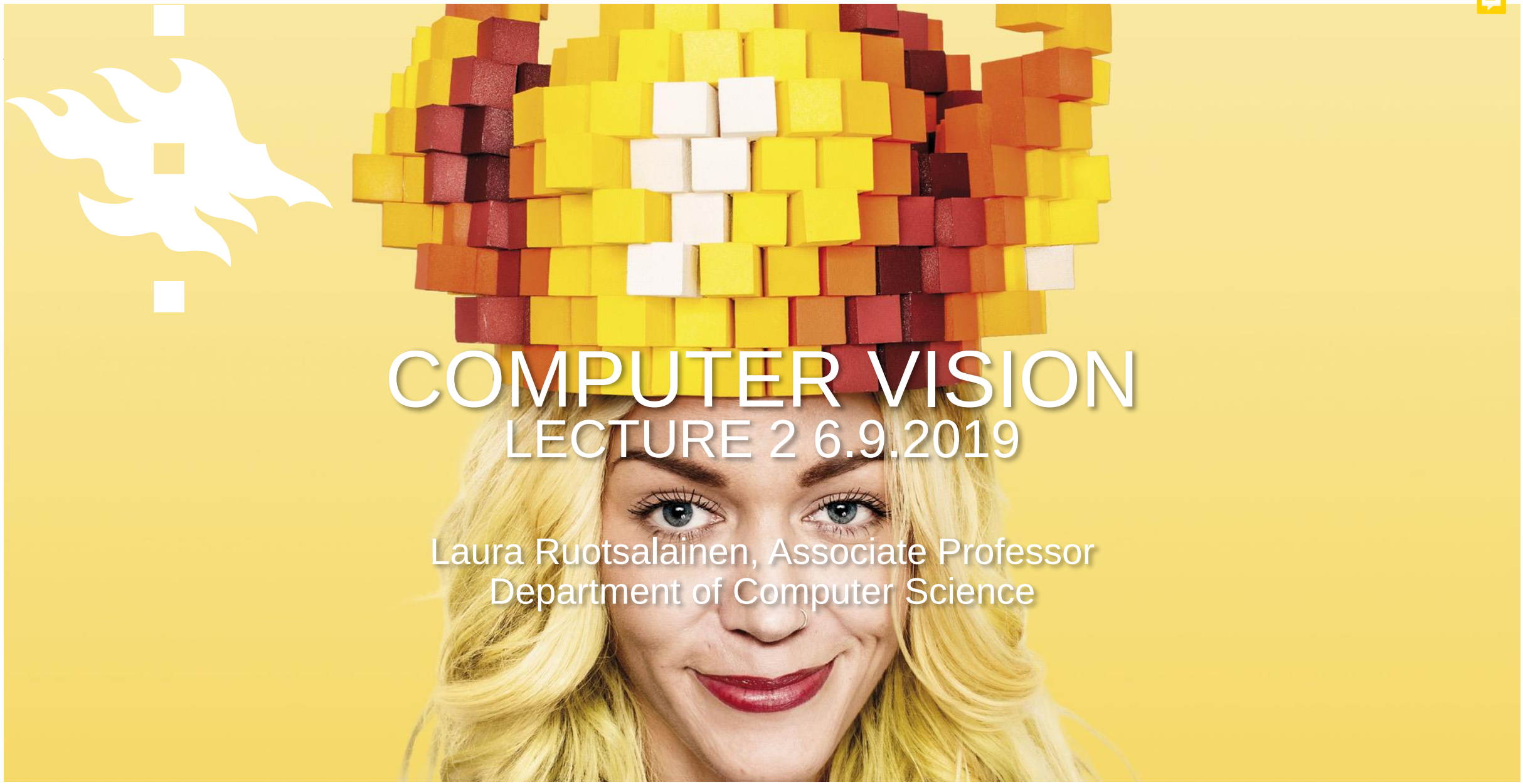


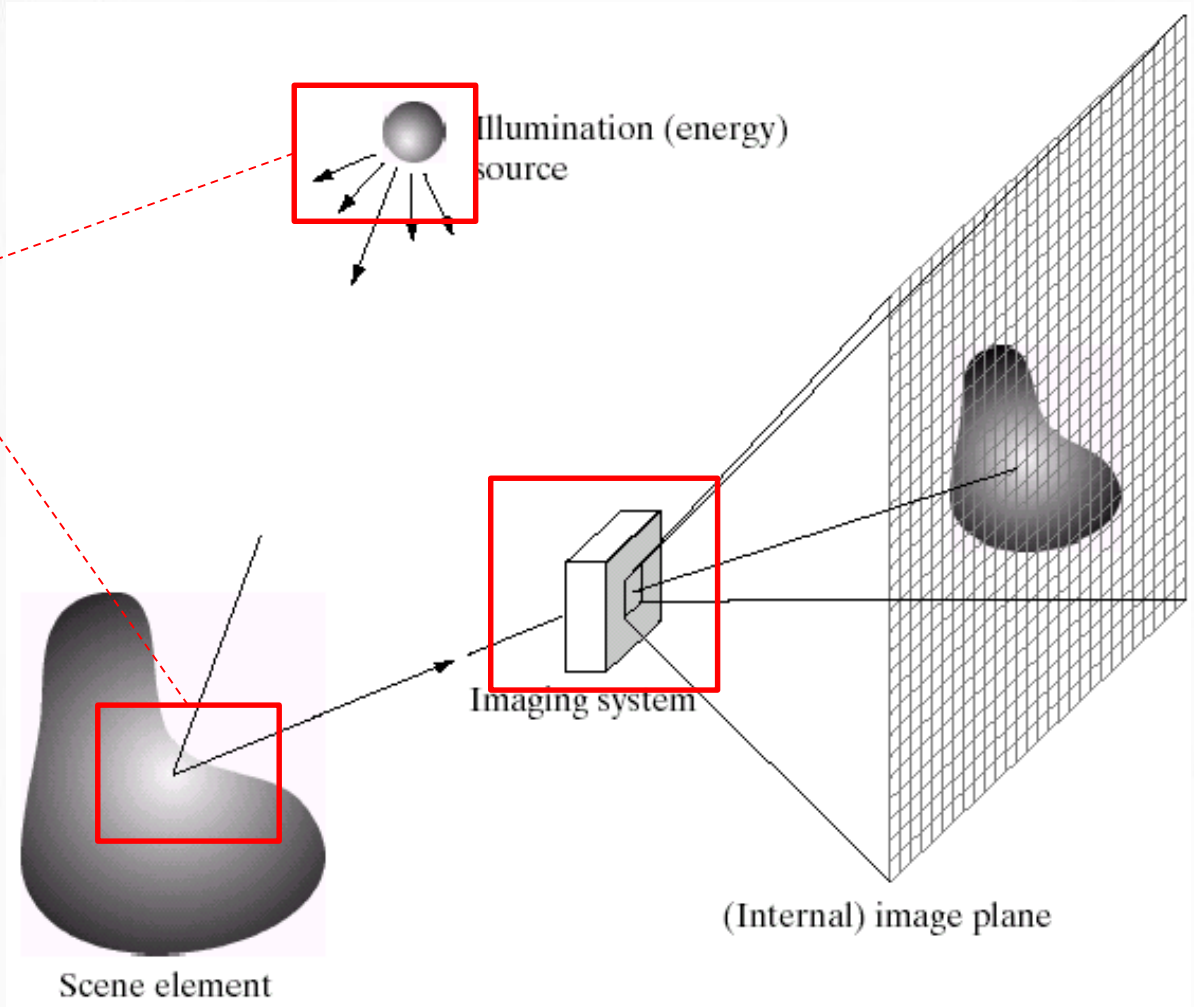
COMPUTER VISION

LECTURE 2 6.9.2019

Laura Ruotsalainen, Associate Professor
Department of Computer Science

WHAT IS AN IMAGE?

Photometric
image formation



A camera is a mapping between 3D world and a 2D image

- Geometry and transformations
- Camera models
- Camera calibration

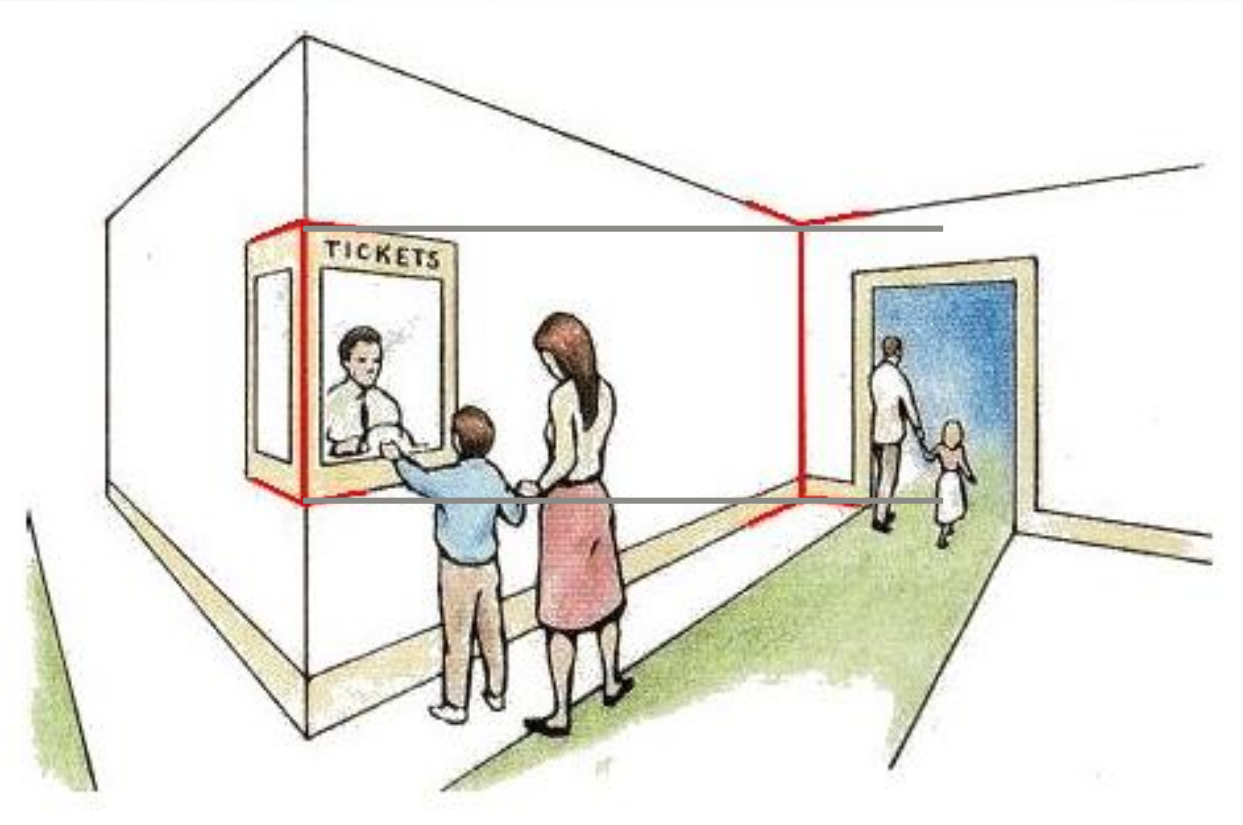
TODAY'S LECTURE

- Transformations
- Projections
- Vanishing points and lines
- Camera parameters and matrix

One view computer vision

- Szeliski Chapter 2

MÜLLER-LYER ILLUSION



http://www.michaelbach.de/ot/sze_muelue/index.html

GEOMETRY

- 3D world is based on Euclidean geometry (angles and shapes of objects)
  - Parallel lines meet at “infinity”
  - Addition of “ideal points”
- Homogeneous coordinates
  - Euclidean 2-space, point (x,y)
  - Adding an extra coordinate (x,y,1) represents the same point as well as (kx, ky, k)
  - (x,y,0) = point at infinity, form a line at infinity
- Points represented with homogeneous coordinates => Projective geometry

TRANSFORMATIONS

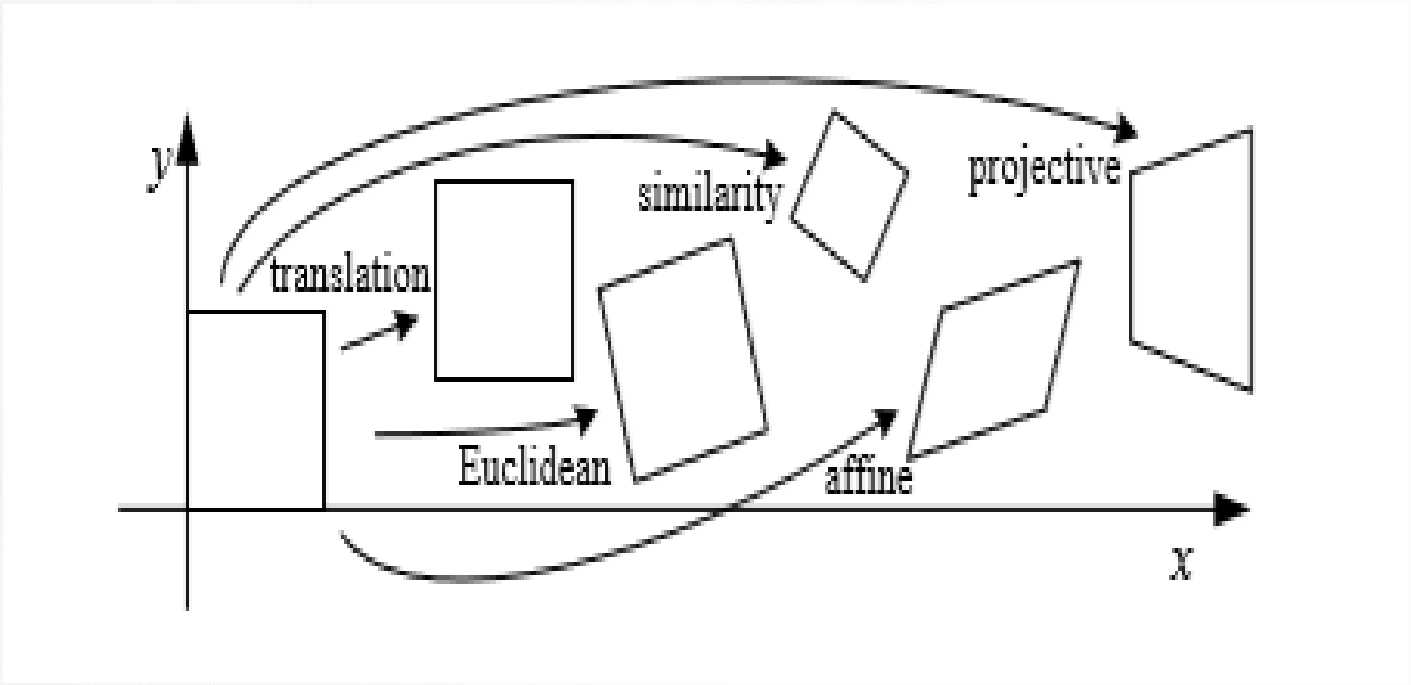
- In Computer Vision projective space is a convenient way of representing real 3D world (3D projective space, plane at infinity)
- Images are formed (usually) by projecting the world onto a 2D representation (2D projective space, line at infinity)

Homogeneous coordinates

- Represent 2D point with a 3D vector
- 3D vectors are only defined up to scale

TRANSFORMATIONS

$$\boldsymbol{x}' = f(\boldsymbol{x}; p)$$



Basic 2D planar transformations (Szeliski)

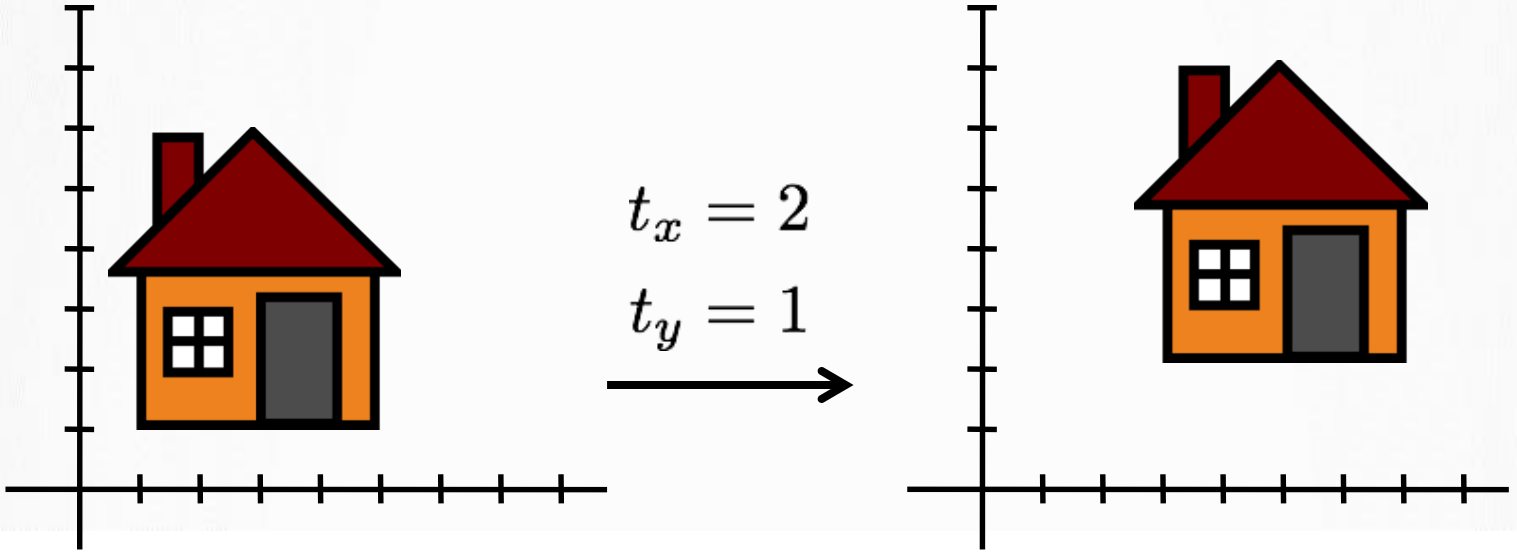
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

parameters p

point $\boldsymbol{x}$

2D TRANSLATION USING HOMOGENEOUS COORDINATES

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



Slide credit:
Kris Kitani

MATRIX COMPOSITION

Transformations can be combined by matrix multiplication:

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

p' = translation(t_x,t_y) rotation(θ) scale(s,s) **p**

Does the multiplication order matter?

Slide credit:
Kris Kitani

TRANSFORMATIONS

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{3\times4}$	3		
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{3\times4}$	6		
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{3\times4}$	7		
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{3\times4}$	12		
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{4\times4}$	15		

Hiearchy of 3D transformations (Szeliski)

Projective = perspective transformation

DoF: Degrees of Freedom
Number of independent parameters defining the configuration

AFFINE TRANSFORMATIONS

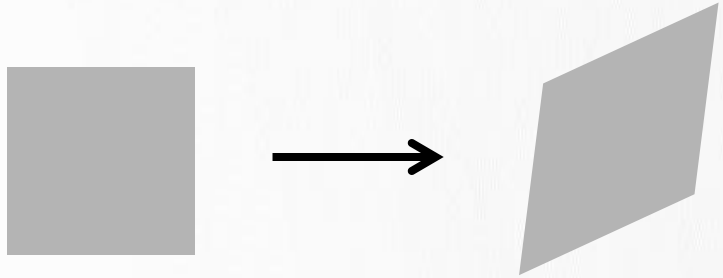
Affine transformations are combinations of

- arbitrary (4-DOF) linear transformations; and
- translations

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

Properties of affine transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines map to parallel lines
- ratios are preserved
- compositions of affine transforms are also affine transforms



PROJECTIVE TRANSFORMATIONS

Projective transformations are combinations of

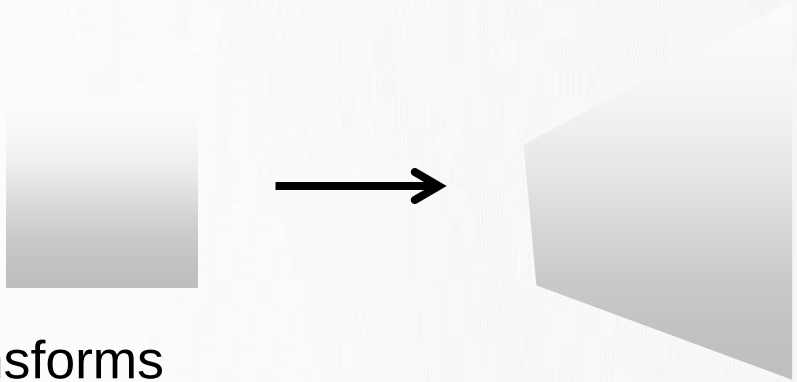
- affine transformations; and
- projective wraps

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

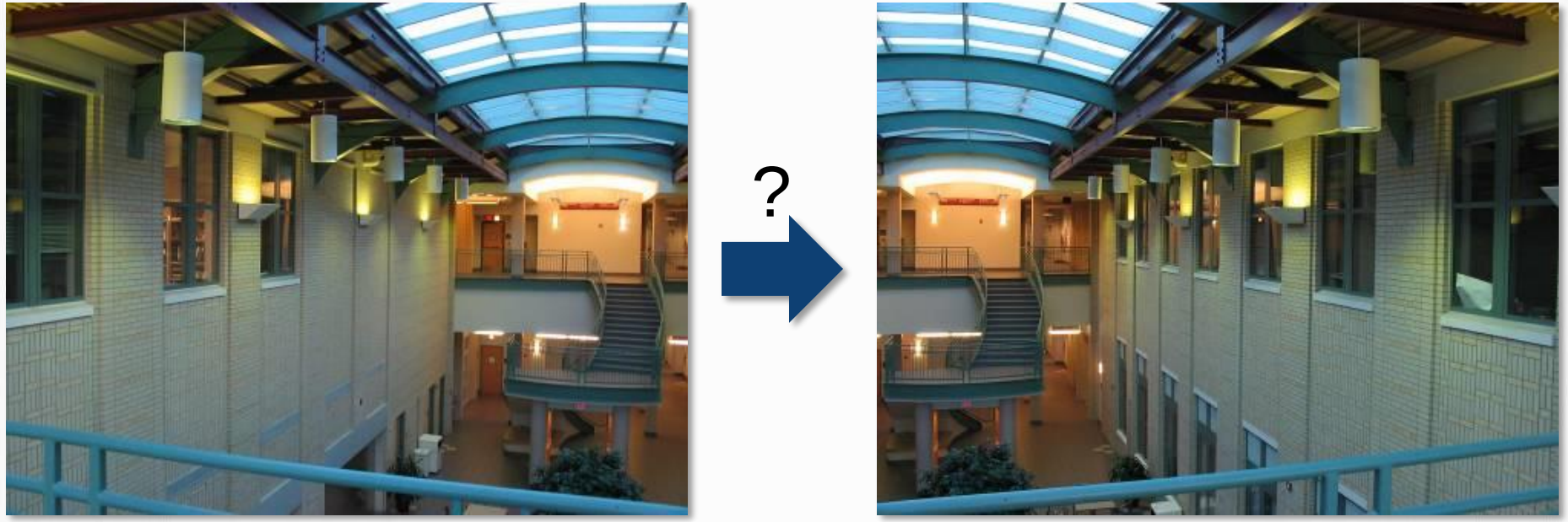
8 DOF: vectors (and therefore matrices) are defined up to scale)

Properties of projective transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved
- compositions of projective transforms are also projective transforms



WHAT IS THE GEOMETRIC RELATIONSHIP BETWEEN THESE TWO IMAGES?



Slide credit:
Noah Snavely

WHAT IS THE GEOMETRIC RELATIONSHIP BETWEEN THESE TWO IMAGES?

Very important for creating mosaics!

First, we need to know what this transformation is.

Second, we need to figure out how to compute it using feature matches.

Lecture 4,
Lecture 8

Slide credit:
Noah Snavely

PROJECTIONS 1

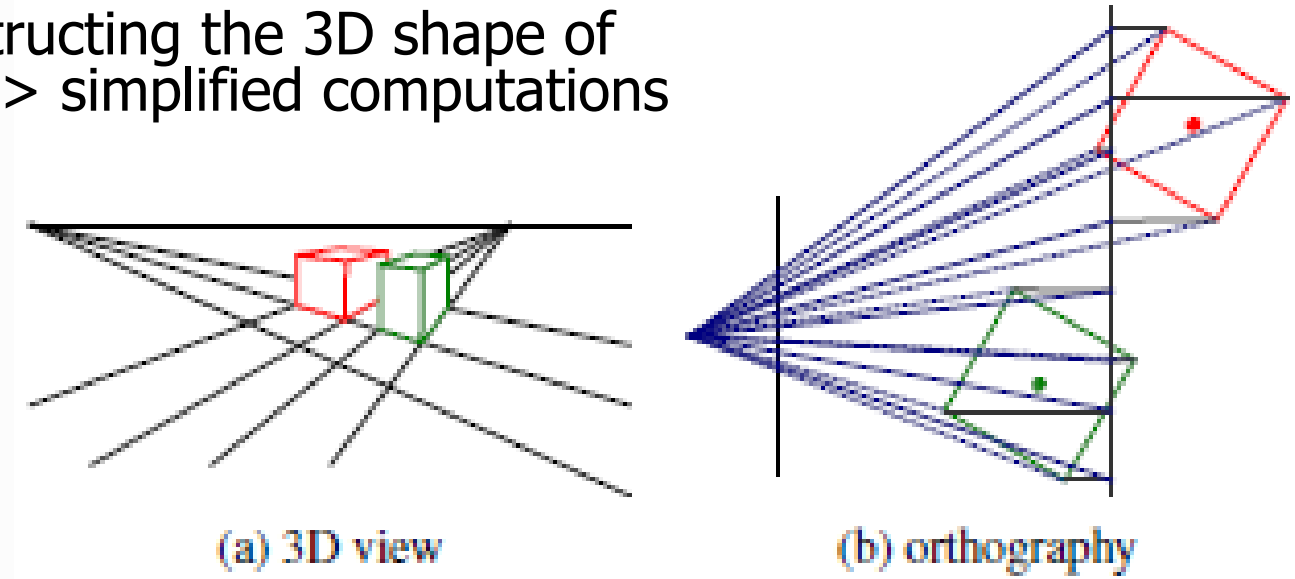
- Usually images are 2D **perspective** projections of the 3D scene
  - Object $X = [\ X\ Y\ Z]^T$, image $x= [x\ y]^T$
  - Homogenous: object $X = [X\ Y\ Z\ 1]^T$, image $x = [x\ y\ 1]^T$
  - Size of the object in the image is dependent on its distance from the camera
- We can do the projection (mapping) by using **linear** 3D to 2D projection matrix

PROJECTIONS 2

- Also a simple model called **orthography** is used due to its simplicity
  - Just drops the Z component of the 3D object coordinate
  - Approximate model for long focal length (telephoto) and objects whose depth is shallow relative to their distance to the camera
  - Scaled orthography used for reconstructing the 3D shape of objects far away from the camera => simplified computations

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

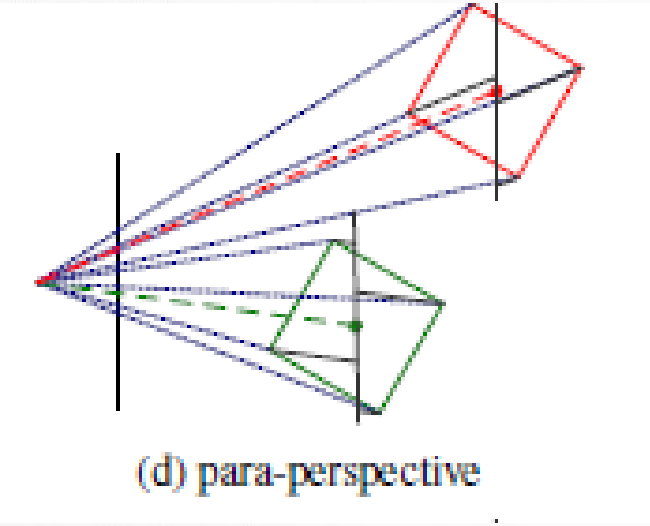
$$X = [x\ y\ z\ 1]^T$$



(a) 3D view (b) orthography

PROJECTIONS 3

- **Affine** projections are para-perspective
  - First projected onto a local reference frame parallel to the image plane, then projection onto image plane
  - More accurate than orthography, less complex than perspective



(d) para-perspective

$$x = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ 0 & 0 & 0 & 1 \end{bmatrix} X$$

A is an arbitrary matrix

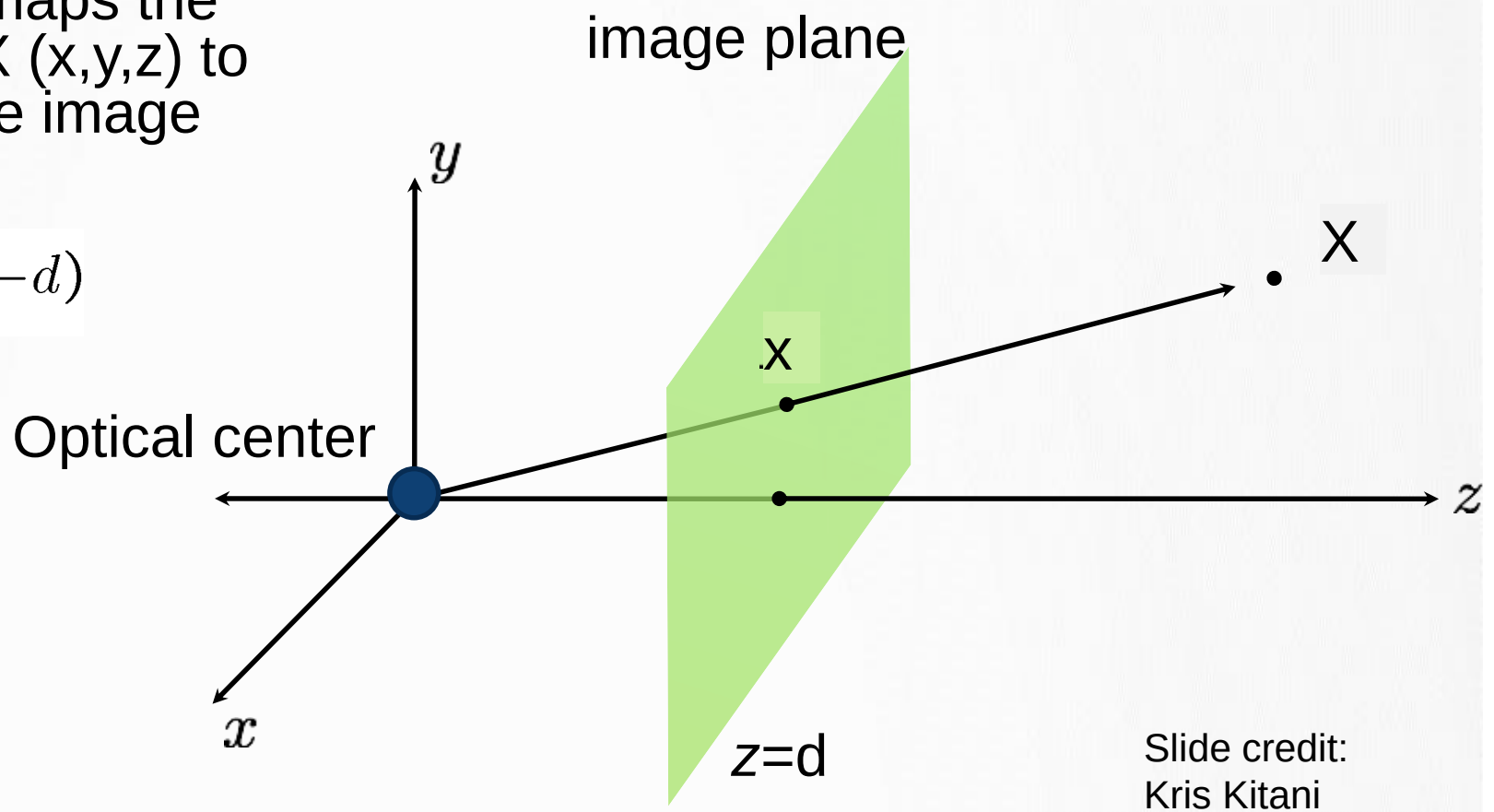
PROJECTIONS 4

- **Perspective** projection maps the intersection of ray from X (x,y,z) to the optical center with the image plane

$$(x, y, z) \rightarrow (-d\frac{x}{z}, \ -d\frac{y}{z}, \ -d)$$

- Perspective projection deletes the last component

$$(x, y, z) \rightarrow (-d\frac{x}{z}, \ -d\frac{y}{z})$$



Slide credit: Kris Kitani

PERSPECTIVE PROJECTION

Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**

- (Can also represent as a 4x4 matrix – OpenGL does something like this)

Slide credit:
Kris Kitani

PERSPECTIVE PROJECTION

How does scaling the projection matrix change the transformation?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

Slide credit:
Kris Kitani

MODELING PROJECTION

Is this a linear transformation?

- no—division by z is nonlinear

Homogeneous coordinates to the rescue

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

homogeneous image coordinates

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

homogeneous scene coordinates

Converting *from* homogeneous coordinates

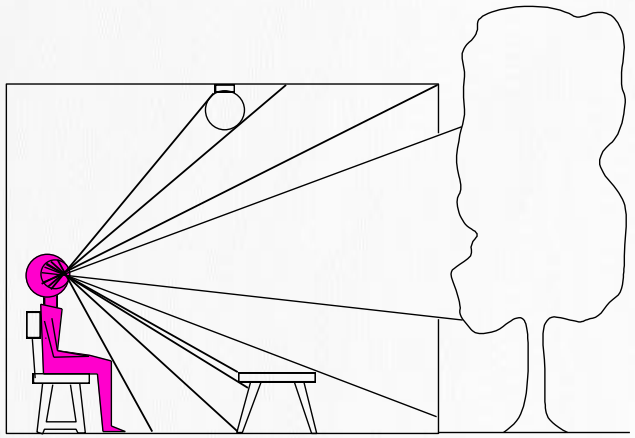
$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

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

Slide credit: Kris Kitani

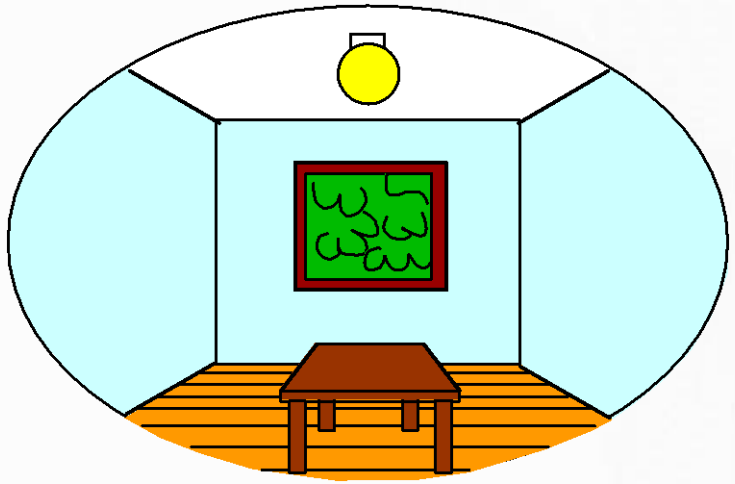
DIMENSIONALITY REDUCTION (3D TO 2D)

3D world



Point of observation

2D image



What have we lost?

- Angles
- Distances (lengths) => it is not possible to resolve the distance of the object from the image (could use range sensors, stereo matching)

PROJECTION PROPERTIES

- Many-to-one: any points along same ray map to same point in image
- Points → points
- Lines → lines (collinearity is preserved)

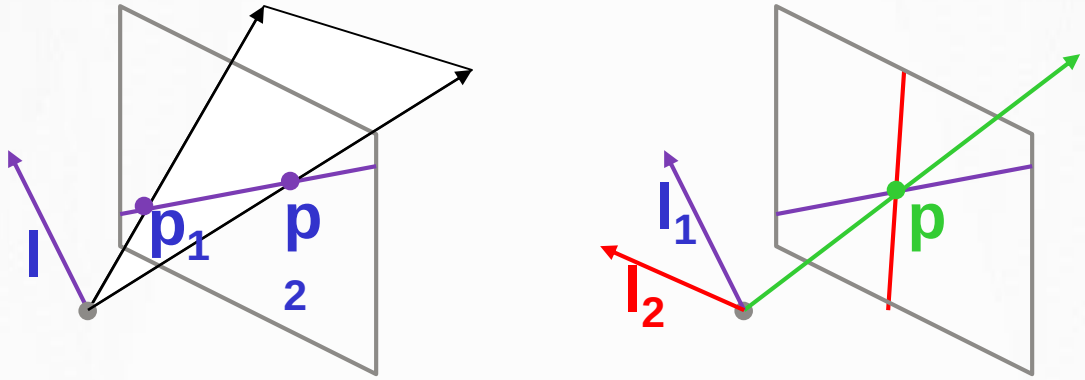
  But line through focal point projects to a point
- Planes → planes (or half-planes)

  But plane through focal point projects to line

POINT AND LINE DUALITY

A line **l** is a homogeneous 3-vector

It is ⊥ to every point (ray) **p** on the line: **l**·**p**=0



What is the line **l** spanned by rays $\mathbf{p}_1$ and $\mathbf{p}_2$?

- **l** is ⊥ to $\mathbf{p}_1$ and $\mathbf{p}_2$ ⇒ $\mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$
- **l** can be interpreted as a *plane normal*

What is the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$?

- **p** is ⊥ to $\mathbf{l}_1$ and $\mathbf{l}_2$ ⇒ $\mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2$

Points and lines are *dual* in projective space

Slide credit:
Noah Snavely

VANISHING POINTS 1

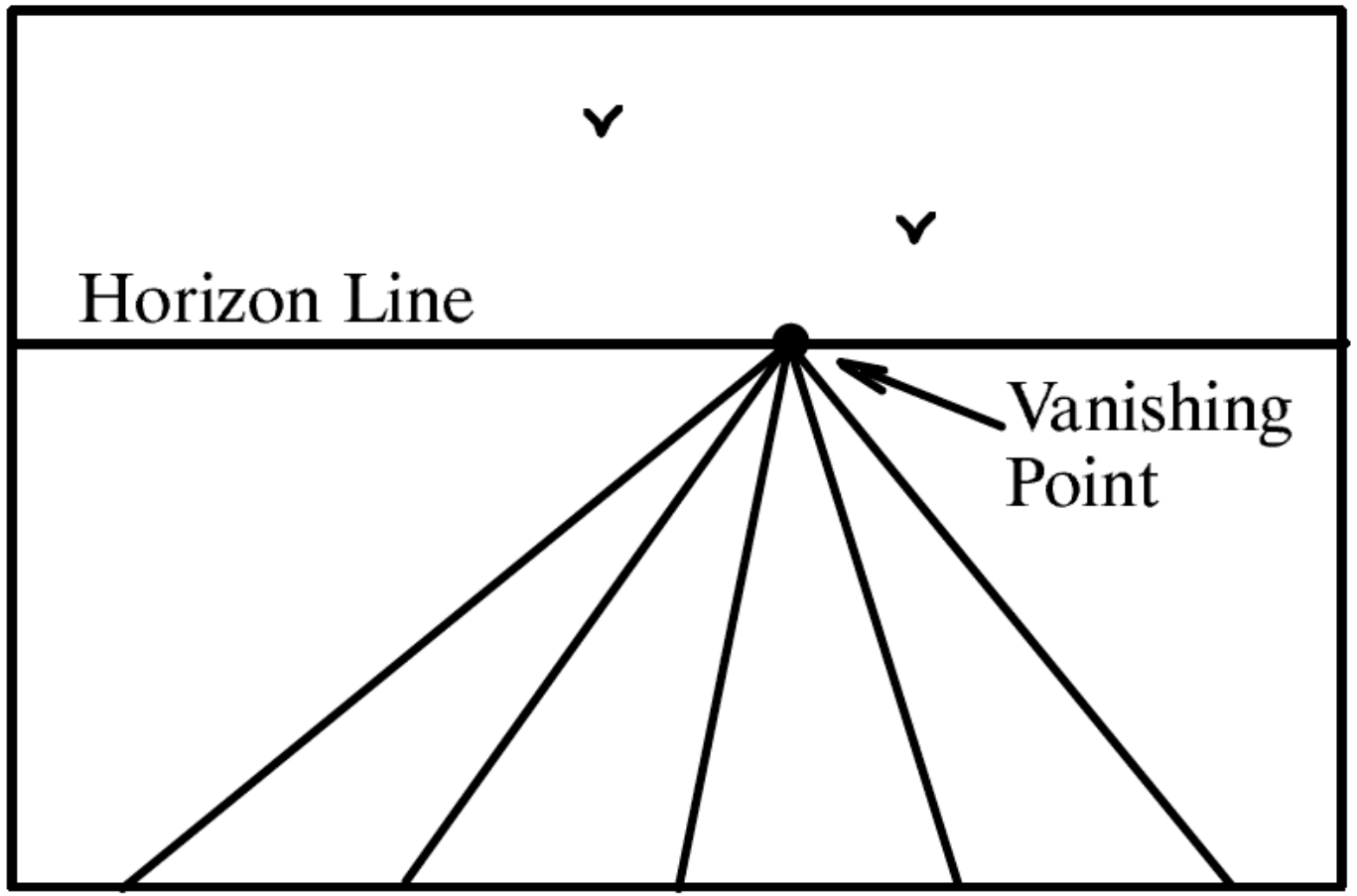
A virtual point where 3D parallel lines intersect in an image = projection of a point at infinity

Can often (but not always) project to a finite point in the image

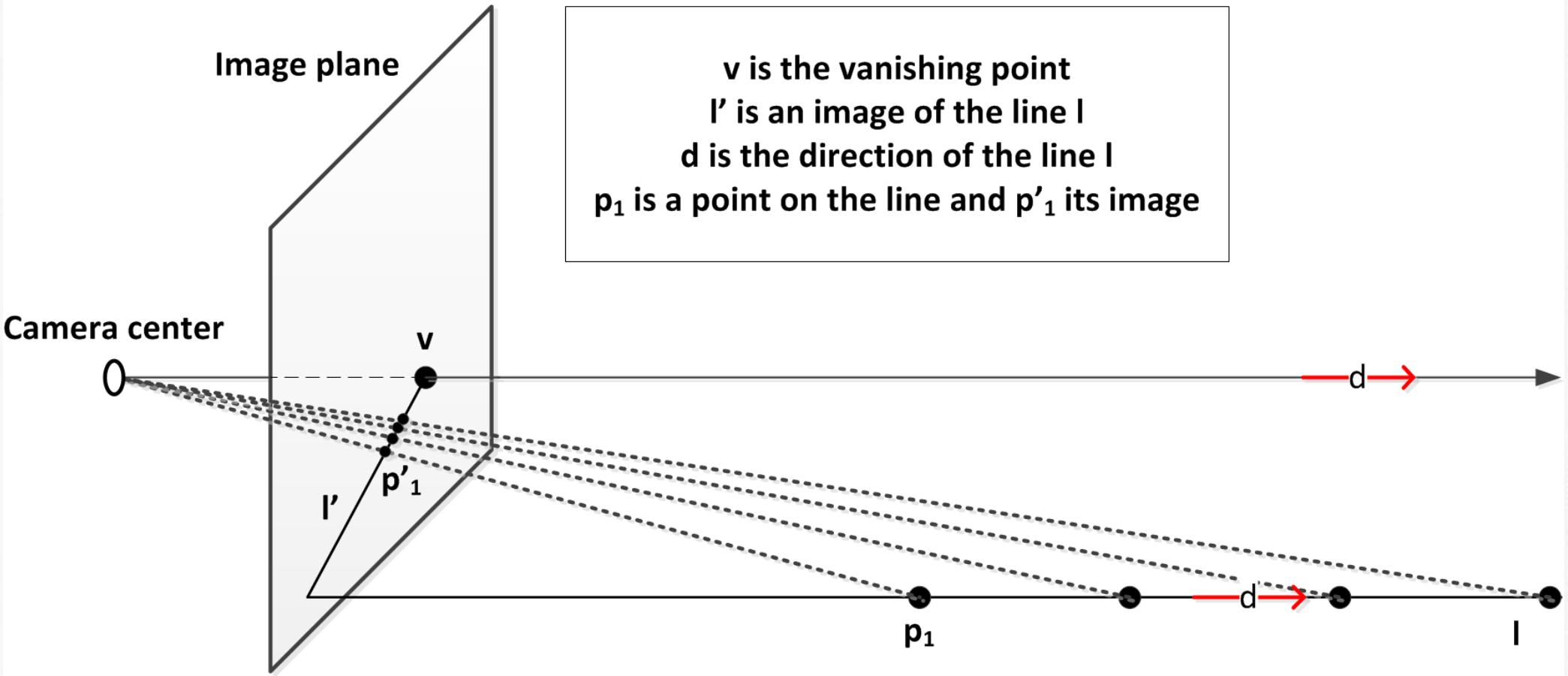
An image may have more than one vanishing point

- in fact, every image point is a potential vanishing point

Vanishing points will provide information about the change of camera's orientation



VANISHING POINTS 2



Any two parallel lines (in 3D) have the same vanishing point **v**
The ray from **C** through **v** is parallel to the lines

VANISHING POINTS 3

Vanishing points are important for determining e.g. the camera pose and its calibration information

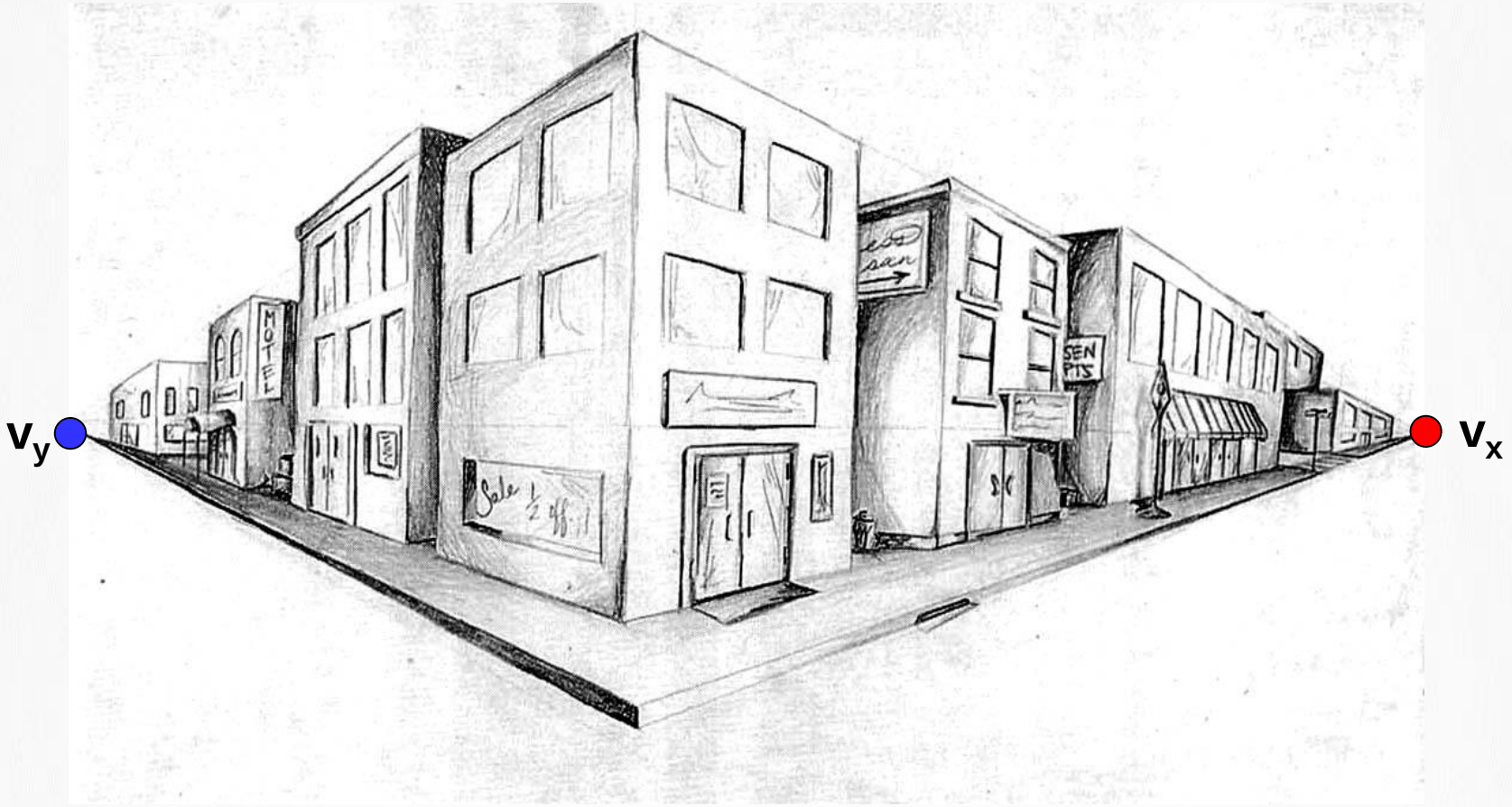
(a) (b) (c)

Figure 4.45 Real-world vanishing points: (a) architecture (Sinha, Steedly, Szeliski *et al.* 2008), (b) furniture (Mičušík, Wildenauer, and Košecká 2008) © 2008 IEEE, and (c) calibration patterns (Zhang 2000).

ONE-POINT PERSPECTIVE

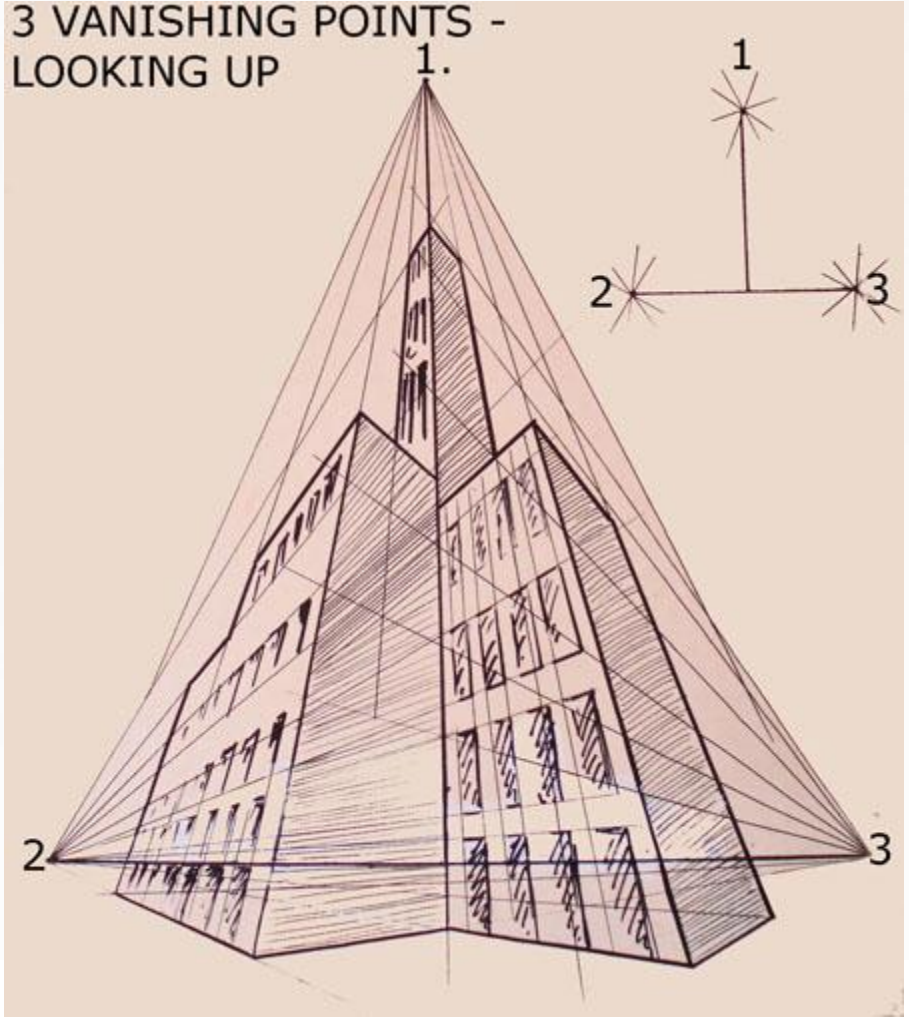
Slide credit:
Noah Snavely

TWO-POINT PERSPECTIVE



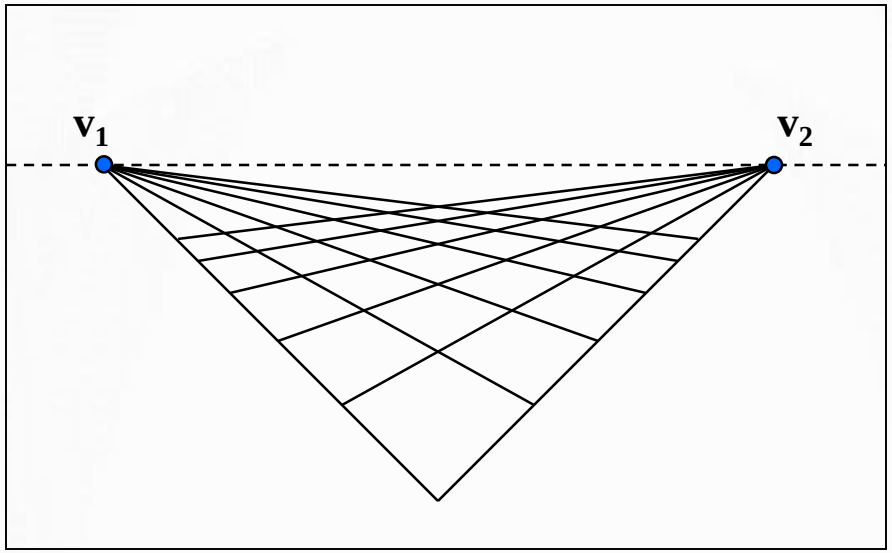
Slide credit:
Noah Snavely

THREE-POINT PERSPECTIVE



Slide credit:
Noah Snavely

VANISHING LINES



Multiple Vanishing Points

Any set of parallel lines on the plane define a vanishing point

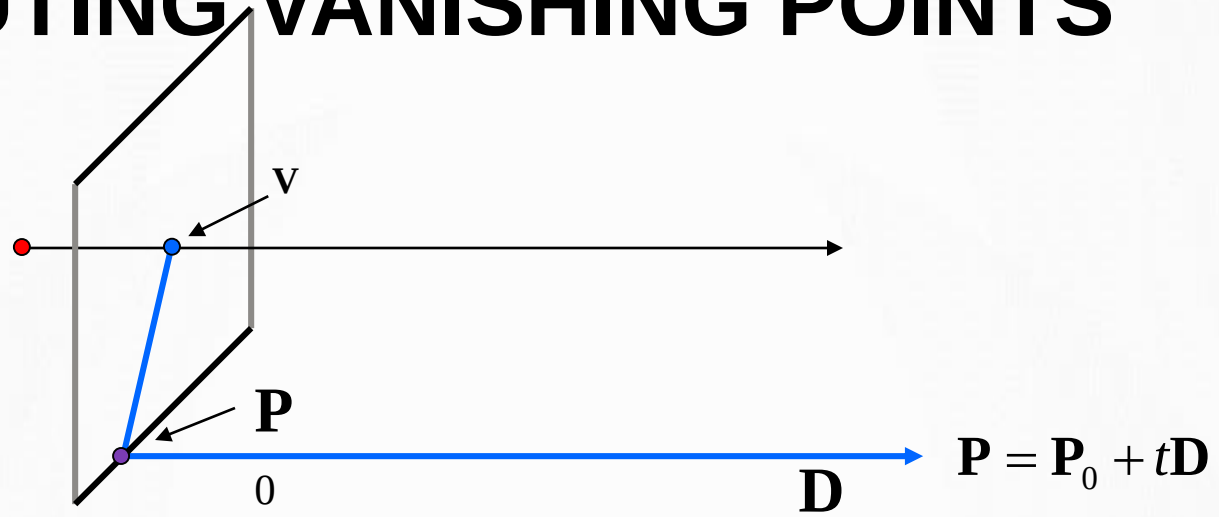
The union of all of these vanishing points is the *horizon line*

– also called *vanishing line*

Note that different planes (can) define different vanishing lines

Slide credit:
Noah Snavely

COMPUTING VANISHING POINTS



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

Properties

$$\mathbf{v} = \Pi\mathbf{P}_\infty$$

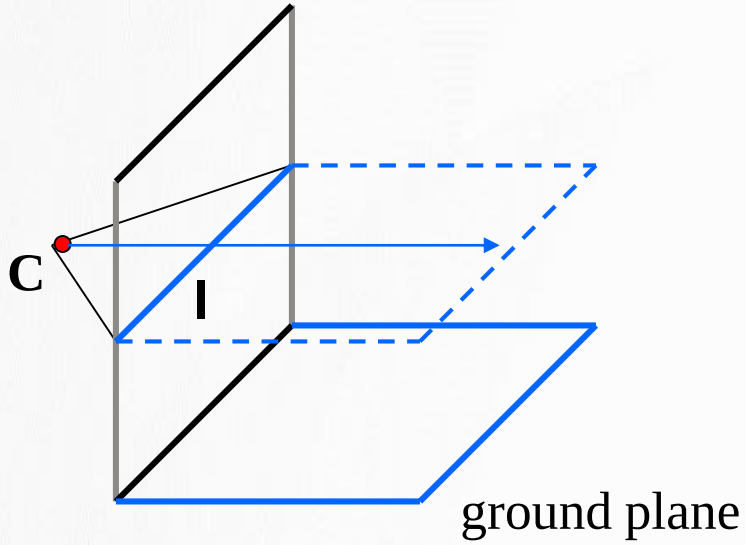
$\mathbf{P}_\infty$ is a point at *infinity*, **v** is its projection

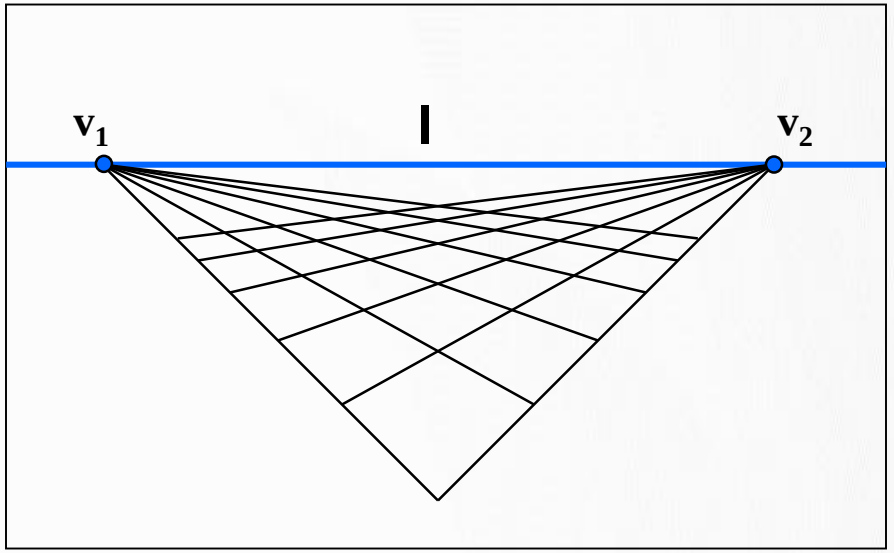
Depends only on line *direction*

Parallel lines $\mathbf{P}_0 + t\mathbf{D}$, $\mathbf{P}_1 + t\mathbf{D}$ intersect at $\mathbf{P}_\infty$

Slide credit:
Noah Snavely

COMPUTING VANISHING LINES





Properties

l is intersection of horizontal plane through **C** with image plane

Compute **l** from two sets of parallel lines on ground plane

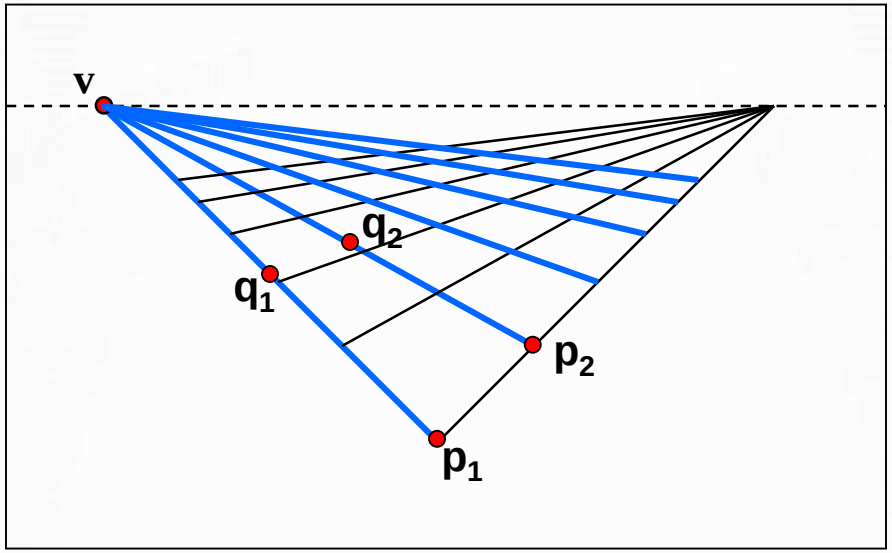
All points at same height as **C** project to **l**

– points higher than C project above l

Provides way of comparing height of objects in the scene

Slide credit:
Noah Snavely

COMPUTING VANISHING POINTS (FROM LINES)



Intersect p_1q_1 with p_2q_2

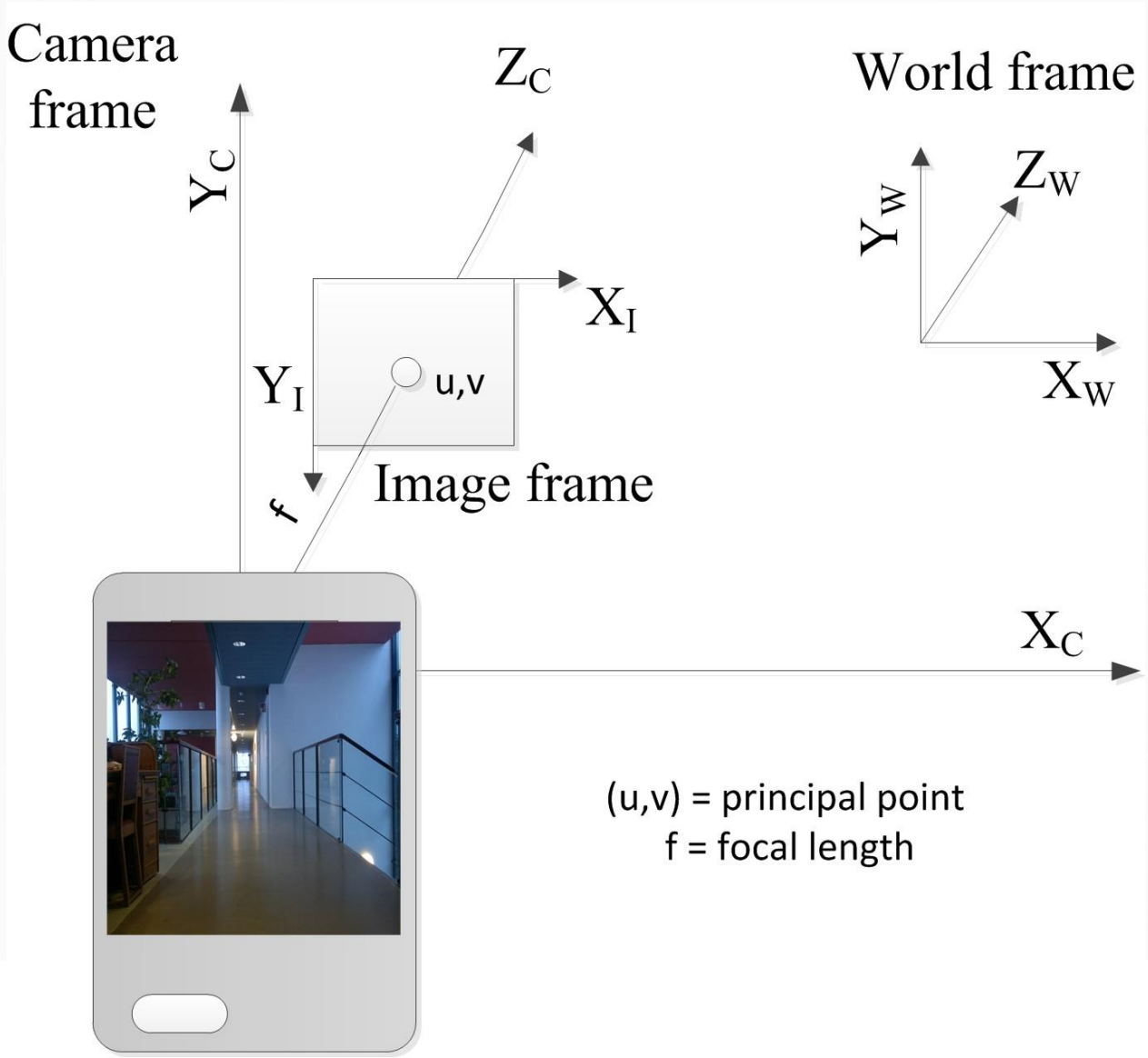
$$v = (p_1 \times q_1) \times (p_2 \times q_2)$$

Least squares version

- Better to use more than two lines and compute the “closest” point of intersection

Slide credit:
Noah Snavely

COORDINATE FRAMES



(u,v) = principal point
f = focal length

CAMERA PARAMETERS

To project a point (*x*,*y*,*z*) in *world* coordinates into a camera

- First transform (*x*,*y*,*z*) into *camera* coordinates
- Need to know
  - Camera position (in world coordinates) => **t**
  - Camera orientation (in world coordinates) => **R**
- Then project into the image plane to get a pixel coordinate
  - Need to know camera *intrinsics* => matrix **K**

CAMERA MATRIX P

$$\boldsymbol{x} = \mathbf{PX}$$

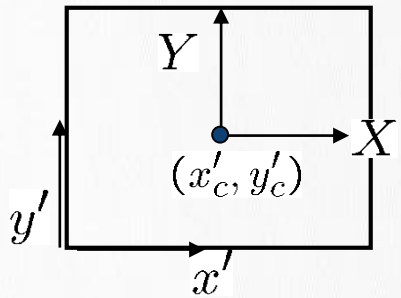
$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous image coordinates	camera matrix	homogeneous world coordinates
3 x 1	3 x 4	4 x 1

CAMERA PARAMETERS

A camera is described by several parameters

- Translation T of the optical center from the origin of world coords
- Rotation R of the image plane
- focal length f, principal point (x'_c, y'_c), pixel size (s_x, s_y)
- blue parameters are called "extrinsics," red are "intrinsics"



Projection equation

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{P}\mathbf{X}$$

- The projection matrix models the cumulative effect of all parameters

identity matrix

$$\mathbf{P} = \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

intrinsics projection rotation translation

Slide credit:
Kris Kitani

PROJECTION MATRIX

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

intrinsics

projection | rotation | translation

The **K** matrix converts 3D rays in the camera's coordinate system to 2D image points in image (pixel) coordinates.

This part converts 3D points in world coordinates to 3D rays in the camera's coordinate system. There are 6 parameters represented (3 for position/translation, 3 for rotation).

Slide credit: Kris Kitani

CAMERA (PROJECTION) MATRIX

$$\mathbf{P} = \mathbf{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}}_{\text{translation}}$$

intrinsics

$$\begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{c} \end{bmatrix}$$

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} & -\mathbf{t} \end{bmatrix}$$

Slide credit:
Kris Kitani

ANOTHER REPRESENTATION OF P

- Image points and camera (projection) matrix P may be represented also using the following forms, which might be convenient for some computations

$$x = \frac{1}{Z} PX \qquad => \qquad \left\{ x = \frac{p_1 \cdot X}{p_3 \cdot X} \,,\;\; y = \frac{p_2 \cdot X}{p_3 \cdot X} \right\}$$

p_i^T are the rows of P

$$P = \begin{bmatrix} \alpha_x \mathbf{r}_1^T + s\mathbf{r}_2^T + \beta_x \mathbf{r}_3^T & \alpha_x t_1 + s t_2 + \beta_x t_3 \\ \alpha_y \mathbf{r}_2^T + \beta_y \mathbf{r}_3^T & \alpha_y t_2 + \beta_y t_3 \\ \mathbf{r}_3^T & t_3 \end{bmatrix}$$

r_i^T = rows of R
t_i = coordinates of vector t

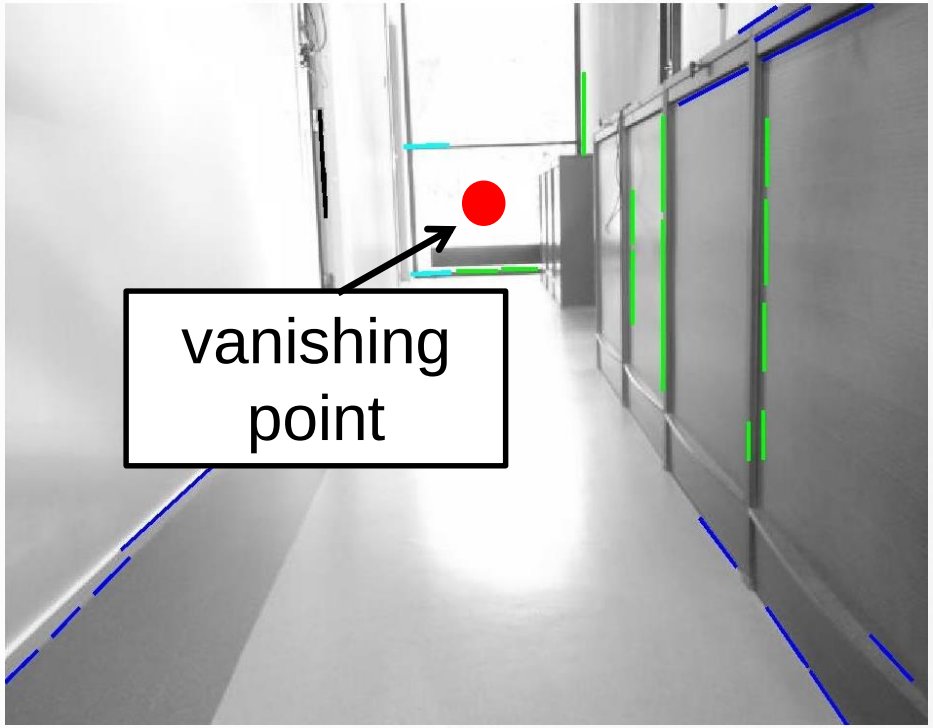
intrinsic parameters from last slide

CAMERA ROTATION FROM VANISHING POINTS

- Lines in three orthogonal directions => three vanishing points
- Locations of vanishing points (**v**) are related to camera's intrinsic parameters (matrix **K**) and rotation of the camera (matrix **R**)

$$[\mathbf{v}_x\ \mathbf{v}_y\ \mathbf{v}_z] = \mathbf{KR}$$

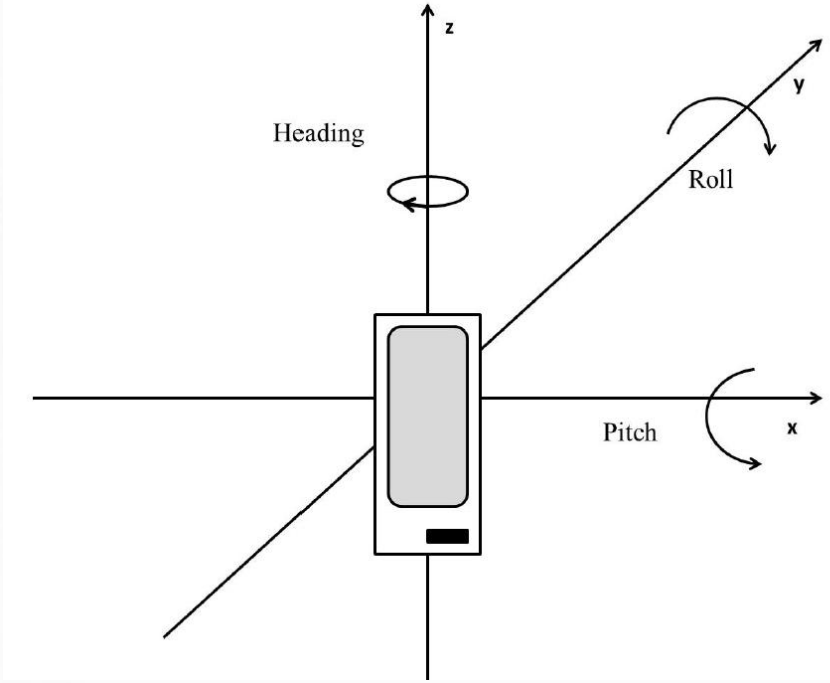
- Two vanishing points provide enough information for computing 3D rotation



Ruotsalainen 2013, PhD dissertation

CAMERA ROTATION FROM VANISHING POINTS

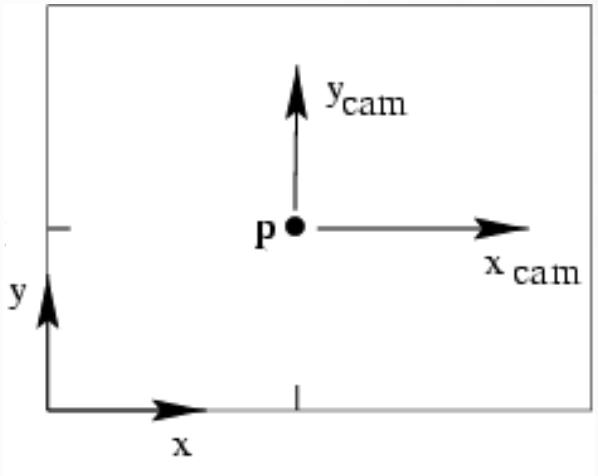
θ = heading
β = pitch
ϕ = roll



$$\mathbf{R} = \begin{bmatrix} \cos\beta\cos\theta - \sin\beta\sin\phi\sin\theta & -\sin\beta\cos\phi & \cos\beta\sin\theta + \sin\beta\sin\phi\cos\theta \\ \sin\beta\cos\theta + \cos\beta\sin\phi\sin\theta & \cos\beta\cos\phi & \sin\beta\sin\theta - \cos\beta\sin\phi\cos\theta \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{bmatrix}$$

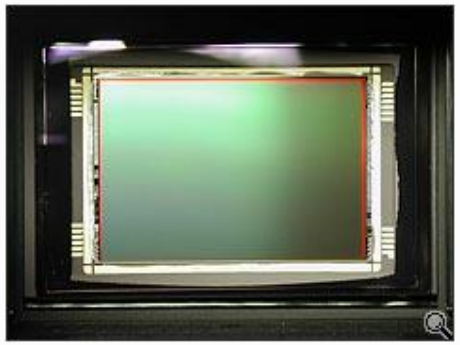
CALIBRATION MATRIX

- **Principal point (p):** point where principal axis intersects the image plane
- Normalized coordinate system: origin of the image is at the principal point
- Image coordinate system: origin is in the corner



We want the principal point to map to (p_x, p_y) instead of (0,0)

PIXEL COORDINATES



Pixel size: $\frac{1}{m_x} \times \frac{1}{m_y}$

m_x pixels per meter in horizontal direction,
m_y pixels per meter in vertical direction

$$K = \underbrace{\begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix}}_{\text{pixels/m}} \underbrace{\begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}}_{\text{m}} = \underbrace{\begin{bmatrix} \alpha_x & & \beta_x \\ & \alpha_y & \beta_y \\ & & 1 \end{bmatrix}}_{\text{pixels}}$$

Slide credit:
Kris Kitani

CALIBRATION MATRIX

- Calibration matrix K (intrinsics) converts from 3D rays in camera coordinate system to pixel coordinates

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & \beta_x \\ & \alpha_y & \beta_y \\ & & 1 \end{bmatrix}$$

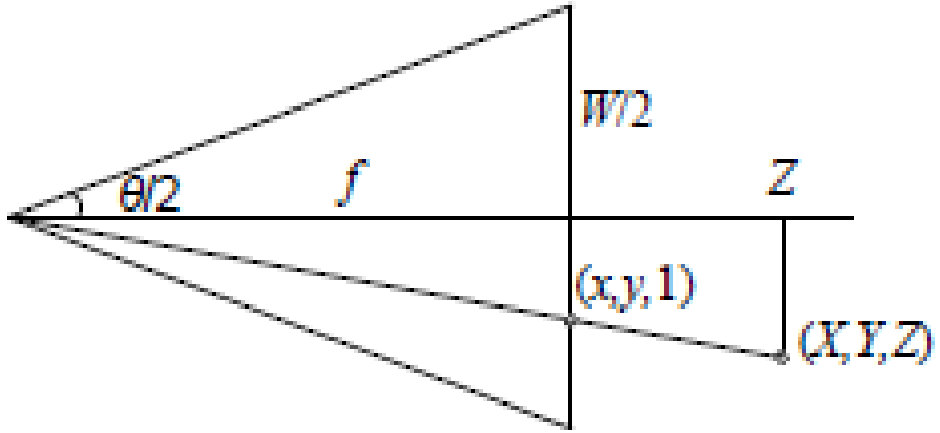
(upper triangular matrix)

s : **skew** (0 unless pixels are shaped like rhombi/parallelograms)

(β_x, β_y) : **principal point** (~center of the image)

FOCAL LENGTH

- If pixel values of an image is given with an integer, i.e. image size is [0,W) x [0,H), focal length is given in pixels
- Camera manufacturers however give the focal length usually in mm



- The relationship between focal length (f), sensor width (W) and field of view (θ) is

$$f = \frac{W}{2}(tan\frac{\theta}{2})^{-1}$$

- Setting W = 2 we obtain a unitless relationship and $f^{-1} = tan\frac{\theta}{2}$
- f expressed in pixels to the equivalent 35mm focal length => f*35/W

FOCAL LENGTH

Can think of as “zoom”, also related to *field of view* as was seen in the previous slide

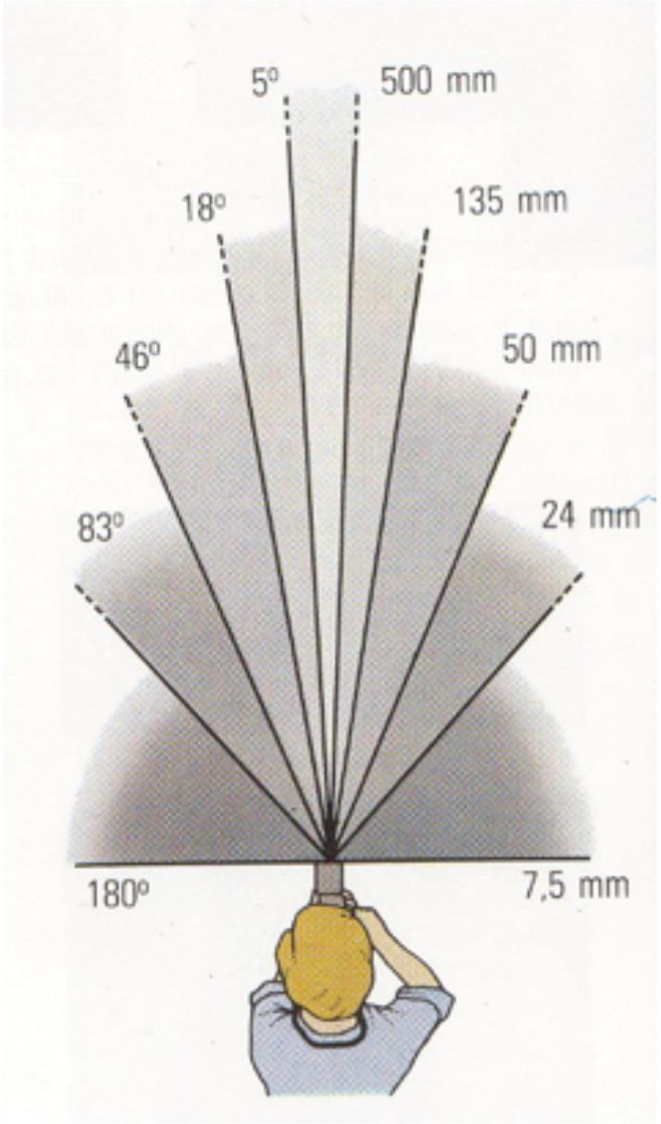
24mm

50mm

200mm

800mm

Focal length in practice



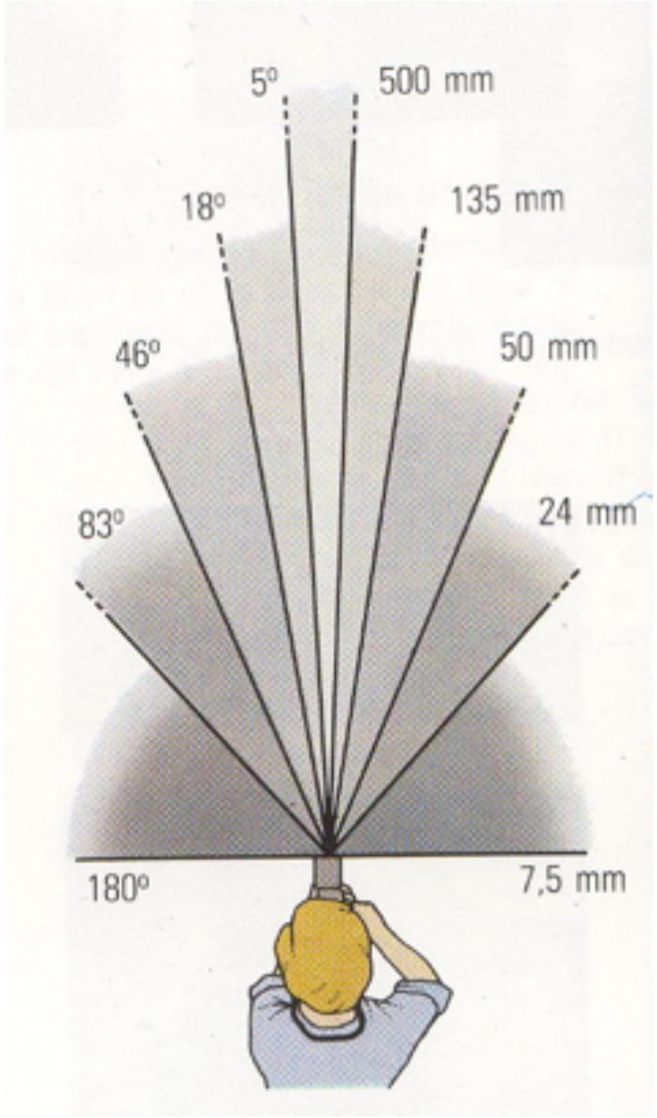
24mm

50mm

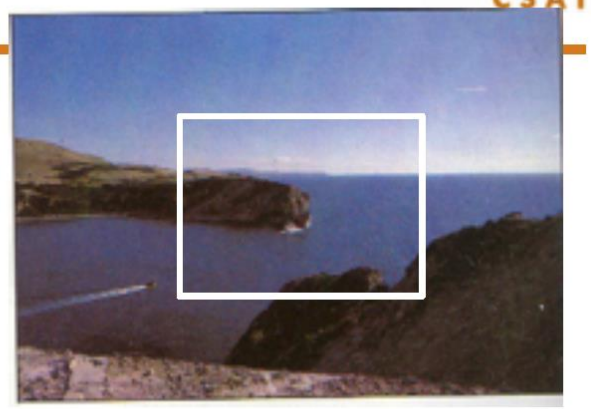
135mm

Fredo Durand

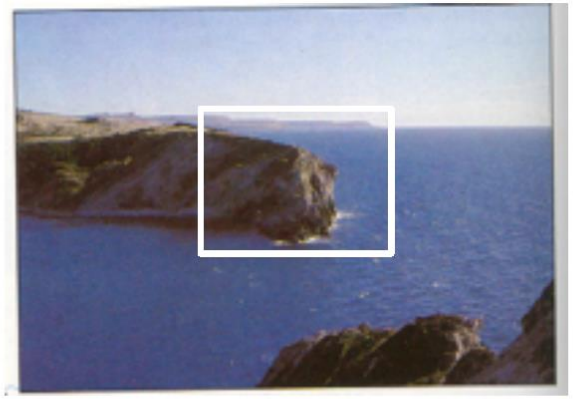
Focal length = cropping



24mm



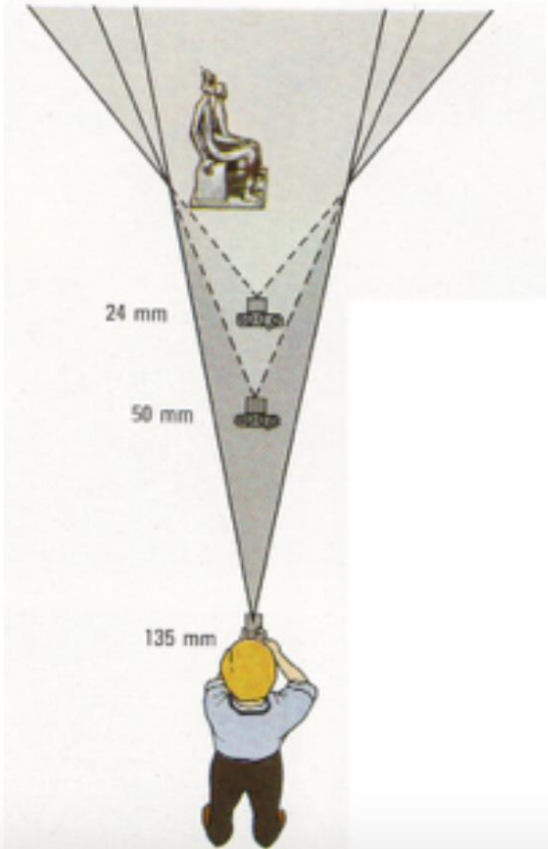
50mm



135mm

Focal length vs. viewpoint

- **Telephoto makes it easier to select background (a small change in viewpoint is a big change in background.**



Grand-angulaire 24 mm

Normal 50 mm

Longue focale 135 mm

Fredo Durand

VIEW FROM THE SAME POINT WITH DIFFERENT CAMERAS

GoPro Hero

focal length 8mm

f-number f/2.8

Sony HXR-MCI

focal length 79.5 mm

f-number f/3.2

Nokia N8

focal length 28mm

f-number f/2.8

aperture size


60˚ 10 1.2 N, 24˚ 57 18 E