf{P}^{\boldsymbol{n}}(\mathbf{R}) = (\mathbf{R}^{\boldsymbol{n+1}}\backslash\{\mathbf{0}\})/\sim$$

where ~ is the equivalence relation defined by: $(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n)$ if there is a non-zero real number λ such that $(x_0, \ldots, x_n) = (\lambda y_0, \ldots, \lambda y_n)$.

KAIST

Recall: Homogeneous Coordinates in Image Formation

- In the projection process, two different points (X, Y, Z) and (aX, aY, aZ) (a≠0) are mapped onto the same point in the image plane.
  - Given a point (x, y) on the Euclidean plane, and choosing a non-zero real number Z, the triple (X, Y, Z) = (xZ, yZ, Z) is called a set of homogeneous coordinates for the point.
  - Projective (or homogeneous) coordinates of a point in projective space

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

  - Converting from homogeneous coordinates

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

Recall: Homogeneous Coordinates

- Homogeneous representation of lines

$$ax + by + c = 0 \quad (a,b,c)^T$$

$$(ka)x + (kb)y + kc = 0, \forall k \neq 0 \quad (a,b,c)^T \sim k(a,b,c)^T$$

- Homogeneous representation of points

$$\mathrm{x} = (x, y)^T \text{ on } 1 = (a,b,c)^T \text{ if and only if } ax + by + c = 0$$

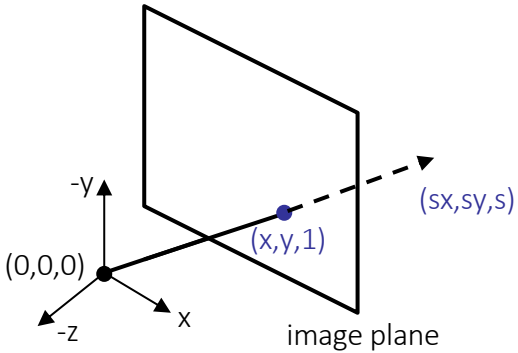
$$(x,y,1)(a,b,c)^T = (x,y,1)1 = 0 \quad (x, y, 1)^T \sim k(x, y, 1)^T, \forall k \neq 0$$

  - Why do we need homogeneous coordinates?
    - We can represent points at infinity, homographies, perspective projection, multi-view relationships

KAIST

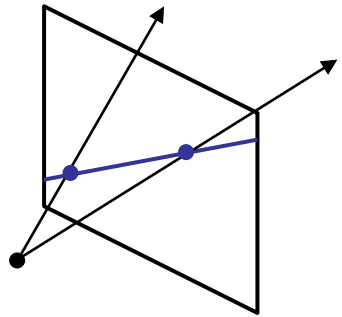
Point and Line in Projective Space

- A point in the image is a *ray* in projective space



  - Each point (x,y) on the plane is represented by a ray (sx,sy,s)
    - all points on the ray are equivalent: (x, y, 1) ≅ (sx, sy, s)

- A line is a plane of rays through origin

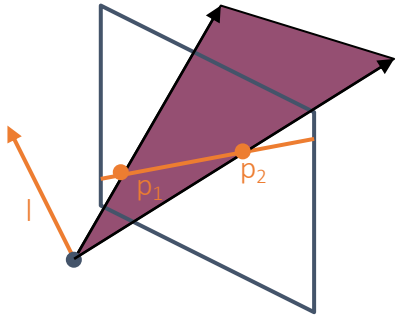


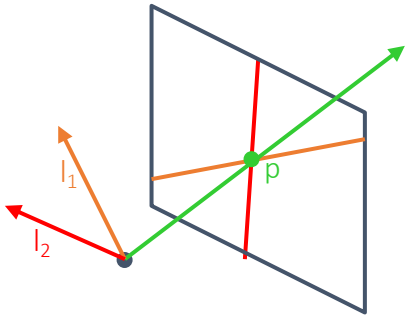
  - All rays (x,y,z) satisfying: ax + by + cz = 0

$$0 = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Point and Line Duality

A line l is a homogeneous 3-vector
It is ⊥ to every point (ray) p on the line: $l^T p=0$





- line l spanned by rays p_1 and p_2
  - l is ⊥ to p_1 and p_2 $\Rightarrow$ $l = p_1 \times p_2$
  - l is the plane normal
- intersection of two lines l_1 and l_2
  - p is ⊥ to l_1 and l_2 $\Rightarrow$ $p = l_1 \times l_2$

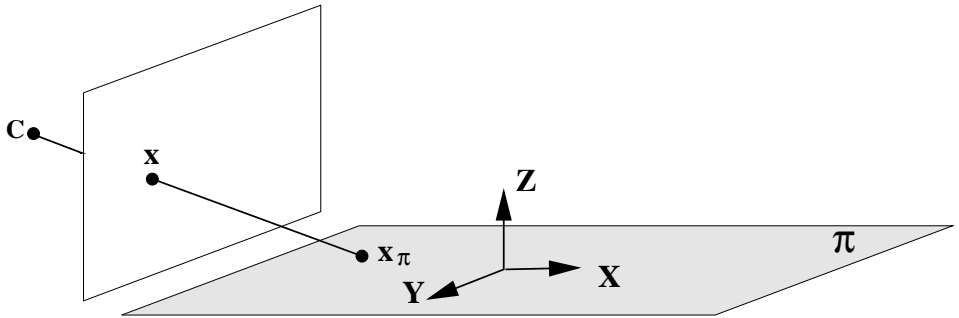
Points and lines are dual in projective space

- given any formula, can switch the meanings of points and lines to get another formula

KAIST

Plane Projective Transformation

Source: Multiple View Geometry
Richard Hartley and Andrew Zisserman
CVPR 1999



Choose the world coordinate system such that the plane of the points has zero Z coordinate. Then the 3×4 matrix $\mathtt{P}$ reduces to

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

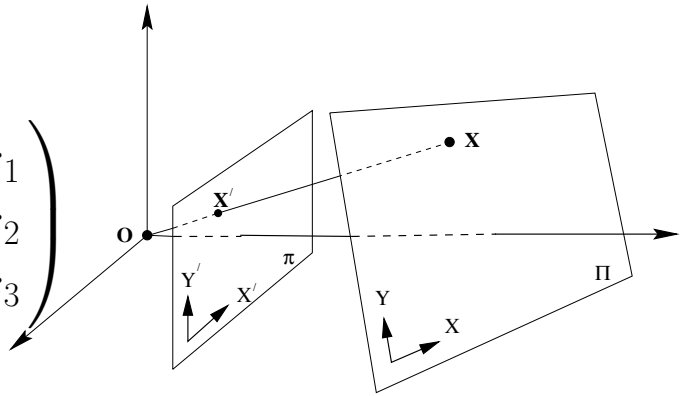
which is a 3×3 matrix representing a general plane to plane projective transformation.

KAIST

Plane Projective Transformation

Source: Multiple View Geometry
Richard Hartley and Andrew Zisserman
CVPR 1999

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$



or $\mathbf{x}' = \mathtt{H}\mathbf{x}$, where $\mathtt{H}$ is a 3×3 non-singular homogeneous matrix.

- This is the most general transformation between the world and image plane under imaging by a perspective camera.
- It is often only the 3×3 form of the matrix that is important in establishing properties of this transformation.
- A projective transformation is also called a “homography” and a “collineation”.
- $\mathtt{H}$ has 8 degrees of freedom.

Image of Planes

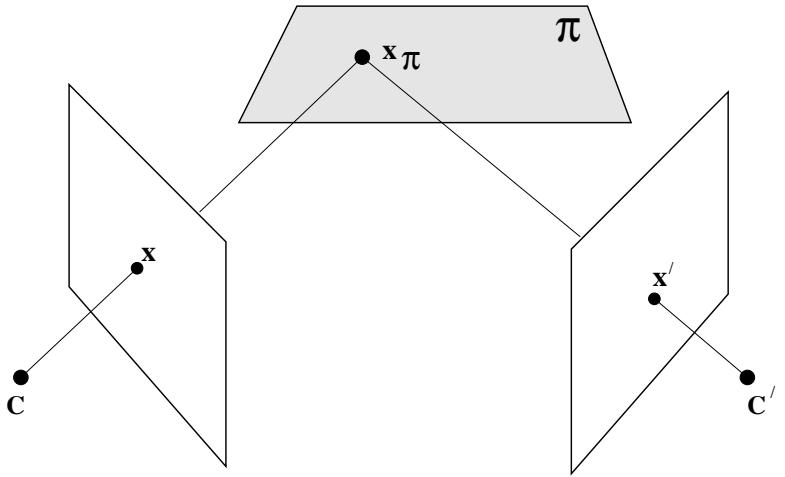
Source: Multiple View Geometry
Richard Hartley and Andrew Zisserman
CVPR 1999

Projective transformation between images induced by a plane

$$\mathbf{x} = \mathrm{H}_{1\pi}\mathbf{x}_\pi \qquad \mathbf{x}' = \mathrm{H}_{2\pi}\mathbf{x}_\pi$$

$$\mathbf{x}' = \mathrm{H}_{2\pi}\mathbf{x}_\pi$$

$$= \mathrm{H}_{2\pi}\mathrm{H}_{1\pi}^{-1}\mathbf{x} = \mathrm{H}\mathbf{x}$$



- H can be computed from the correspondence of four points on the plane.

2D Projective Transformations

- Definition
  - A *projectivity* is an invertible mapping h from P^2 to itself such that three points x_1,x_2,x_3 lie on the same line iff $h(x_1),h(x_2),h(x_3)$ do.

- Theorem
  - A mapping h:$P^2 \rightarrow P^2$ is a projectivity iff there exists a non-singular 3x3 matrix **H** such that for any point in P^2 represented by a vector x it is true that $h(x)=\mathbf{H}x$

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad \text{or} \quad \mathbf{x}' = \mathbf{H}\,\mathbf{x}$$

8DOF

terms: projectivity=collineation=projective transformation=homography

Homographies of Points and Lines

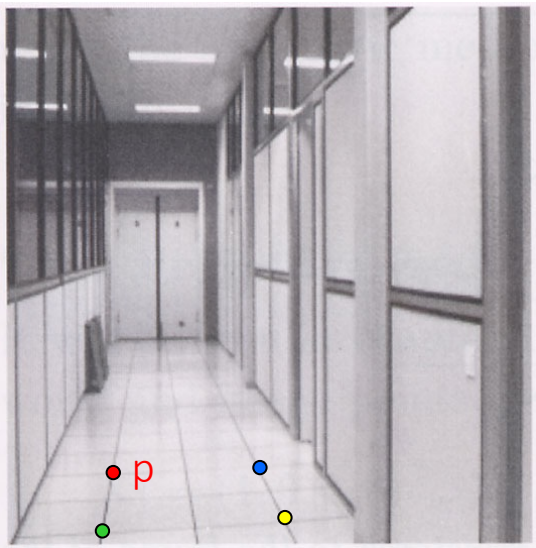
- Computed by 3x3 matrix multiplication
  - To transform a point: $\mathbf{p'} = \mathbf{Hp}$
  - To transform a line: $\mathbf{l}^T\mathbf{p}=0 \rightarrow \mathbf{l'}^T\mathbf{p'}=0$
    - $0 = l^Tp = l^TH^{-1}Hp = l^TH^{-1}p' \Rightarrow (l')^T = l^TH^{-1}$ or $l' = H^{-T}l$

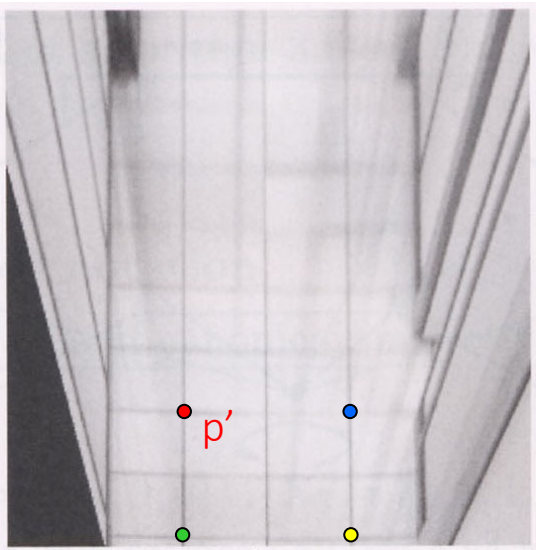
*A homography is an invertible transformation from a projective plane to a projective plane that maps straight lines to straight lines.

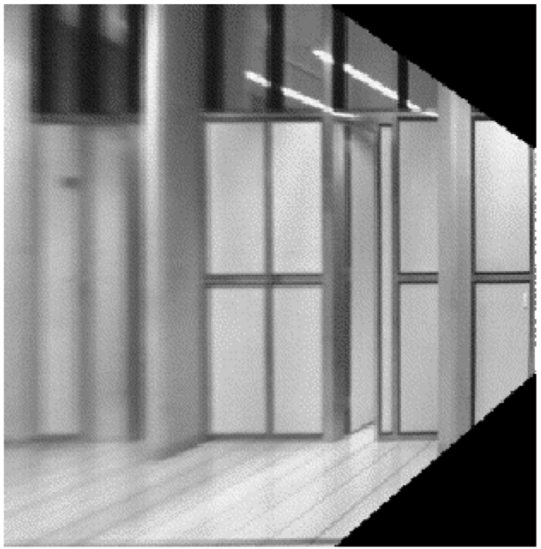
3D Projective Geometry

- These concepts generalize naturally to 3D
  - Homogeneous coordinates
    - Projective 3D points have four coords: $\mathbf{P} = (X,Y,Z,W)$
  - Duality
    - A plane $\mathbf{N}$ is also represented by a 4-vector
    - Points and planes are dual in 3D: $\mathbf{N}\,\mathbf{P}=0$
  - Projective transformations
    - Represented by 4x4 matrices $\mathbf{T}$: $\mathbf{P}' = \mathbf{T}\mathbf{P}, \quad \mathbf{N}' = \mathbf{N}\,\mathbf{T}^{-1}$

Image Rectification







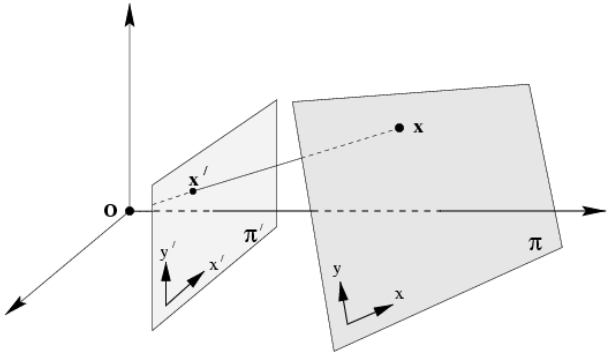
To unwarp (rectify) an image

- solve for homography H given p and p’
- solve equations of the form: wp’ = Hp
  - linear in unknowns: w and coefficients of H
  - H is defined up to an arbitrary scale factor
  - how many points are necessary to solve for H?

Homography Computation

For one pair of correspondences

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



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

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

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

KAIST

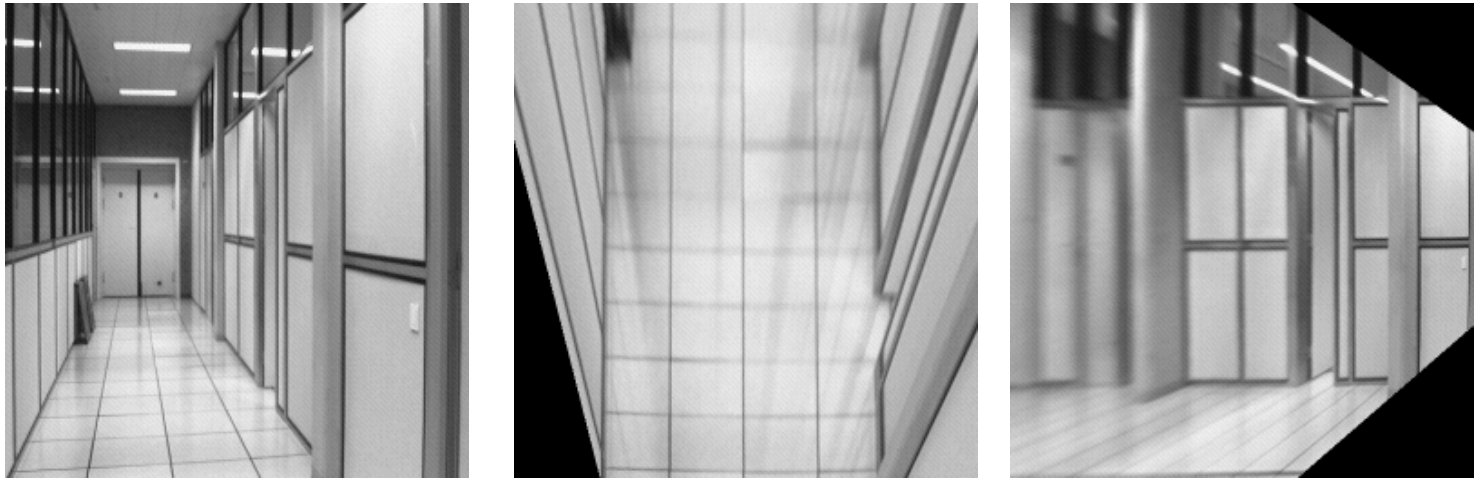
Homography Computation

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{A} \qquad \underset{9}{h} \qquad \underset{2n}{0}$$

- Since h is only defined up to scale, solve for unit vector $\hat{h}$
  - $\hat{h}$ = eigenvector of A^TA with smallest eigenvalue
  - Need 4 or more points since H has 8 dof and we have 2 linearly independent equations for one pair of points.

Using Homography for Synthetic Rotations



The synthetic images are produced by projectively warping the original image so that four corners of an imaged rectangle map to the corners of a rectangle. Both warpings correspond to a synthetic rotation of the camera about the (fixed) camera centre.

Hierarchy of Transformations

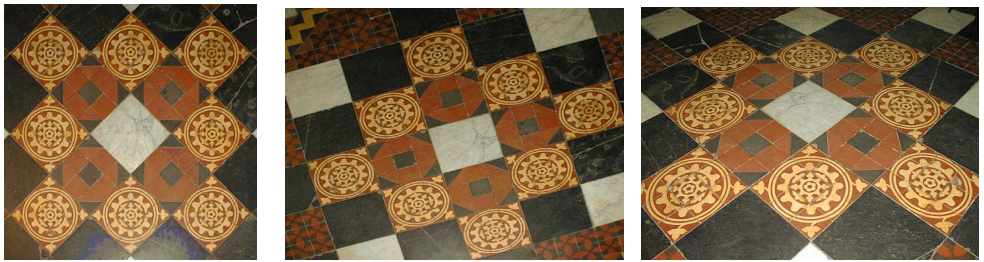
*Projective linear group

*Affine group (last row (0,0,1))

*Euclidean group (upper left 2x2 orthogonal)

*Oriented Euclidean group (upper left 2x2 det 1)

- Alternative, characterize transformation in terms of elements or quantities that are preserved or invariant
  - e.g. Euclidean transformations leave distances unchanged



Class I: Isometries

(iso=same, metric=measure)

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \varepsilon\cos\theta & -\sin\theta & t_x \\ \varepsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \varepsilon = \pm 1$$

orientation preserving: $\varepsilon = 1$

orientation reversing: $\varepsilon = -1$

$$\mathrm{x}' = \mathbf{H}_E \, \mathrm{x} = \begin{bmatrix} \mathbf{R} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \mathrm{x} \qquad \mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{I}$$

3DOF (1 rotation, 2 translation)

special cases: pure rotation, pure translation

Invariants: length, angle, area

Class II: Similarities

(isometry + scale)

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\mathrm{x}' = \mathbf{H}_S \, \mathrm{x} = \begin{bmatrix} s\mathbf{R} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \mathrm{x} \qquad \mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{I}$$

4DOF (1 scale, 1 rotation, 2 translation)

also know as equi-form (shape preserving)

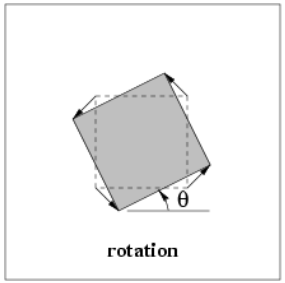
metric structure = structure up to similarity (in literature)

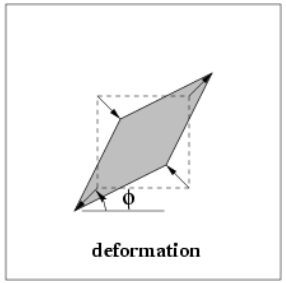
Invariants: ratios of length, angle, ratios of areas, parallel lines

Class III: Affine Transformations

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\mathbf{x}' = \mathbf{H}_A \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \mathbf{x}$$





$$\mathbf{A} = \mathbf{R}(\theta)\mathbf{R}(-\phi)\mathbf{D}\mathbf{R}(\phi) \qquad \mathbf{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

6DOF (2 scale, 2 rotation, 2 translation)

non-isotropic scaling! (2DOF: scale ratio and orientation)

Invariants: parallel lines, ratios of parallel lengths,
ratios of areas

Class VI: Projective Transformations

$$\mathbf{x}' = \mathbf{H}_P \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \mathbf{x} \qquad \mathbf{v} = (v_1, v_2)^T$$

8DOF (2 scale, 2 rotation, 2 translation, 2 line at infinity)

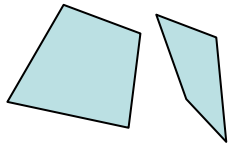
Action non-homogeneous over the plane

Invariants: cross-ratio of four points on a line
(ratio of ratio)

Overview of Transformations

Projective 8dof

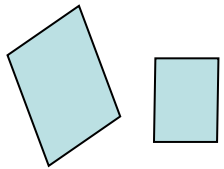
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$



Concurrency, collinearity, order of contact (intersection, tangency, inflection, etc.), cross ratio

Affine 6dof

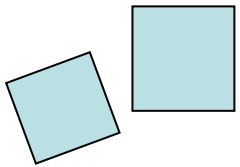
$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Parallellism, ratio of areas, ratio of lengths on parallel lines (e.g midpoints), linear combinations of vectors (centroids).
The line at infinity l_∞

Similarity 4dof

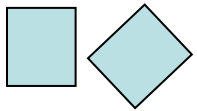
$$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Ratios of lengths, angles.
The circular points I,J

Euclidean 3dof

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



lengths, areas.

3D to 2D: “Perspective” Projection

- Matrix Projection:

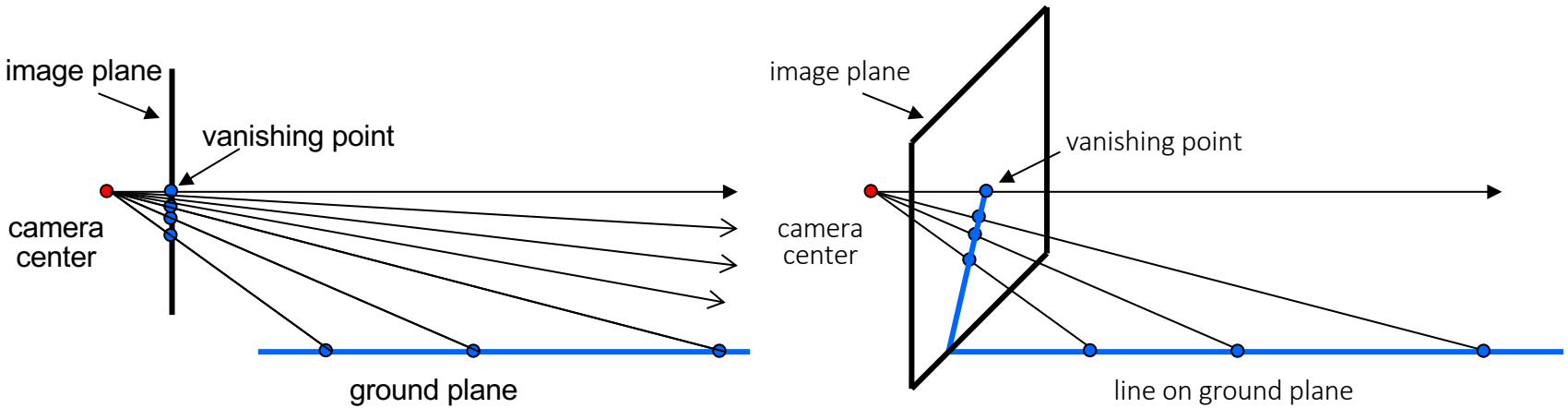
$$\mathbf{p} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$$

What is not preserved under perspective projection?

What IS preserved?

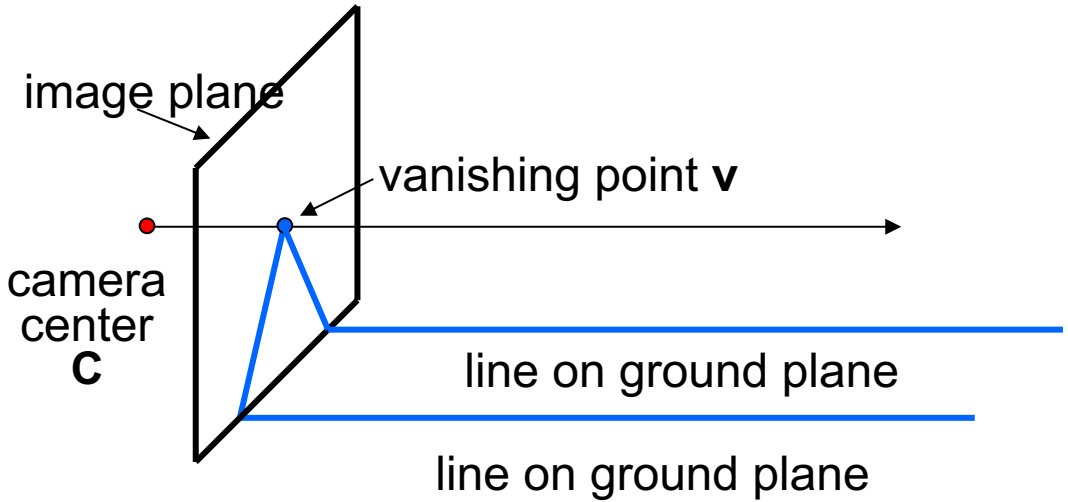
Vanishing Points

- Definition of vanishing point
  - Projection of a point at infinity



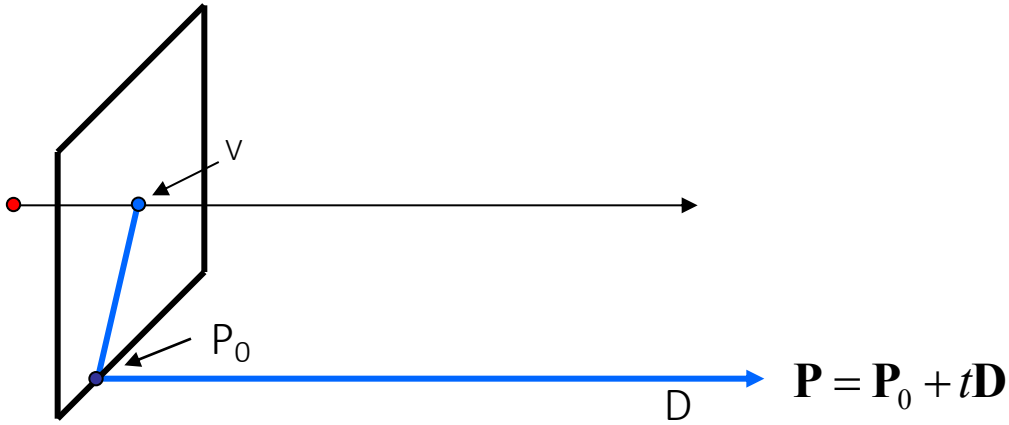
Vanishing Points

- Properties
  - Any two parallel lines have the same vanishing point.
  - The ray from C through v is parallel to the lines.
  - An image may have more than one vanishing point.
    - Every pixel is a potential vanishing point.



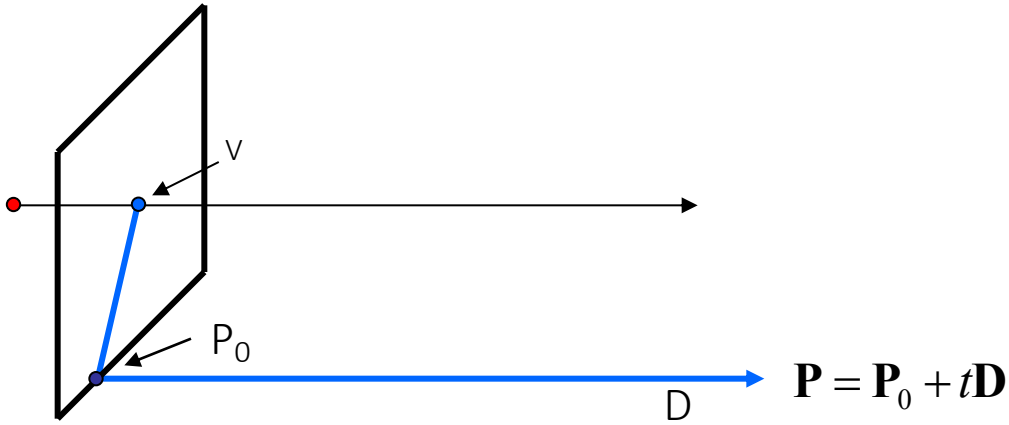
Slide by Seitz

Computing Vanishing Points



$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix}$$

Computing Vanishing Points

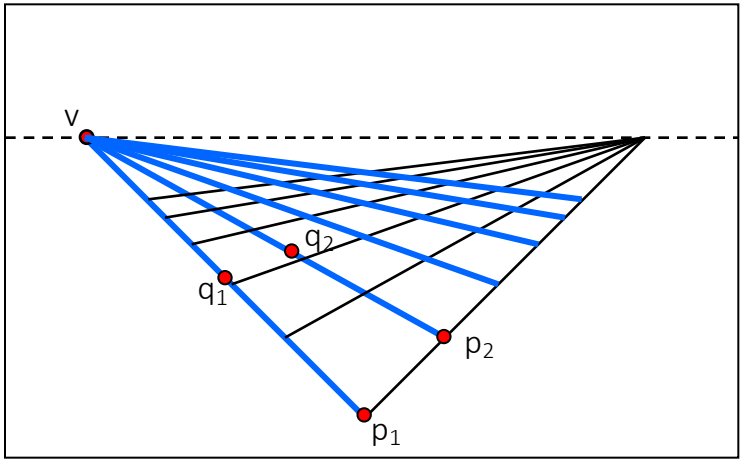


$$\mathbf{P} = \mathbf{P}_0 + t\mathbf{D}$$

$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X/t + D_X \\ P_Y/t + D_Y \\ P_Z/t + D_Z \\ 1/t \end{bmatrix} \qquad t \to \infty \qquad \mathbf{P}_\infty \cong \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix}$$

- Properties $\mathbf{v} = \mathbf{\Pi P}_\infty$
  - $\mathbf{P}_\infty$ is a point at *infinity*, $\mathbf{v}$ is its projection
  - They depend only on line *direction*
  - Parallel lines $\mathbf{P}_0 + t\mathbf{D}$, $\mathbf{P}_1 + t\mathbf{D}$ intersect at $\mathbf{P}_\infty$

Computing Vanishing Points (from lines)



- Intersect p_1q_1 with p_2q_2

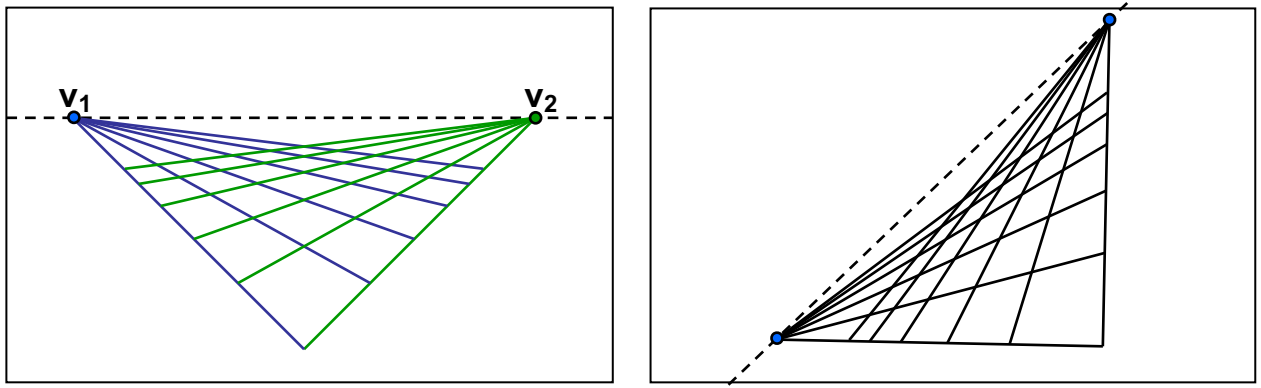
$$v = (p_1 \times q_1) \times (p_2 \times q_2)$$

Least squares version

- Better to use more than two lines and compute the "closest" point of intersection
- See notes by Bob Collins for one good way of doing this:
  - http://www-2.cs.cmu.edu/~ph/869/www/notes/vanishing.txt

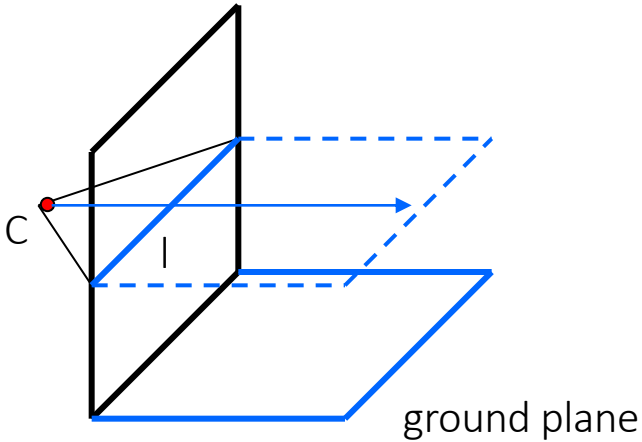
Vanishing Lines

- Multiple vanishing points
  - Any set of parallel lines on the plane define a vanishing point.
  - The union of all of vanishing points from lines on the same plane is the **vanishing line**.
    - For the ground plane, this is called the horizon.
  - Different planes define different vanishing lines.



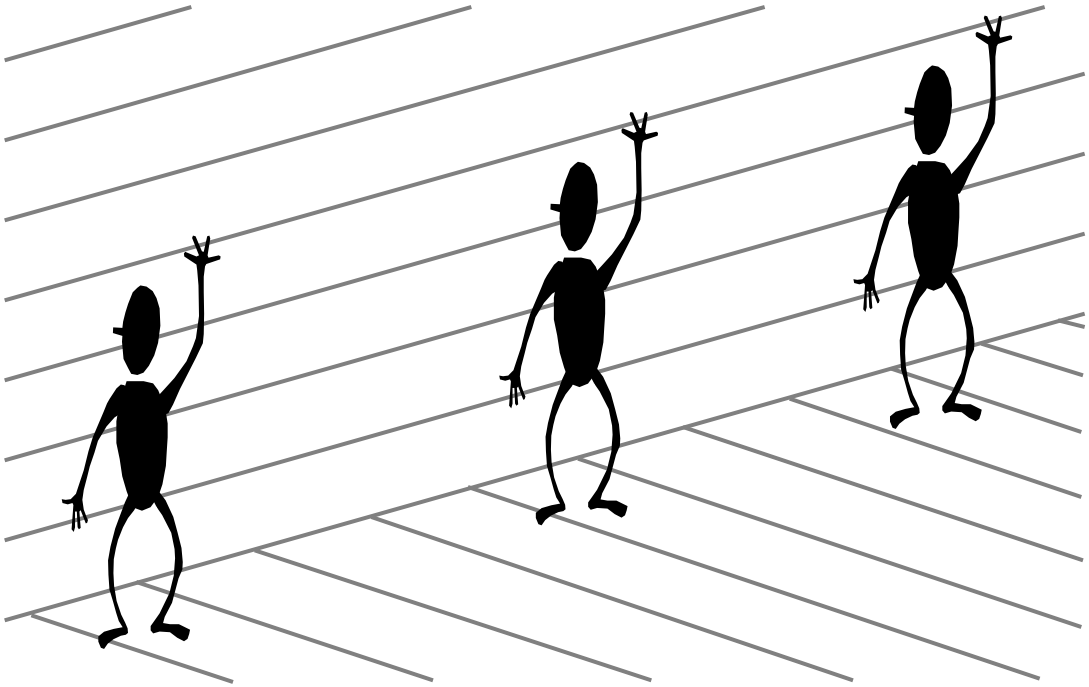
Slide by Seitz

Computing the Horizon

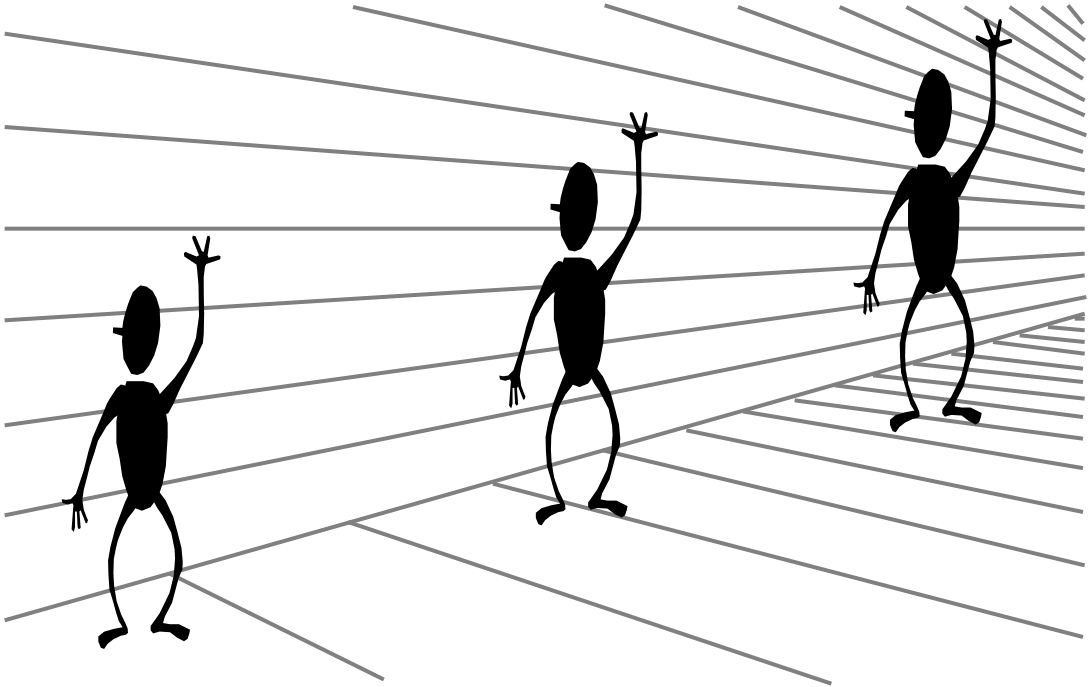


- Properties
  - **l** is intersection of horizontal plane through **C** with image plane
  - Compute **l** from two sets of parallel lines on ground plane
  - All points at same height as **C** project to **l**
    - points higher than **C** project above **l**
  - Provides way of comparing height of objects in the scene

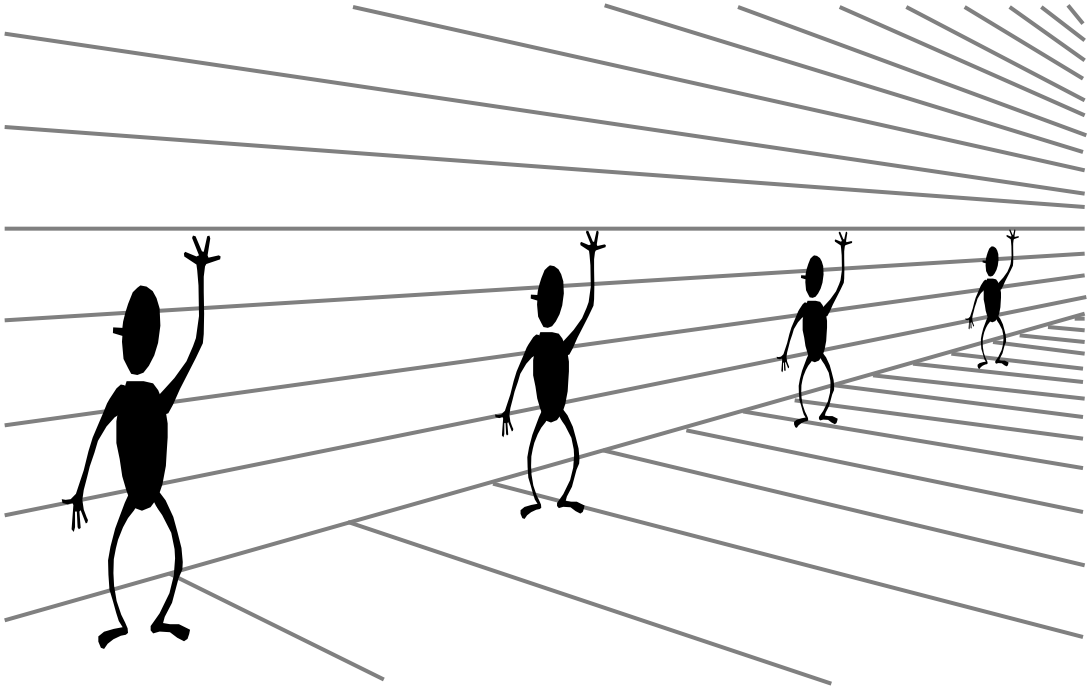
Perspective Cues



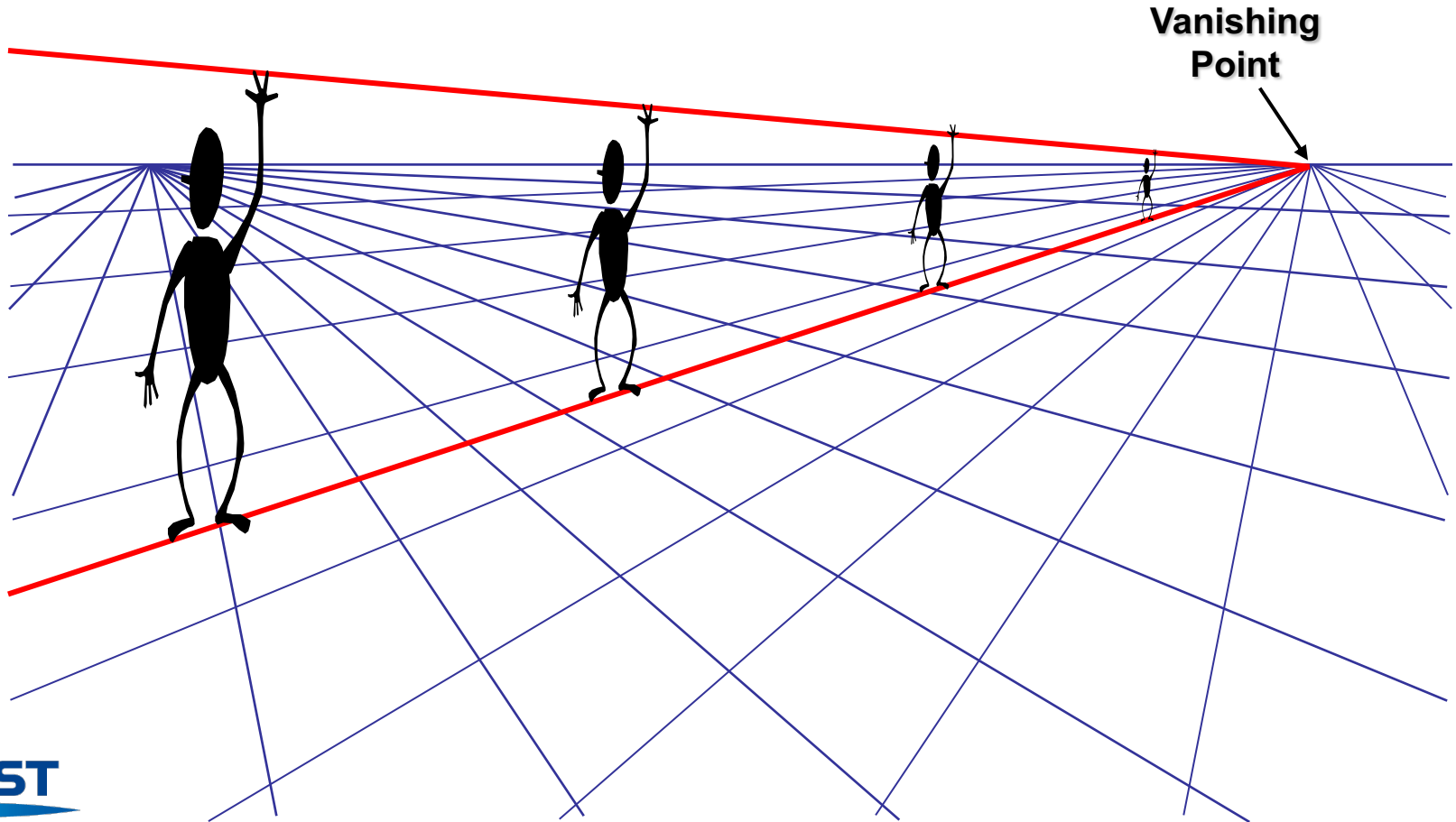
Perspective Cues



Perspective Cues

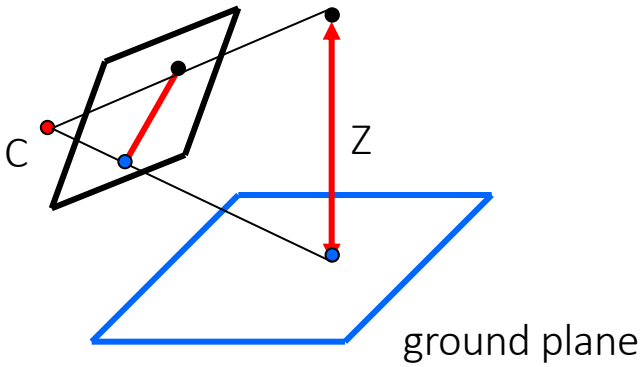


Comparing Heights



KAIST

Measuring Height without a Ruler



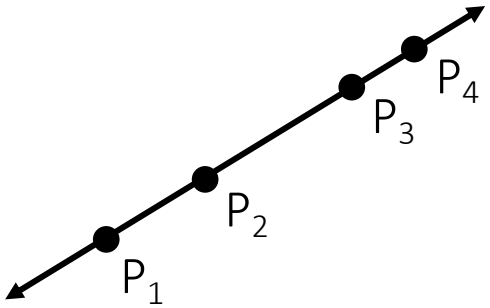
Compute Z from image measurements

- Need more than vanishing points to do this

The Cross Ratio

- A Projective Invariant
  - Something that does not change under projective transformations (including perspective projection)

The cross-ratio of 4 collinear points



$$\frac{\|\mathbf{P}_3 - \mathbf{P}_1\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_3 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_1\|}$$

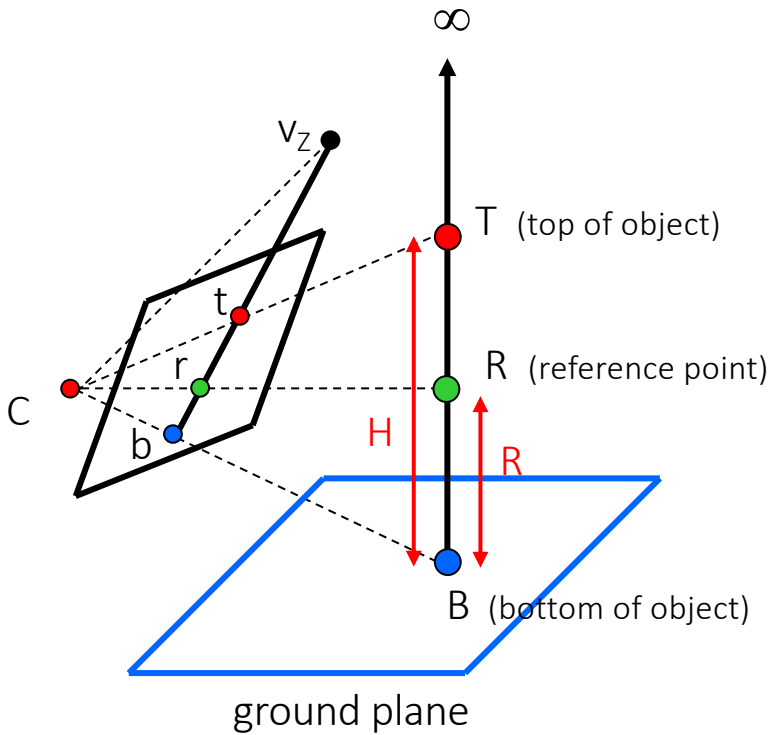
$$\mathbf{P}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Can permute the point ordering $\dfrac{\|\mathbf{P}_1 - \mathbf{P}_3\| \, \|\mathbf{P}_4 - \mathbf{P}_2\|}{\|\mathbf{P}_1 - \mathbf{P}_2\| \, \|\mathbf{P}_4 - \mathbf{P}_3\|}$

- 4! = 24 different orders (but only 6 distinct values)

This is the fundamental invariant of projective geometry

Measuring Height



$$\frac{\|\mathbf{T}-\mathbf{B}\|\,\|\infty-\mathbf{R}\|}{\|\mathbf{R}-\mathbf{B}\|\,\|\infty-\mathbf{T}\|}=\frac{H}{R}$$

scene cross ratio

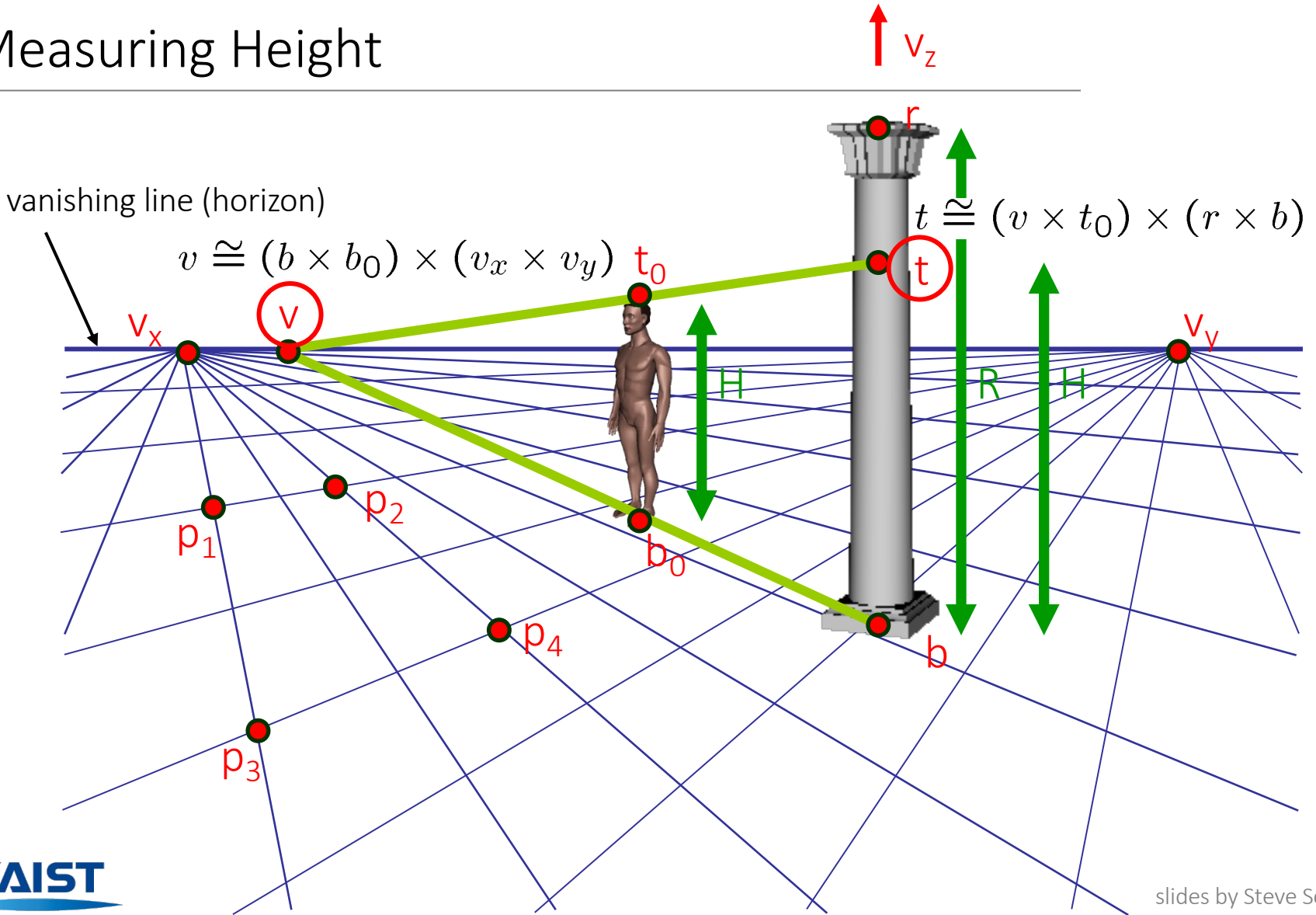
$$\frac{\|\mathbf{t}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{r}\|}{\|\mathbf{r}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{t}\|}=\frac{H}{R}$$

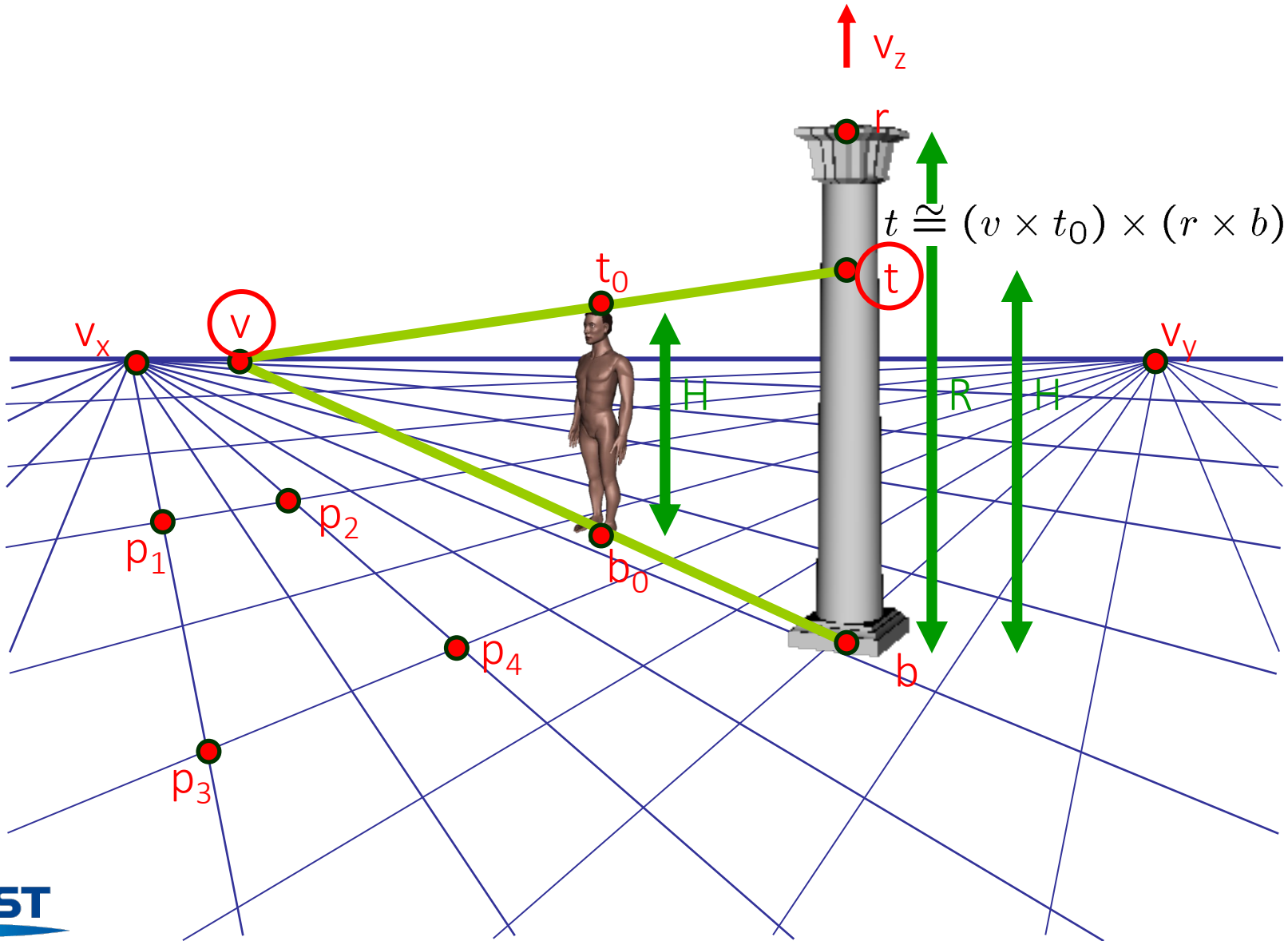
image cross ratio

$$H=R\frac{\|\mathbf{t}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{r}\|}{\|\mathbf{r}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{t}\|}$$

scene points represented as $\mathbf{P}=\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$ image points as $\mathbf{p}=\begin{bmatrix}x\\y\\1\end{bmatrix}$

Measuring Height



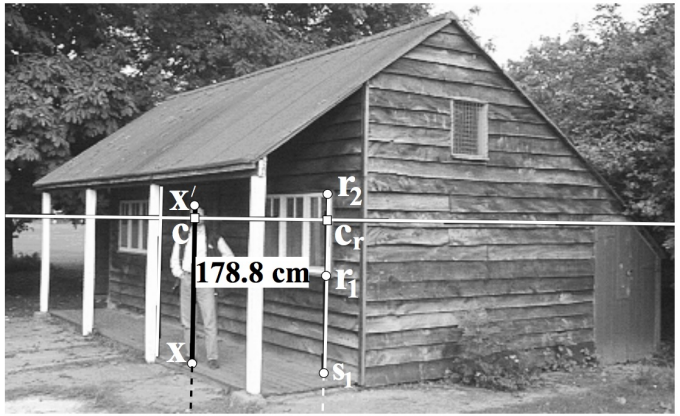

v_z
r
$t \cong (v \times t_0) \times (r \times b)$
t
t_0
v
v_x
v_y
H
R
H
p_1
p_2
p_3
p_4
b_0
b

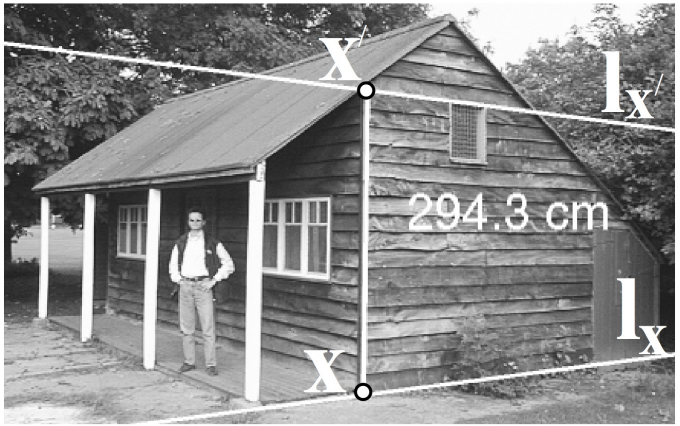
Single View Metrology

- A. Criminisi, I. Reid and A. Zisserman (ICCV 99)
- Make scene measurements from a single image
  - Application: 3D from a single image
- Assumptions
  1. 3 orthogonal sets of parallel lines
  2. 4 known points on ground plane
  3. 1 height in the scene
- Can still get an *affine reconstruction* without 2 and 3

3D Modeling from a Photograph

Single View Metrology

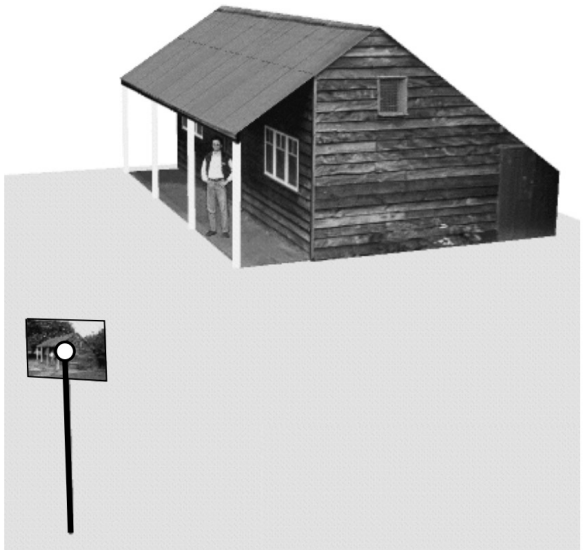




Single View Metrology

a

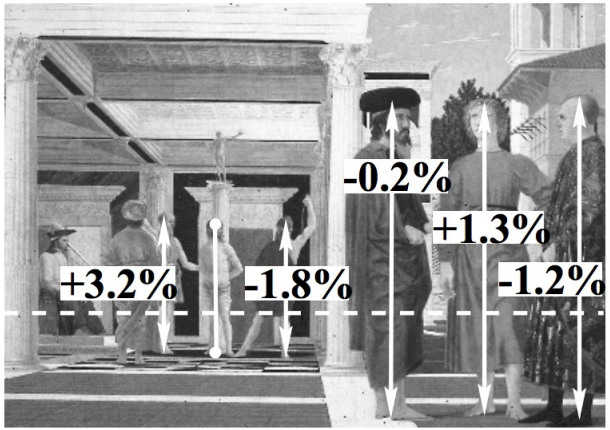
b



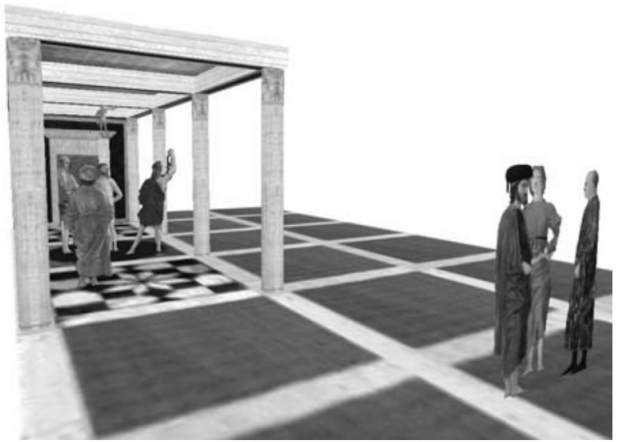
c

Single View Metrology

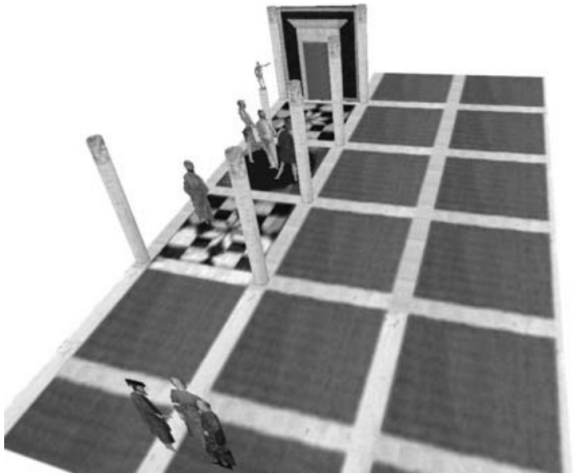
a



b



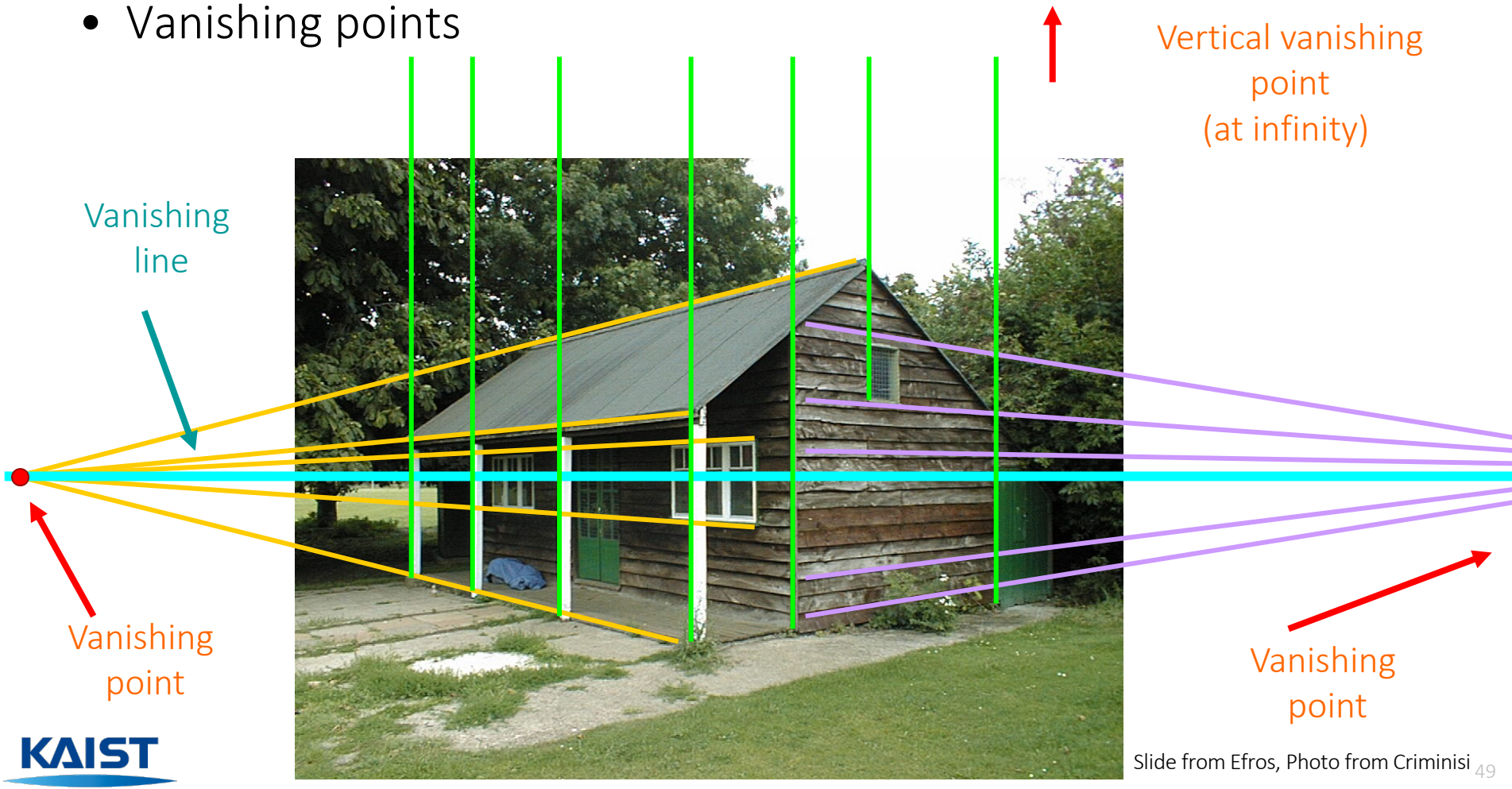
c



d

Single View Metrology

- Vanishing points



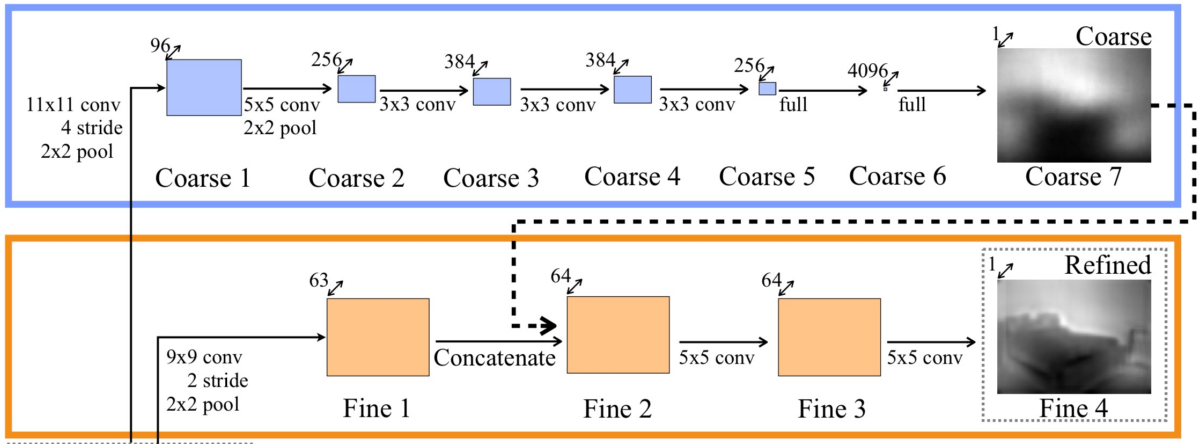
Slide from Efros, Photo from Criminisi

Appendix: Criminisi et al., ICCV 99

- Complete approach
  - Load in an image
  - Click on parallel lines defining X, Y, and Z directions
  - Compute vanishing points
  - Specify points on the reference plane, ref. height
  - Compute 3D positions of several points
  - Create a 3D model from these points
  - Extract texture maps
  - Output a VRML model

Single View Metrology – Deep-learning based (1)

- Depth Map Prediction from a Single Image using a Multi-Scale Deep Network (Eigen et. al, NIPS 2014)

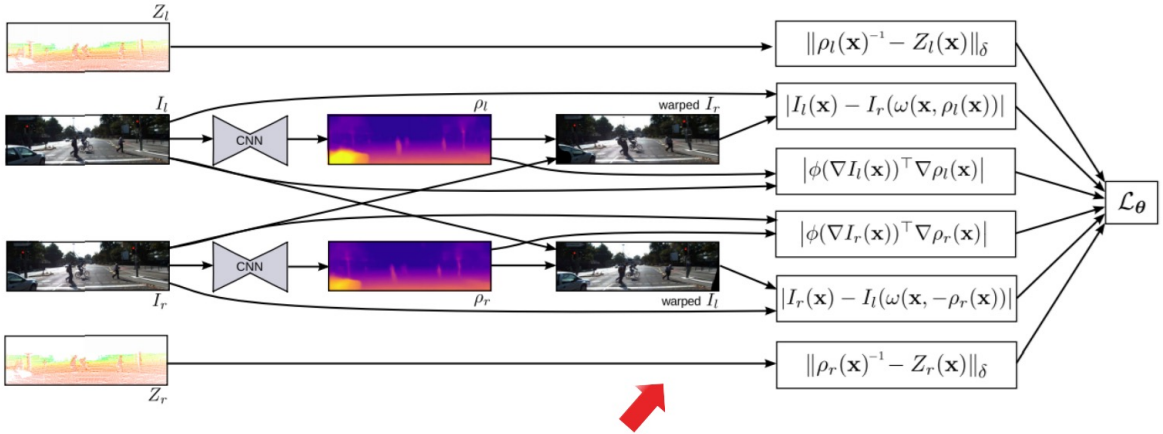


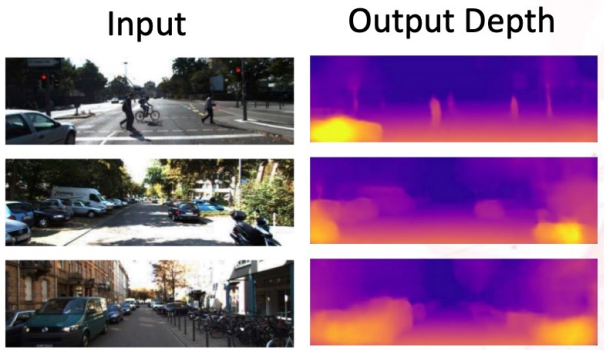
		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Single View Metrology – Deep-learning based (2)

Semi-supervised/Unsupervised schemes

- Use stereo pairs for training
- Based on left-right consistency and smoothness of depth





[8] defines a series of loss functions based on left-right consistency and depth smoothness

[7] C. Godard, O. Mac Aodha, G.J. Brostow"Unsupervised monocular depth estimation with left-right consistency," In *Proc. CVPR*, 2017.

[8] Y Kuznietsov, J Stückler and L. Bastian, "Semi-supervised deep learning for monocular depth map prediction," In *Proc. CVPR*, 2017.

KAIST

Single View Metrology – Deep-learning based (3)

- Digging Into Self-Supervised Monocular Depth Estimation, ICCV 2019

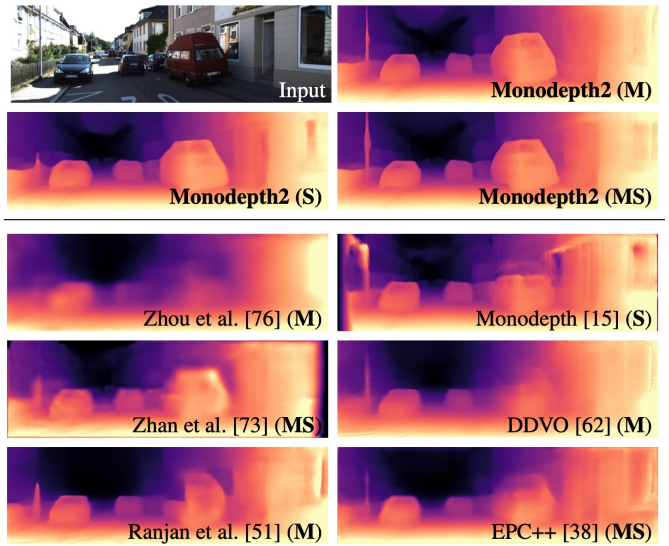


Figure 1. **Depth from a single image.** Our self-supervised model, **Monodepth2**, produces sharp, high quality depth maps, whether trained with monocular (M), stereo (S), or joint (MS) supervision.

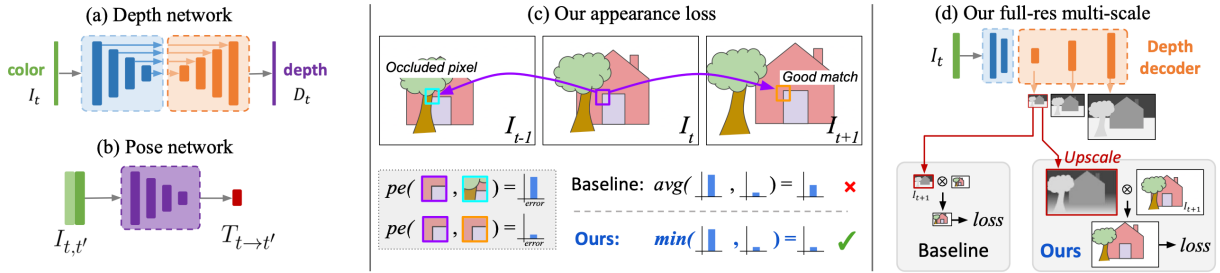


Figure 3. **Overview. (a) Depth network:** We use a standard, fully convolutional, U-Net to predict depth. **(b) Pose network:** Pose between a pair of frames is predicted with a separate pose network. **(c) Per-pixel minimum reprojection:** When correspondences are *good*, the reprojection loss should be *low*. However, occlusions and disocclusions result in pixels from the current time step not appearing in both the previous and next frames. The baseline *average* loss forces the network to match occluded pixels, whereas our *minimum reprojection* loss only matches each pixel to the view in which it is visible, leading to sharper results. **(d) Full-resolution multi-scale:** We upsample depth predictions at intermediate layers and compute all losses at the input resolution, reducing texture-copy artifacts.

- Combining the depth network with the pose network to handle occlusions and moving objects

Single View Metrology – Deep-learning based (4)

- Enforcing geometric constraints of virtual normal for depth prediction, ICCV 2019
  - enforcing a high- order geometric constraint in the 3D space for the depth prediction task.

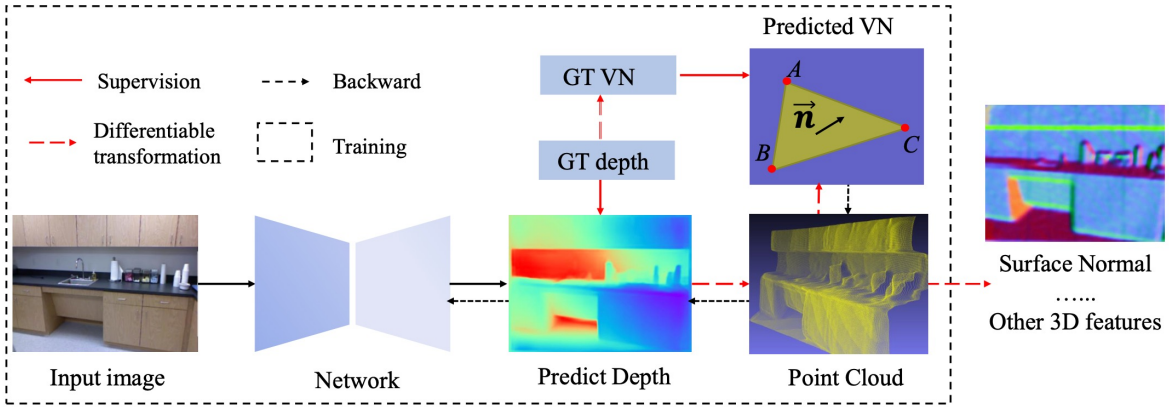


Figure 2. Illustration of the pipeline of our method. An encoder-decoder network is employed to predict the depth, from which the point cloud can be reconstructed. A pixel-wise depth supervision is firstly enforced on the predicted depth, while a geometric supervision, virtual normal constraint, is enforced in 3D space. With the well trained model, other 3D features, such as the surface normal, can be directly recovered from the reconstructed 3D point cloud in the inference.

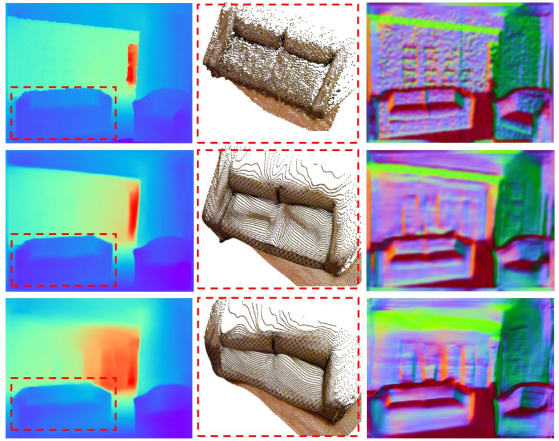
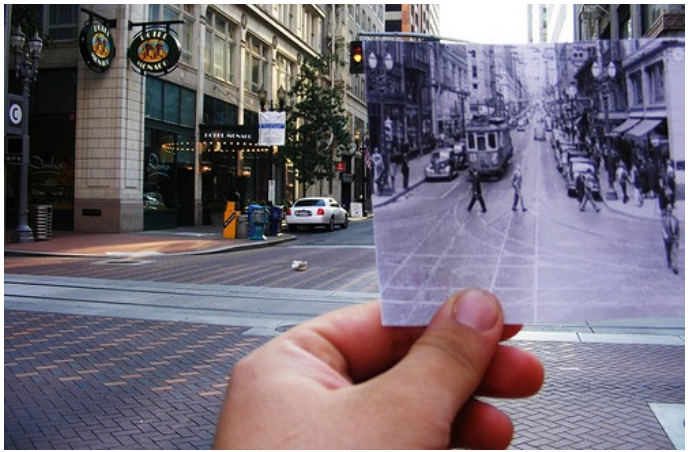


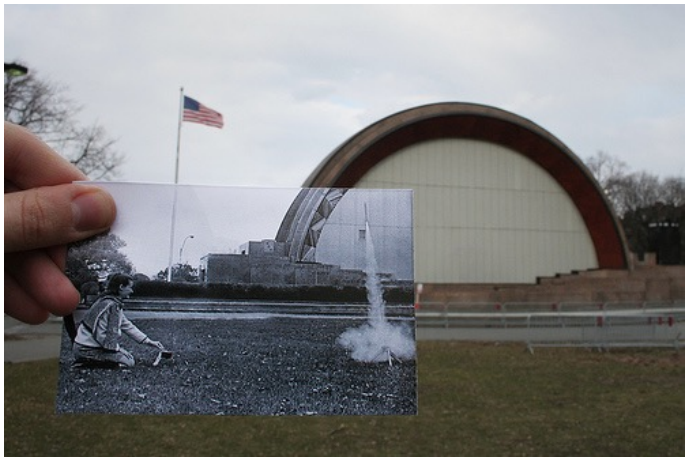
Figure 1. Example results of ground truth (the first row), our method (the second row) and Hu *et al.* [18] (the third row). By enforcing the geometric constraints of virtual normals, our reconstructed 3D point cloud can represent better shape of sofa (see the left part) and the recovered surface normal has much less errors (see green parts) even though the absolute relative error (rel) of our predicted depth is only slightly better than Hu *et al.* (0.108 *vs.* 0.115).

IMAGE STITCHING

KAIST

A Look into the Past

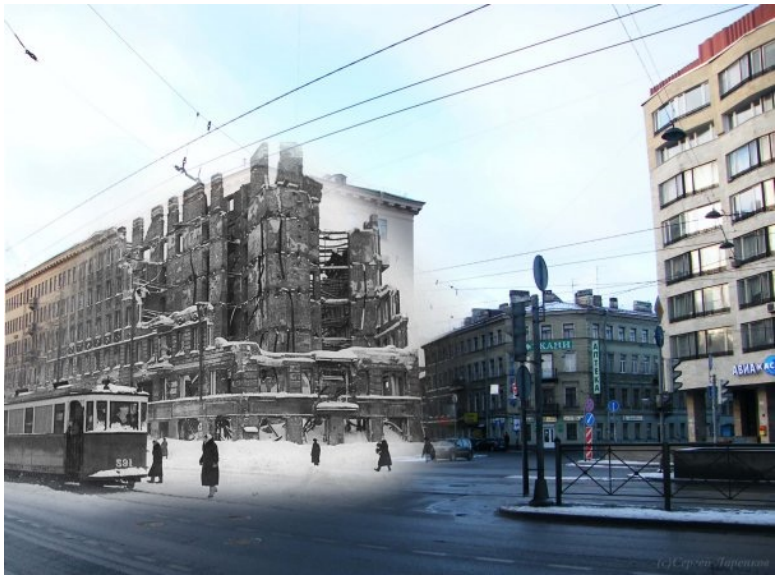




http://blog.flickr.net/en/2010/01/27/a-look-into-the-past/

A Look into the Past

- Leningrad during the blockade



http://komen-dant.livejournal.com/345684.html

Bing Streetside Images

http://www.bing.com/community/blogs/maps/archive/2010/01/12/new-bing-maps-application-streetside-photos.aspx

KAIST

Mosaics

- Getting the whole picture
  - Consumer camera: 50° x 35°

Slide from Brown & Lowe 2003

Mosaics

- Getting the whole picture
  - Consumer camera: 50° x 35°
  - Human Vision: 176° x 135°

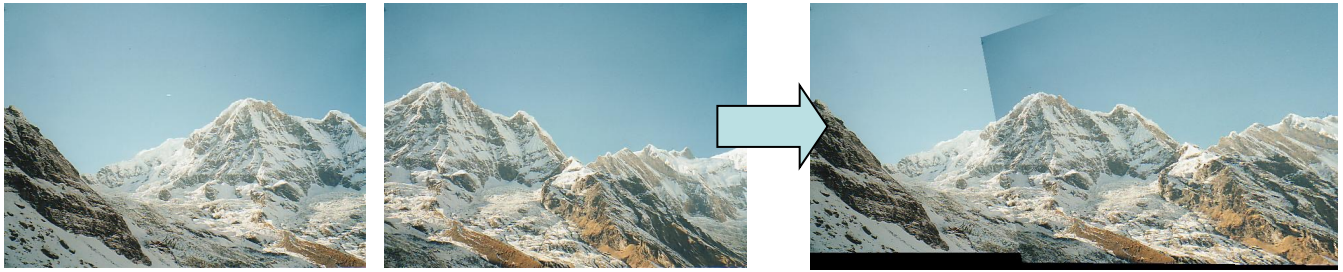
Slide from Brown & Lowe 2003

Mosaics

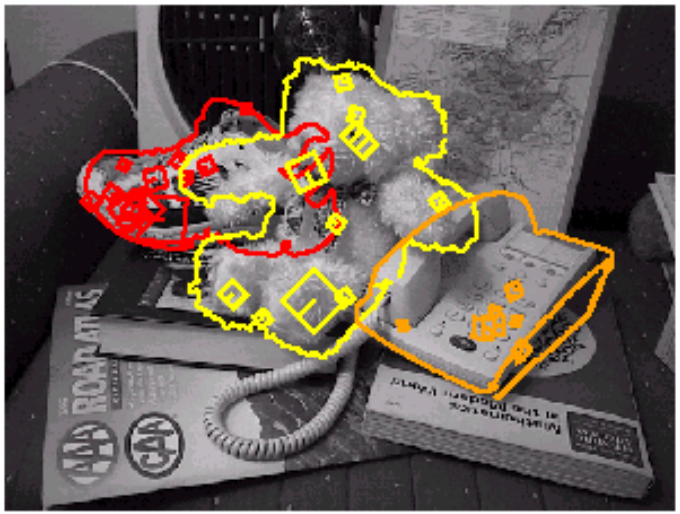
- Getting the whole picture
  - Consumer camera: 50° x 35°
  - Human Vision: 176° x 135°

Slide from Brown & Lowe 2003

Image Alignment: Applications

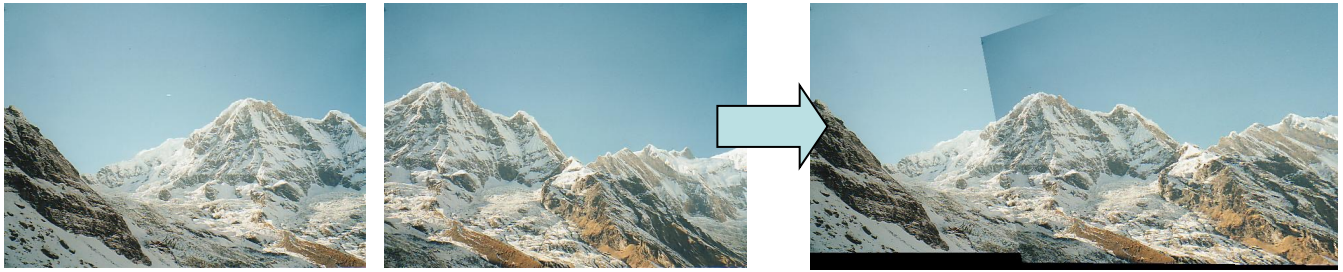


Panorama stitching



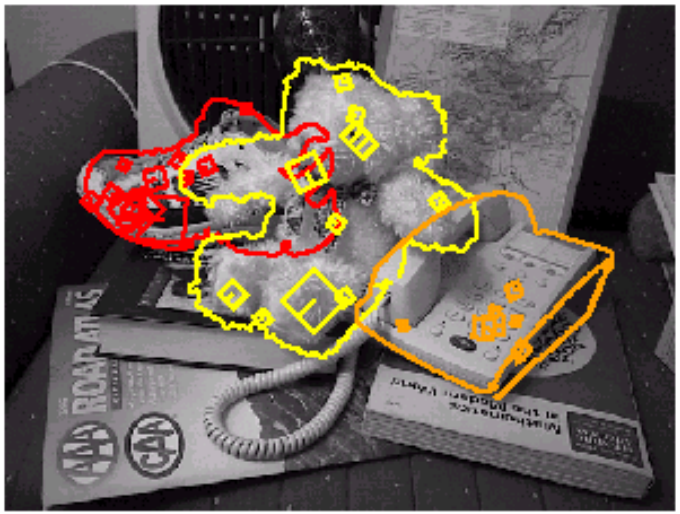
Recognition of object instances

Image Alignment: Challenges



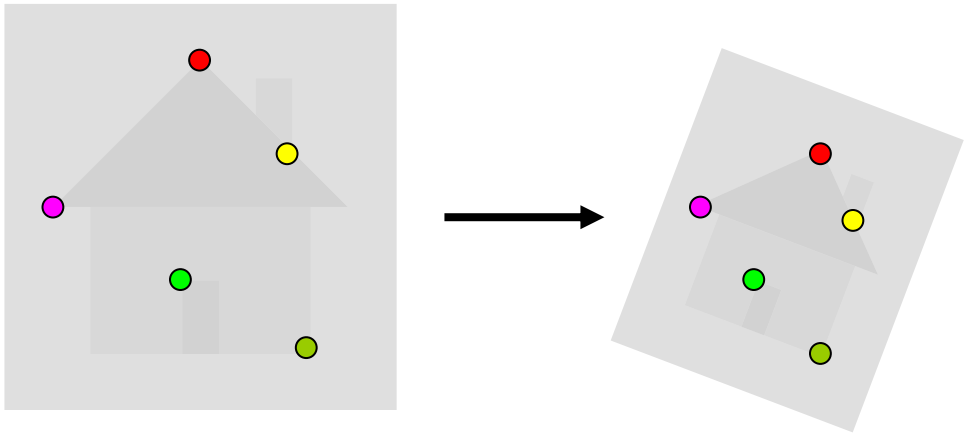
Small degree of overlap

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

KAIST

Motion Models

- What happens when we take two images with a camera and try to align them?
  - translation?
  - rotation?
  - scale?
  - affine?
  - perspective?

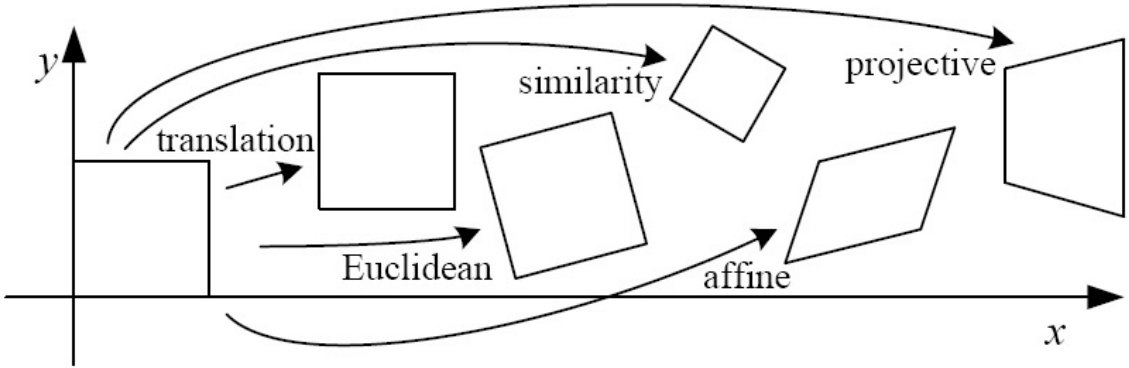
Slide from R. Szeliski

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: $\underline{\boldsymbol{x}}' \cong \boldsymbol{H}\,\underline{\boldsymbol{x}}$

  ($\underline{\boldsymbol{x}} = [x,y,1]^{\mathsf{T}}$ is a *homogeneous* coordinate)

Remind: 2D Image Transformations

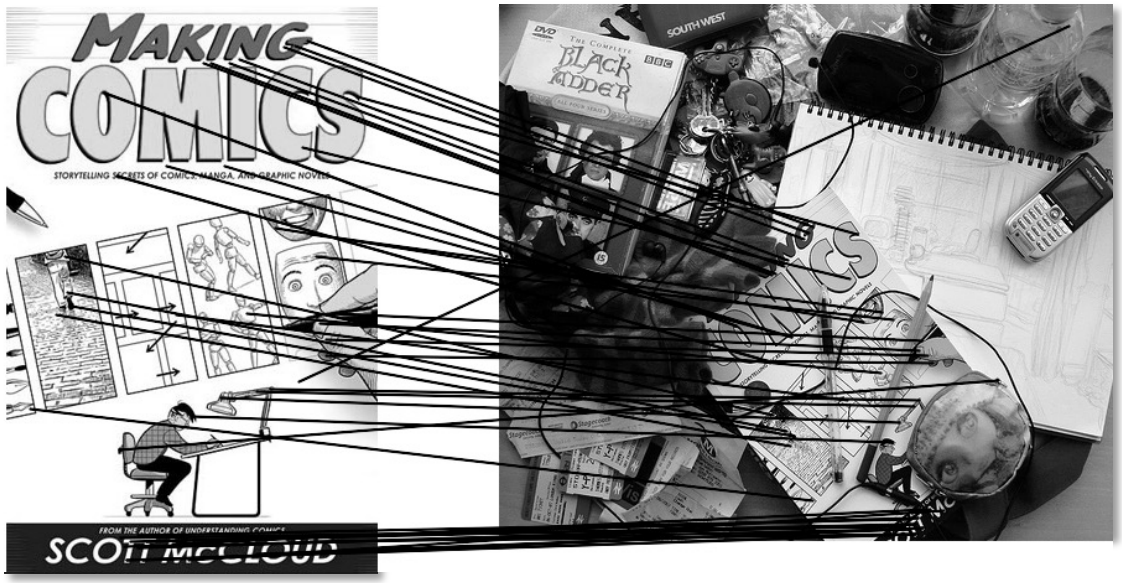


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	4	angles $+\cdots$	
affine	$\left[\boldsymbol{A}\right]_{2\times3}$	6	parallelism $+\cdots$	
projective	$\left[\tilde{\boldsymbol{H}}\right]_{3\times3}$	8	straight lines	

KAIST

Computing Transformations

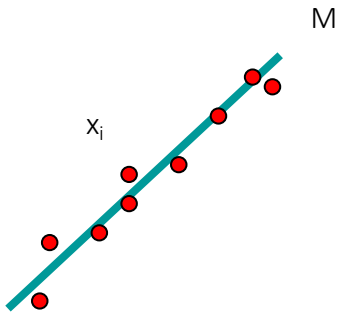
- Given a set of matches between images A and B
  - How can we compute the transform T from A to B?



  - Find transform T that best “agrees” with the matches

Alignment as Fitting

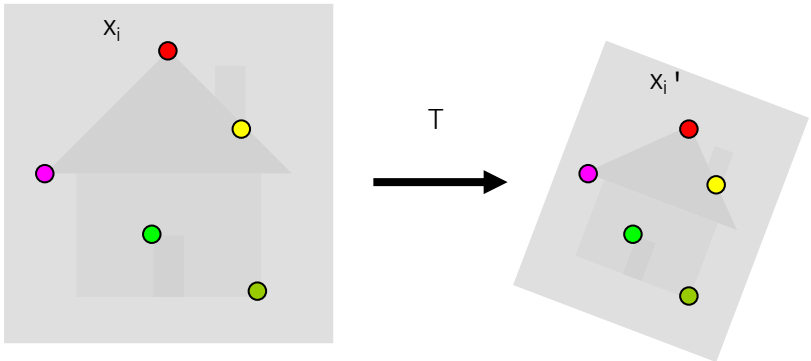
- Fitting a model to features in one image



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

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

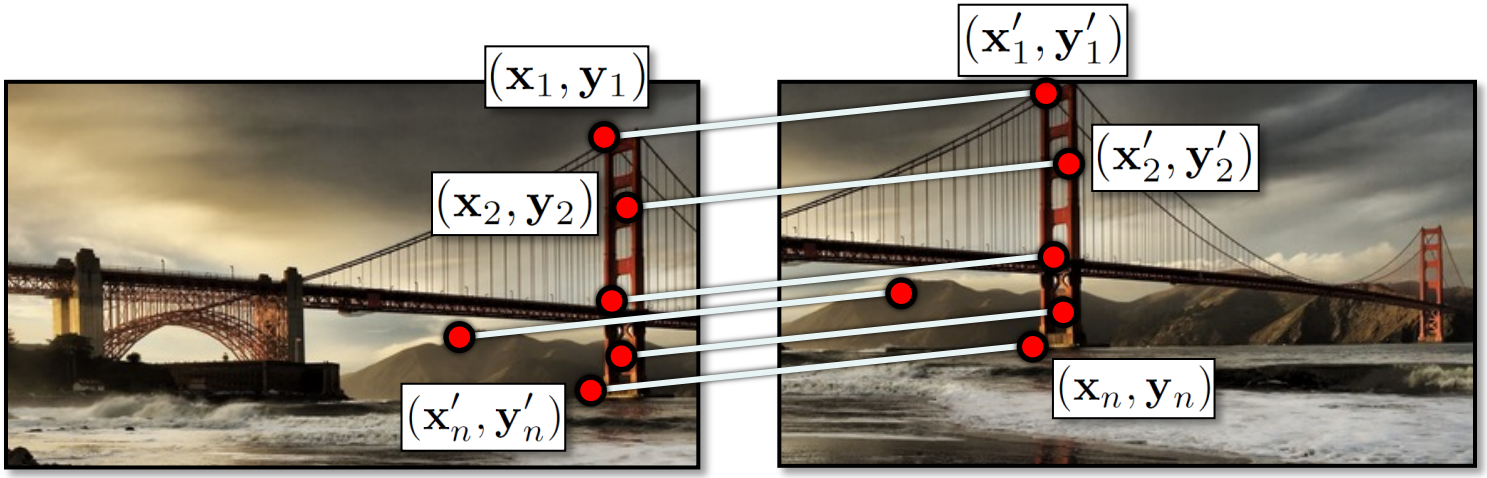
Simple Case: Translations

$(\mathbf{x}_t, \mathbf{y}_t)$

How do we solve for $(\mathbf{x}_t, \mathbf{y}_t)$?

KAIST

Simple Case: Translations

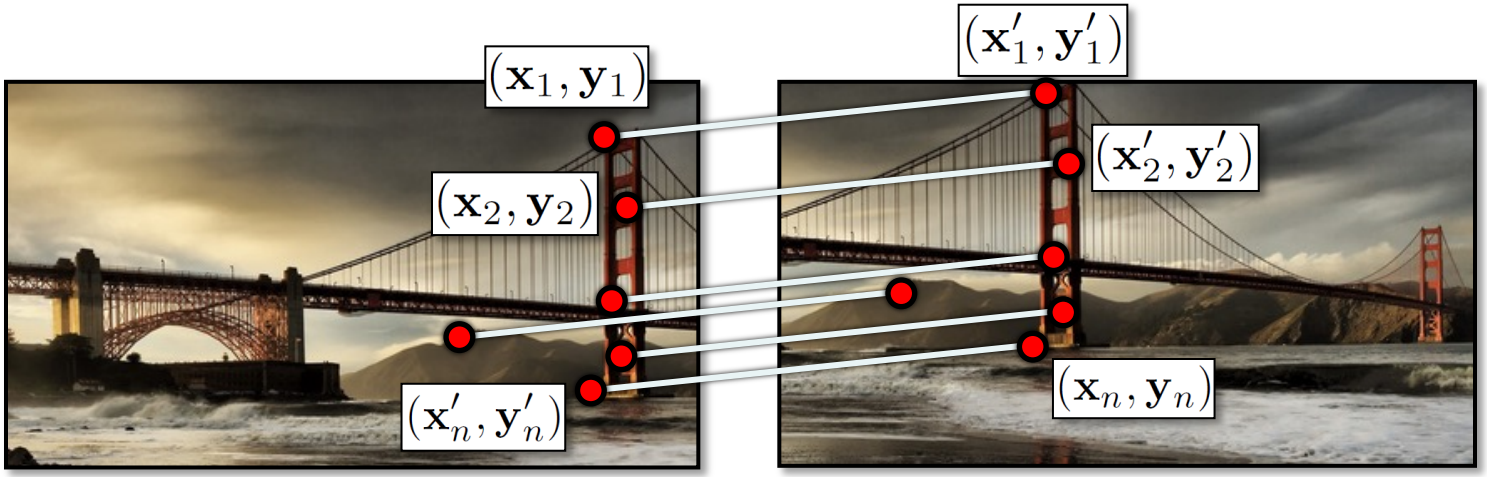


Displacement of match i = $(\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i)$

$$(\mathbf{x}_t, \mathbf{y}_t) = \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}'_i - \mathbf{y}_i \right)$$

KAIST

Another View



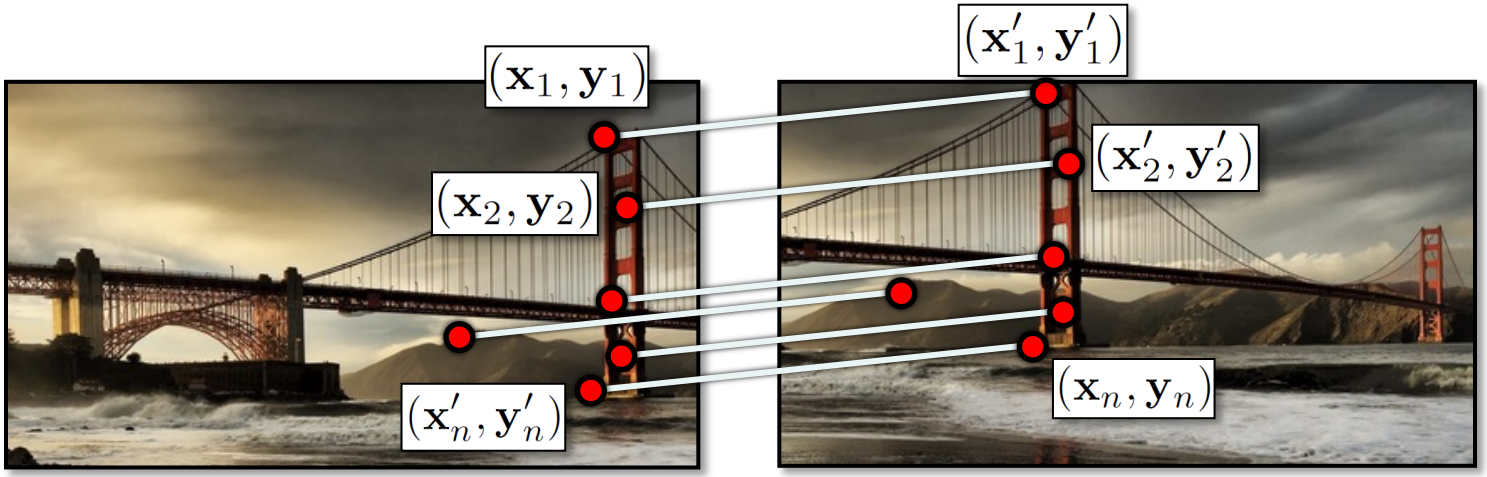
$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?

KAIST

Another View



$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- Problem: more equations than unknowns
  - "Overdetermined" system of equations
  - We will find the least squares solution

Least Squares Formulation

- For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$\begin{aligned} \mathbf{x}_i + \mathbf{x_t} &= \mathbf{x}'_i \\ \mathbf{y}_i + \mathbf{y_t} &= \mathbf{y}'_i \end{aligned}$$

  - We define the *residuals* as

$$\begin{aligned} r_{\mathbf{x}_i}(\mathbf{x}_t) &= (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}'_i \\ r_{\mathbf{y}_i}(\mathbf{y}_t) &= (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}'_i \end{aligned}$$

- Goal: minimize sum of squared residuals

$$C(\mathbf{x}_t, \mathbf{y}_t) = \sum_{i=1}^{n} \left(r_{\mathbf{x}_i}(\mathbf{x}_t)^2 + r_{\mathbf{y}_i}(\mathbf{y}_t)^2 \right)$$

  - "Least squares" solution
  - For translations, it is equal to mean (average) displacement

KAIST

Least Squares Formulation

- Can also write as a matrix equation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\underset{2n \times 2}{\mathbf{A}} \quad \underset{2 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

$$\mathbf{At} = \mathbf{b}$$

- Find t that minimizes

$$||\mathbf{At} - \mathbf{b}||^2$$

- To solve, form the normal equations

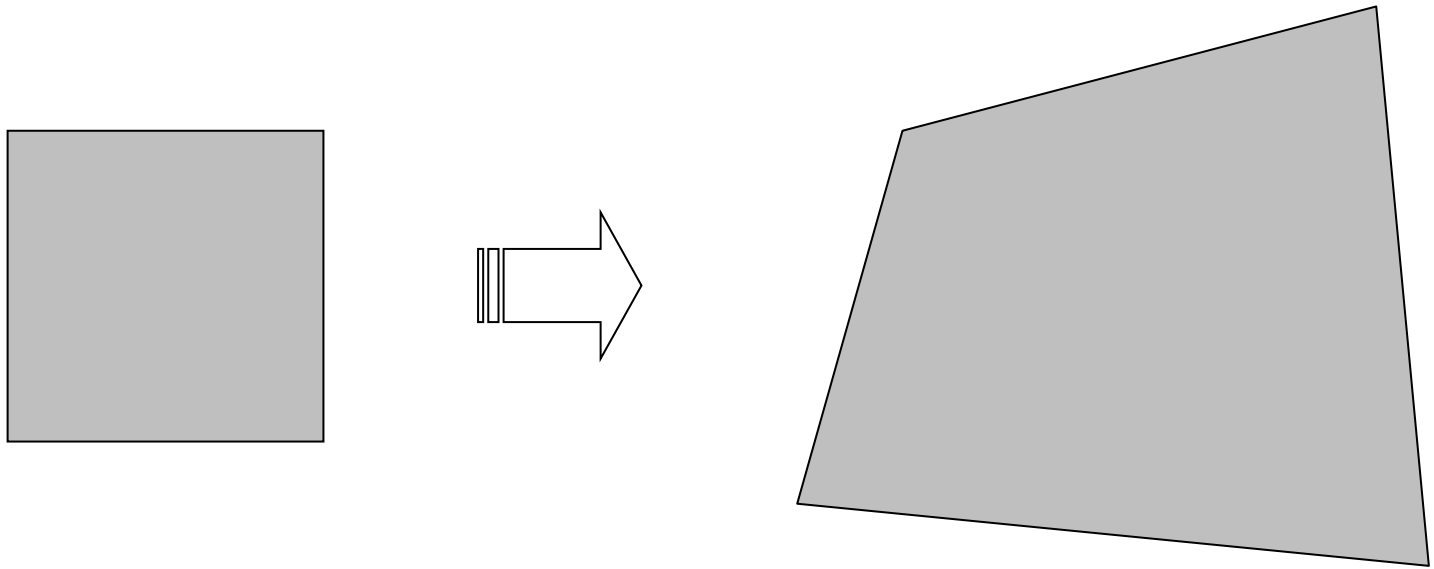
$$\mathbf{A}^{\mathrm{T}}\mathbf{At} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

KAIST

Fitting a Plane Projective Transformation

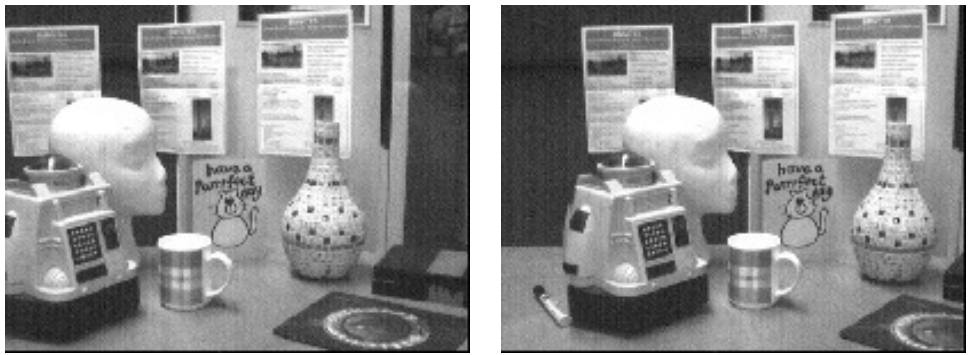
- **Homography:** plane projective transformation
  - transformation taking a quad to another arbitrary quad



Homography

- The transformation between two views of a planar surface

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



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 homogeneous:

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

KAIST

Fitting a Homography

- Equation for homography:

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

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

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

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h_1}^T \mathbf{x}_i \\ \mathbf{h_2}^T \mathbf{x}_i \\ \mathbf{h_3}^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y_i' \mathbf{h_3}^T \mathbf{x}_i - \mathbf{h_2}^T \mathbf{x}_i \\ \mathbf{h_1}^T \mathbf{x}_i - x_i' \mathbf{h_3}^T \mathbf{x}_i \\ x_i' \mathbf{h_2}^T \mathbf{x}_i - y_i' \mathbf{h_1}^T \mathbf{x}_i \end{bmatrix}$$

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

3 equations, only 2 linearly independent

KAIST

Direct Linear Transform

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

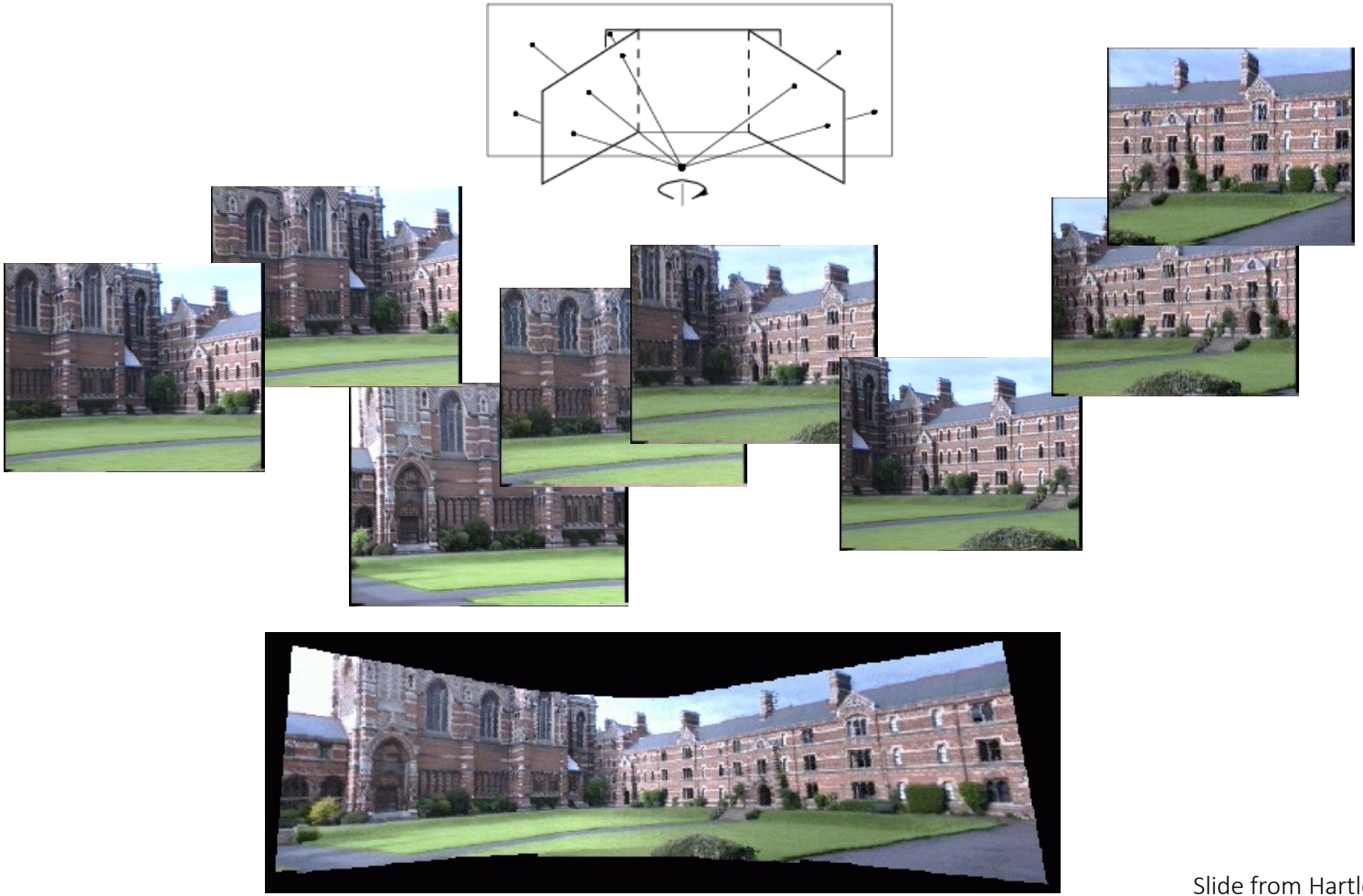
- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Four matches needed for a minimal solution (null space of 8x9 matrix)
- More than four: homogeneous least squares

Recap: Two Common Optimization Problems

Problem statement	Solution
$\text{minimize } \|\mathbf{Ax}-\mathbf{b}\|^2$	$\mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$
least squares solution to $\mathbf{Ax}=\mathbf{b}$	$\mathbf{x}=\mathbf{A}\setminus\mathbf{b}$ (matlab)

Problem statement	Solution
$\text{minimize } \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} \text{ s.t. } \mathbf{x}^T\mathbf{x}=1$	$[\mathbf{v},\lambda]=\text{eig}(\mathbf{A}^T\mathbf{A})$
	$\lambda_1<\lambda_{2..n}:\mathbf{x}=\mathbf{v}_1$
non - trivial lsq solution to $\mathbf{Ax}=0$	

Application: Panorama Stitching



Slide from Hartley & Zisserman

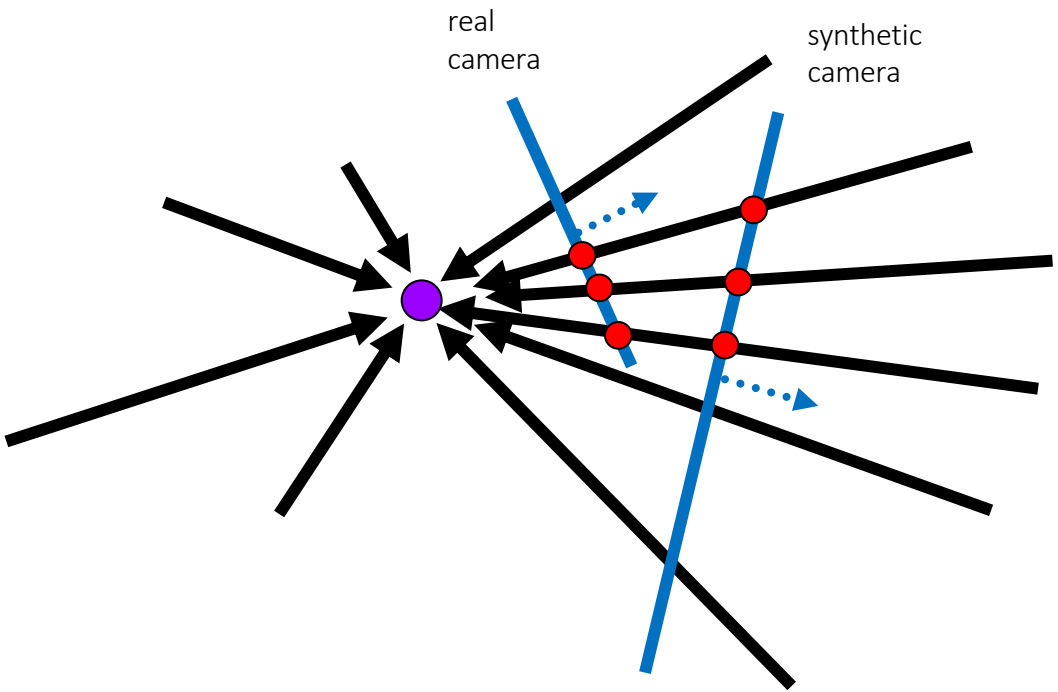
How to Stitch Together a Panorama?

- 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?

Slide from Steve Seitz

Panoramas: Generating Synthetic Views

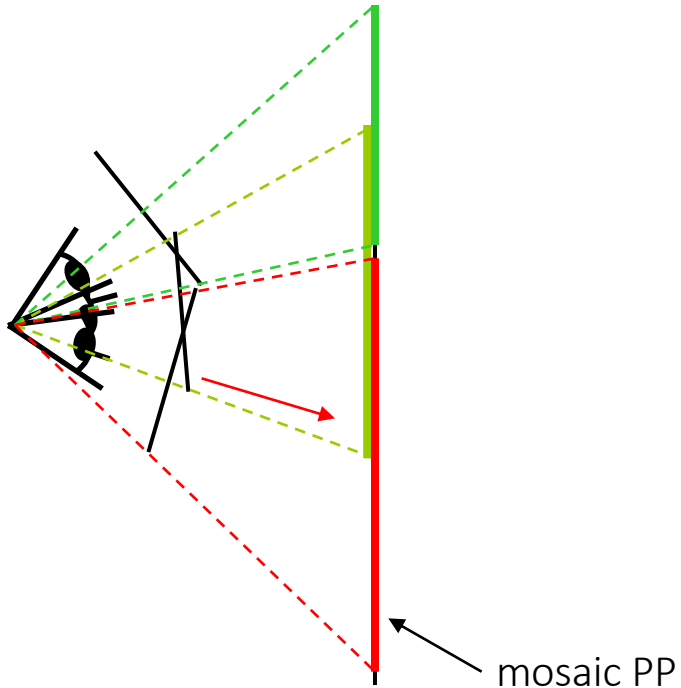


We can generate any synthetic camera view as long as it has the same center of projection!

Slide from Alyosha Efros

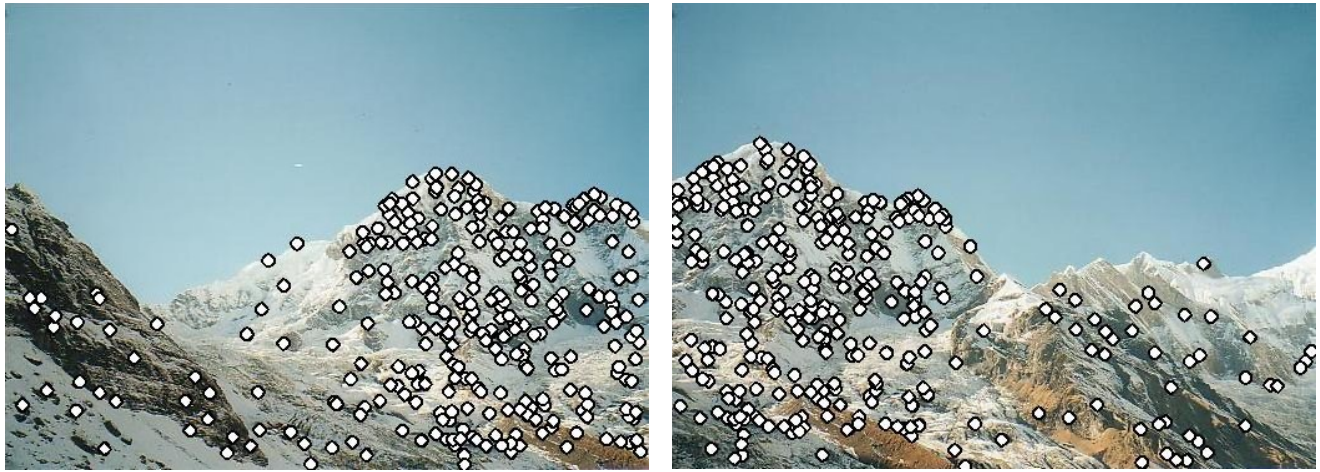
Image Reprojection

- 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*



Slide from Steve Seitz

Feature-based Alignment Outline

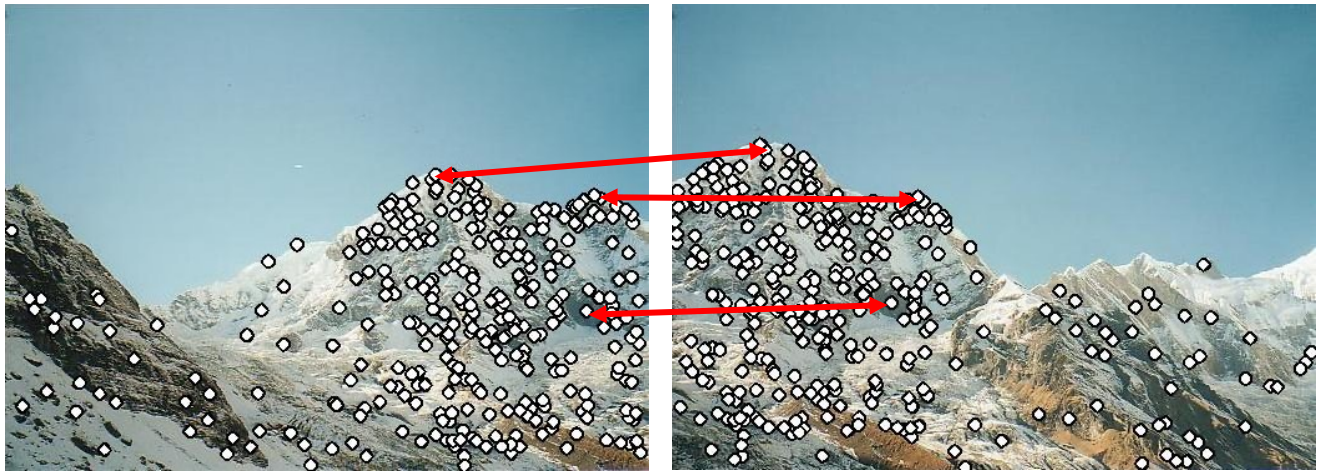


- Extract features

Feature-based Alignment Outline

- Extract features
- Compute *putative matches*

Feature-based Alignment Outline

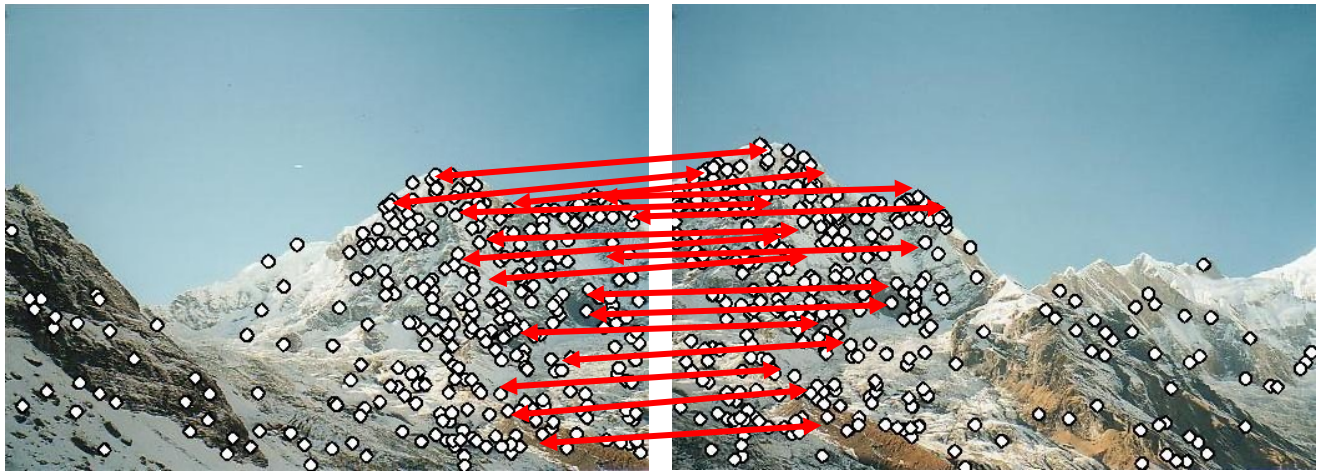


- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)

KAIST

Slide from L. Lazebnik

Feature-based Alignment Outline



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

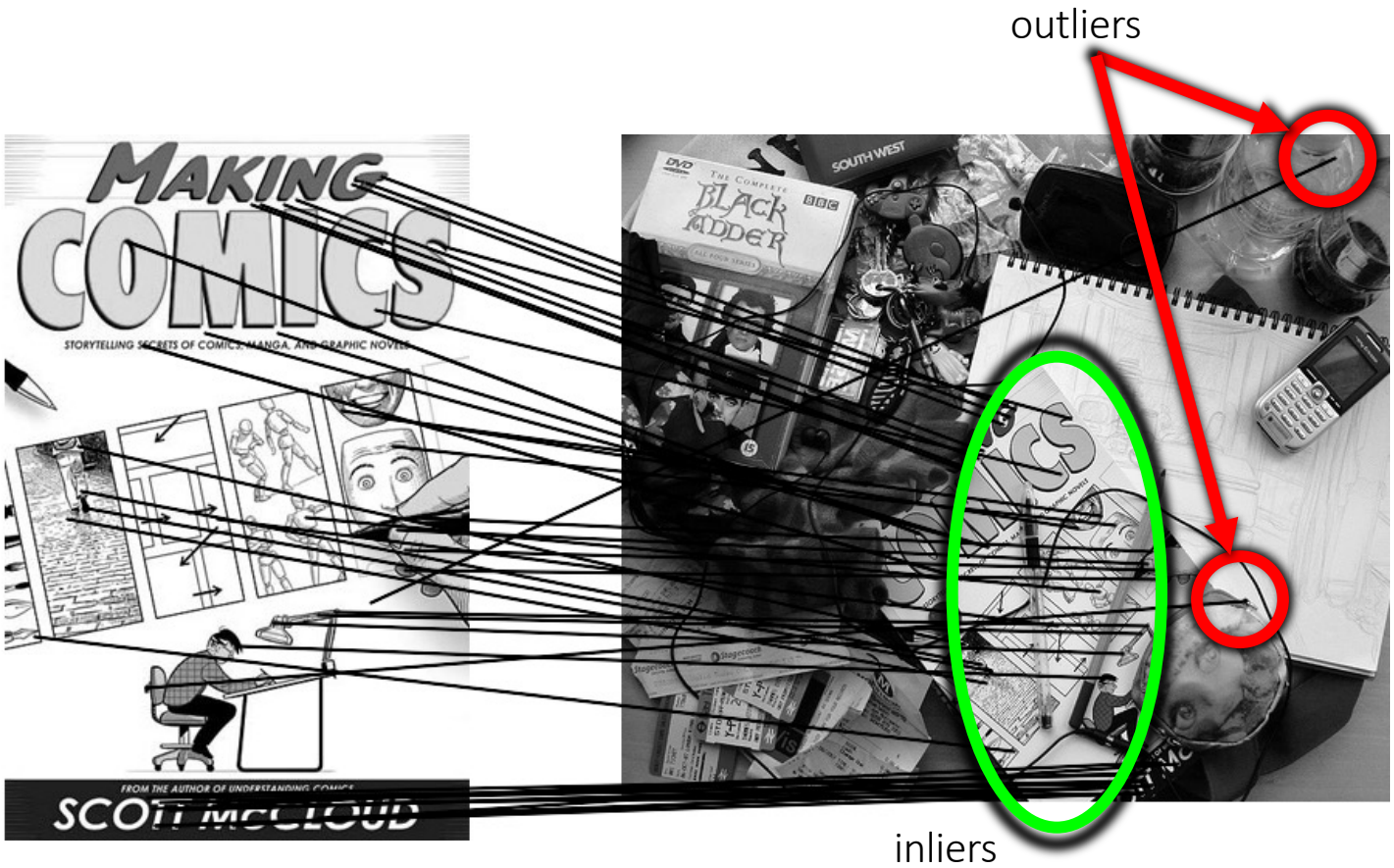
Slide from L. Lazebnik

Feature-based Alignment Outline

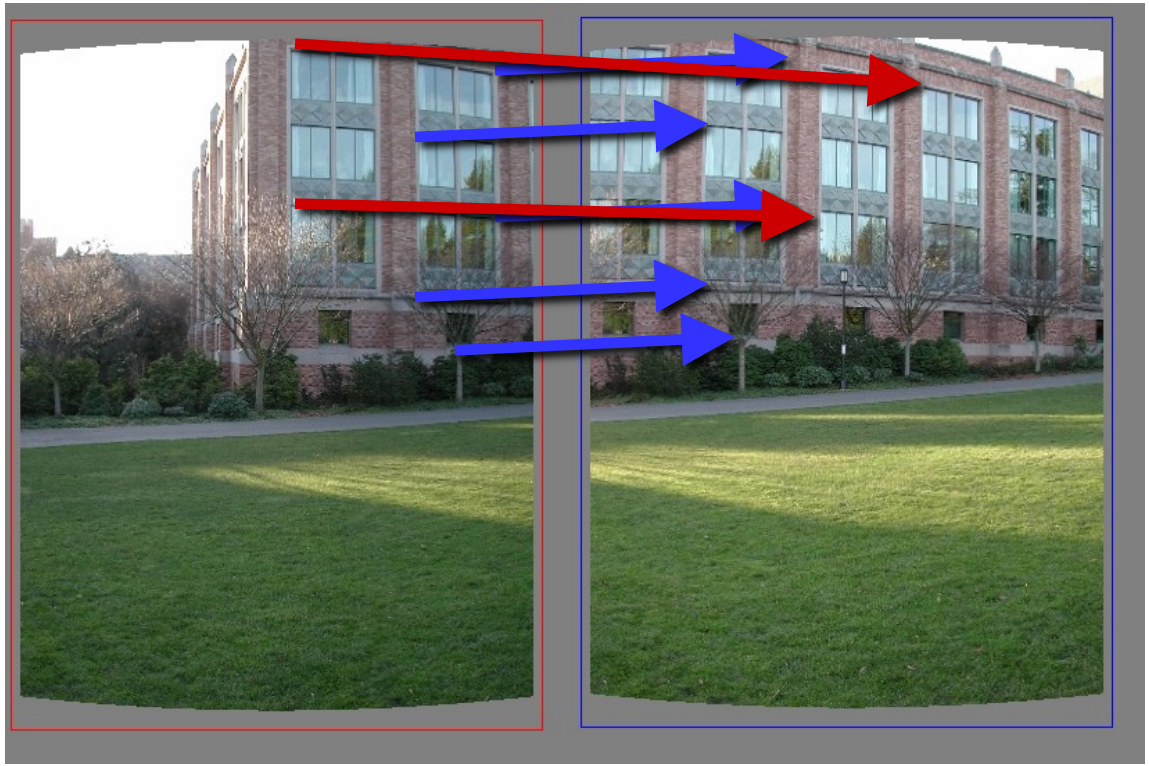
- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Slide from L. Lazebnik

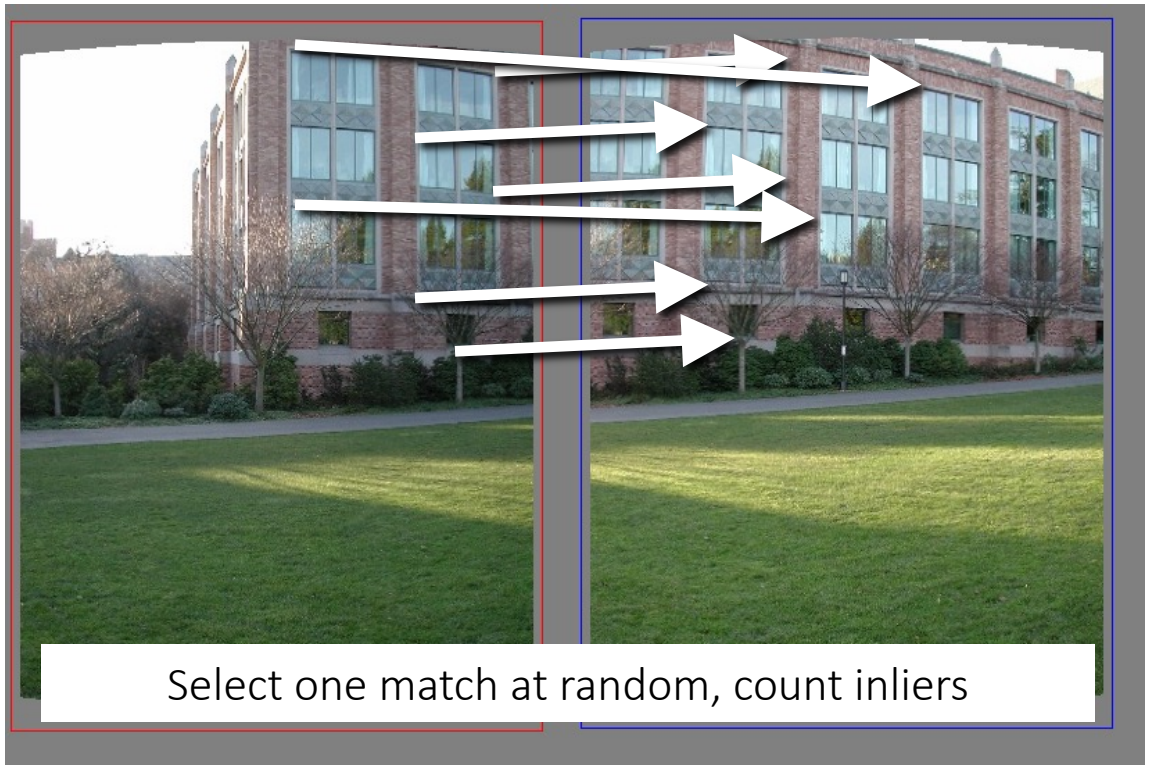
Outliers


outliers
MAKING
COMICS
STORYTELLING SECRETS OF COMICS, MANGA, AND GRAPHIC NOVELS
FROM THE AUTHOR OF UNDERSTANDING COMICS
SCOTT McCLOUD
SOUTH WEST
THE COMPLETE
BLACK ADDER
BBC
inliers

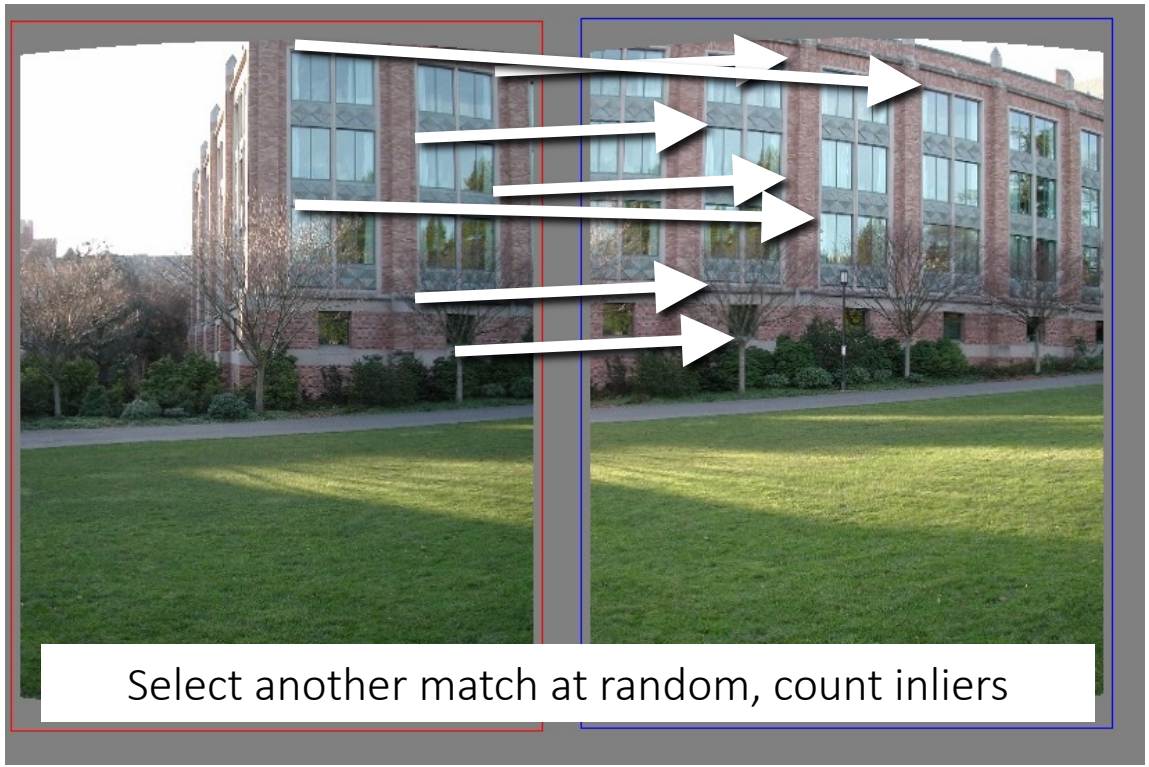
Translations



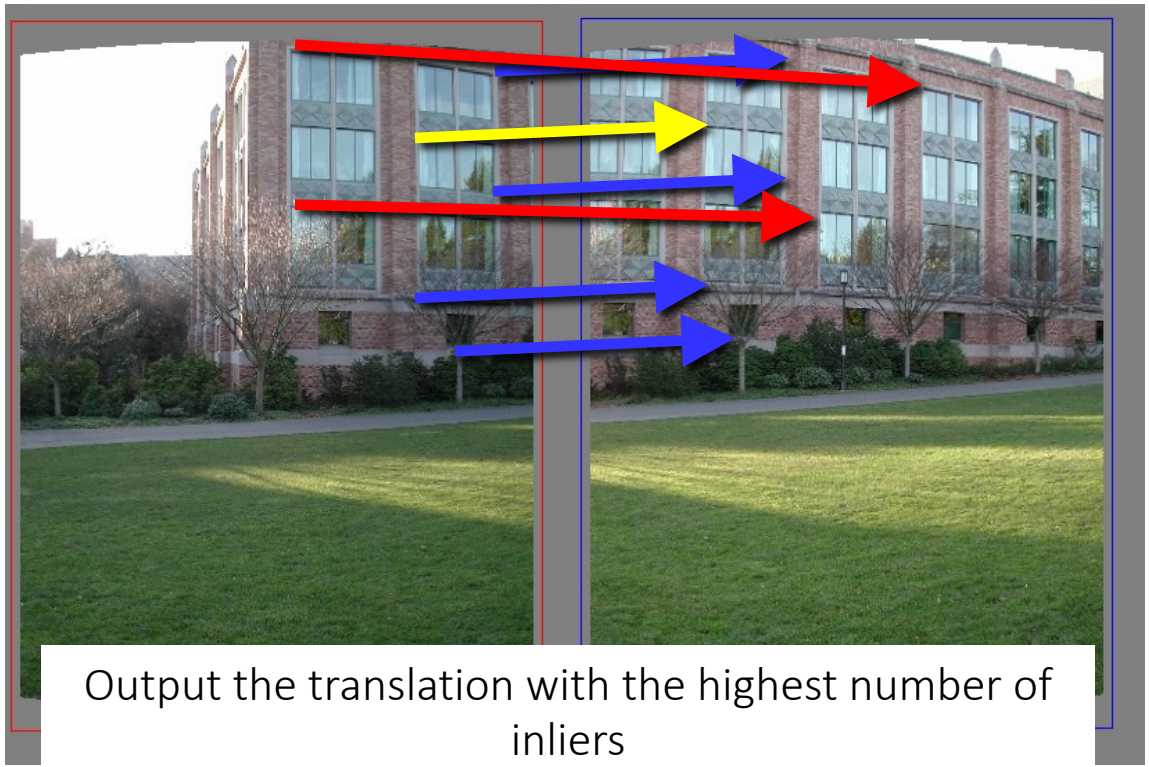
RANdom SAmple Consensus



RANdom SAmple Consensus



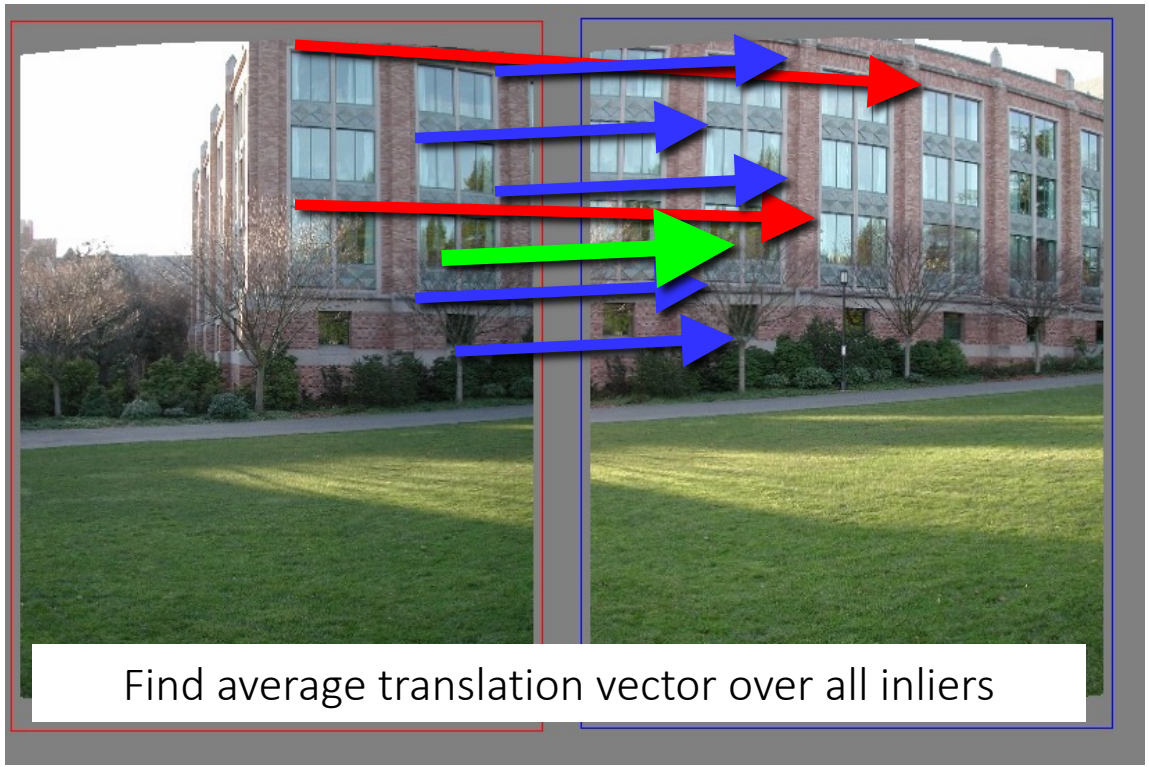
RANdom SAmple Consensus



Output the translation with the highest number of inliers

RANdom SAmple Consensus

Final step: least squares fit



Panoramas

- Now we know how to create panoramas!

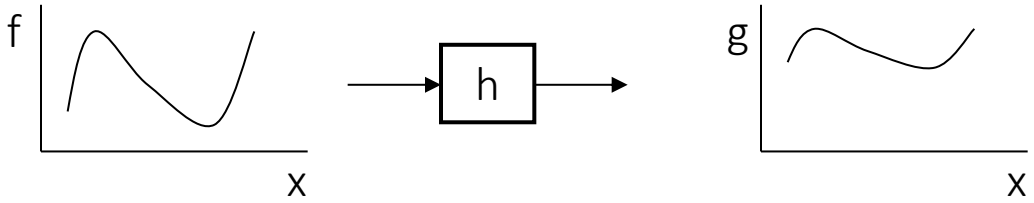
- Given two images:
  - Step 1: Detect features
  - Step 2: Match features
  - Step 3: Compute a homography using RANSAC
  - Step 4: Combine the images together (somehow)

- What if we have more than two images?

Image Warping

- image filtering: change *range* of image

$$g(x) = h(f(x))$$



- image warping: change *domain* of image

$$g(x) = f(h(x))$$

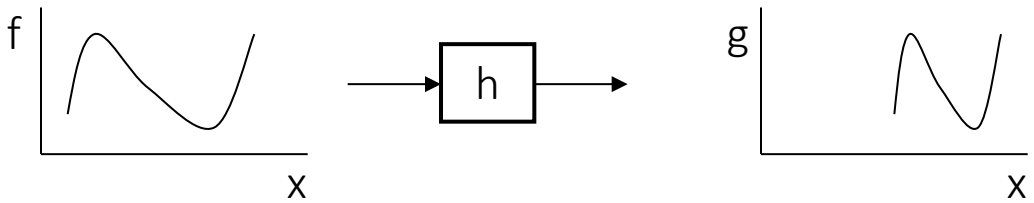
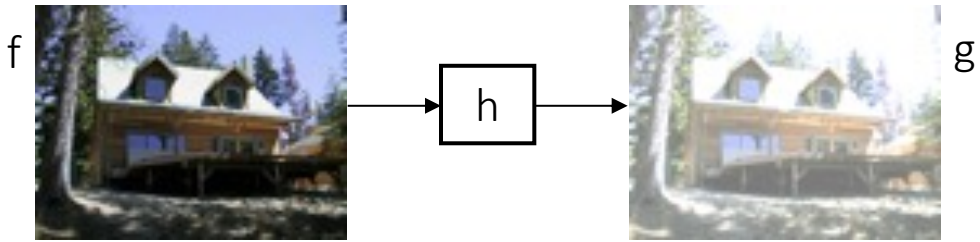


Image Warping

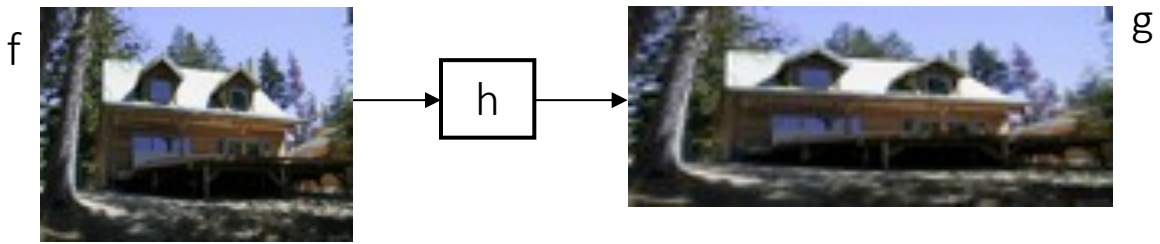
- image filtering: change *range* of image

$$g(x) = h(f(x))$$



- image warping: change *domain* of image

$$g(x) = f(h(x))$$



Parametric (Global) Warping

- Examples of parametric warps:

translation

rotation

aspect

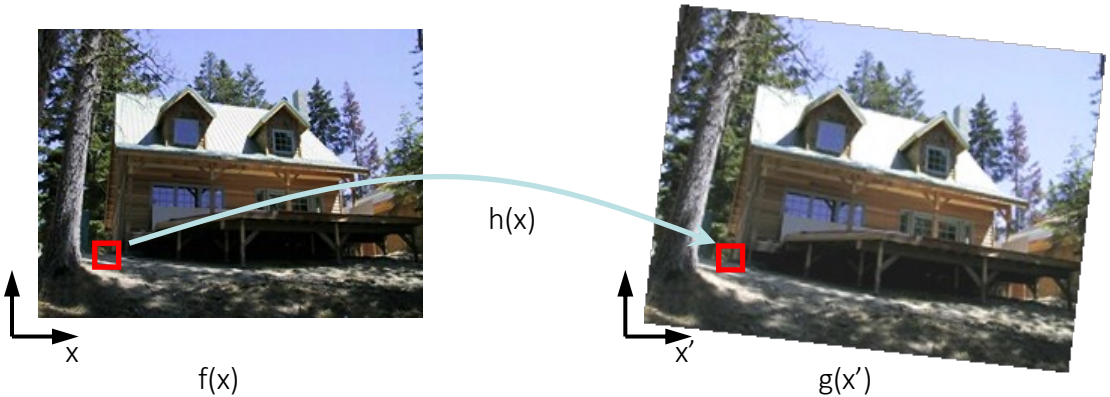
affine

perspective

cylindrical

Image Warping

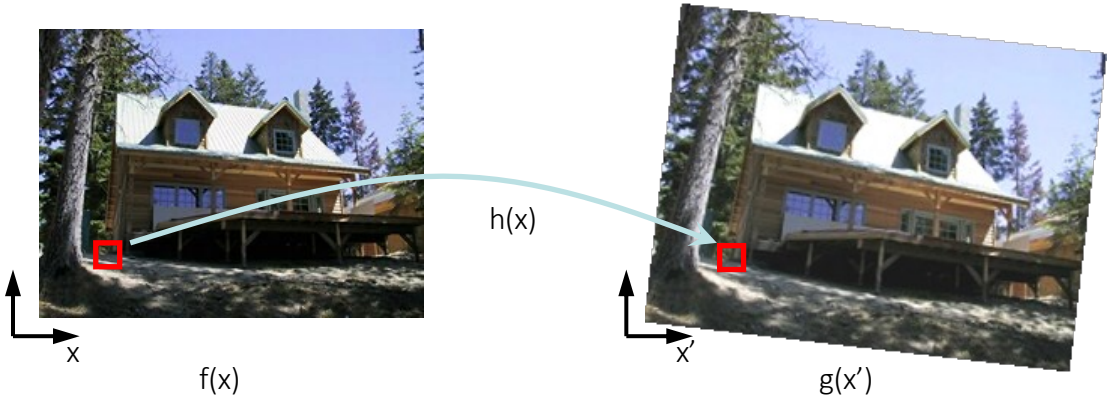
- Given a coordinate transform $x' = h(x)$ and a source image $f(x)$, how do we compute a transformed image $g(x') = f(h(x))$?



Slide from R. Szeliski

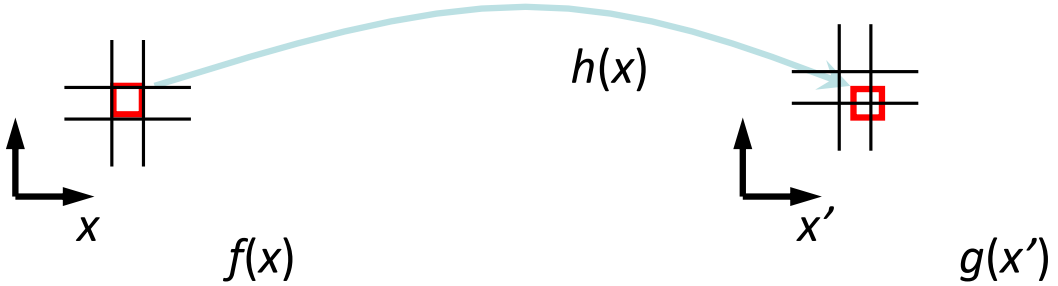
Forward Warping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$



Q: What if pixel lands "between" two pixels?

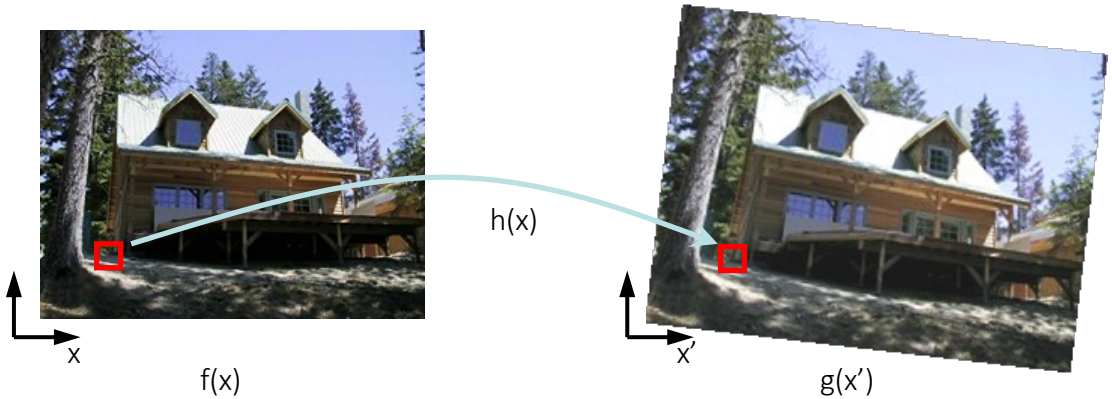
A: Add "contribution" to several pixels, normalize later (splatting)



Slide from R. Szeliski

Inverse Warping

- Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$



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

A: Interpolate color value from neighbors

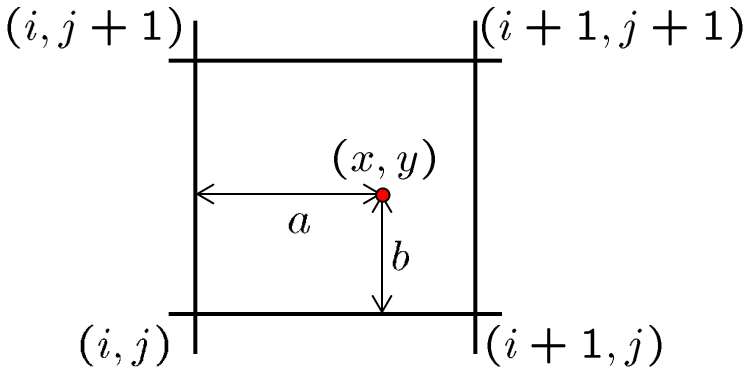
- nearest neighbor, bilinear…

Interpolation

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

Bilinear Interpolation

Sampling at *f(x,y):*



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