COMPUTER VISION AND PHOTOGRAMMETRY

2D GEOMETRY

LAST WEEK

- Image Formation and Projection Matrix
- Camera Calibration
- Pose Estimation

CONCLUSION

- The camera matrix P, can be recovered from a set of known 3D points and their projection from, at least, 6 points.
- K, R, and T, can be recovered from P using RQ Factorization.
- Non-linear iterative methods allow for adding known constraints and include Lens distortion in the model.

CONTENT

- 2D Transformations
- Projective Transformations: Homographies H
- Computing H
- Stereo
  - Rectification
  - matching??
- Epipolar geometry: Fundamental Matrix F
- Getting H and F

2D TRANSFORMATIONS

image filtering: change ***range*** of image

$$g(x) = T(f(x))$$

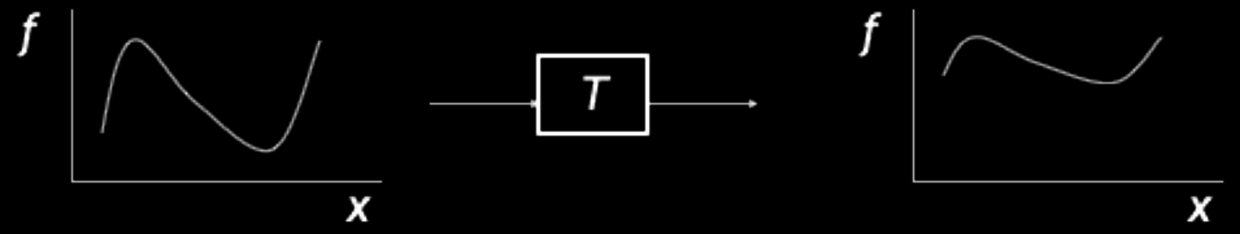
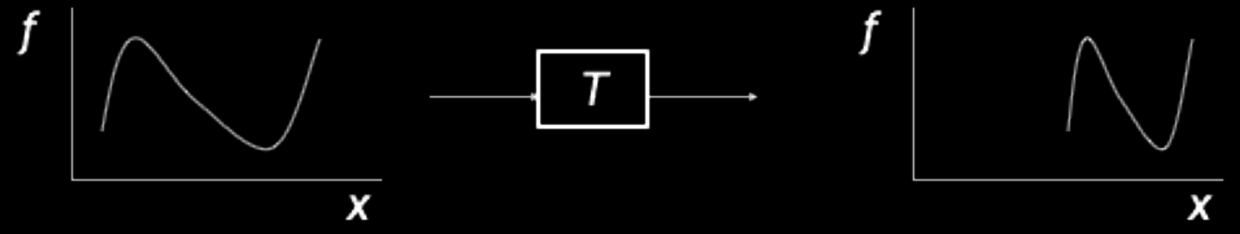


image warping: change ***domain*** of image

$$g(x) = f(T(x))$$



- Examples of parametric warps:

translation

rotation

aspect

affine

perspective

cylindrical

2D TRANSFORMATIONS

- Linear transformations are combinations of …
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror
- Properties of linear transformations:
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

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

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

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Q: How can we represent translation as a 3x3 matrix?

$$x' = x + t_x$$
$$y' = y + t_y$$

A: Using the rightmost column:

$$\mathbf{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2D TRANSFORMATIONS

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations
- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition
  - Models change of basis

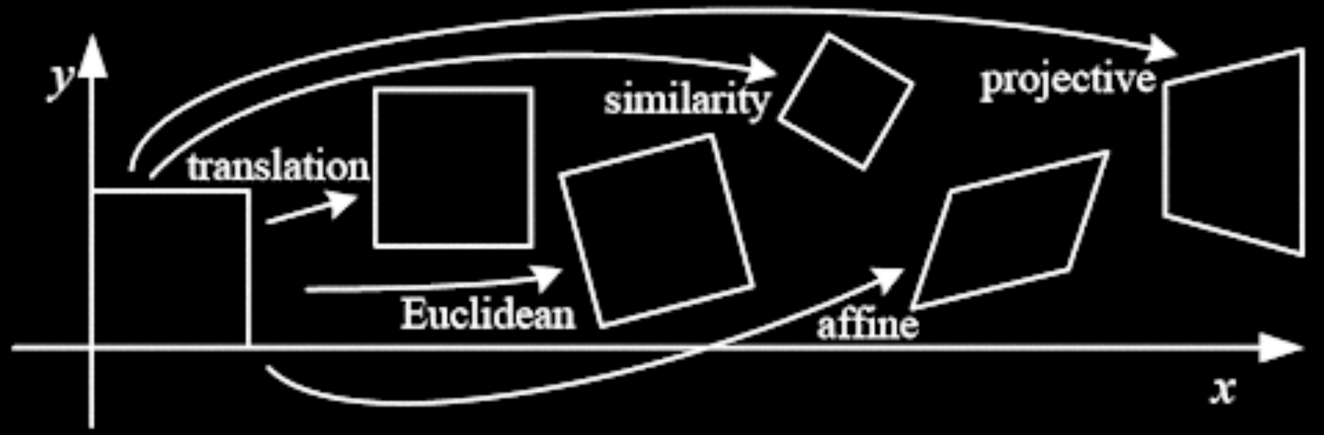
$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

2D TRANSFORMATIONS

- Projective transformations …
  - Affine transformations, and
  - Projective warps
- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition
  - Models change of basis

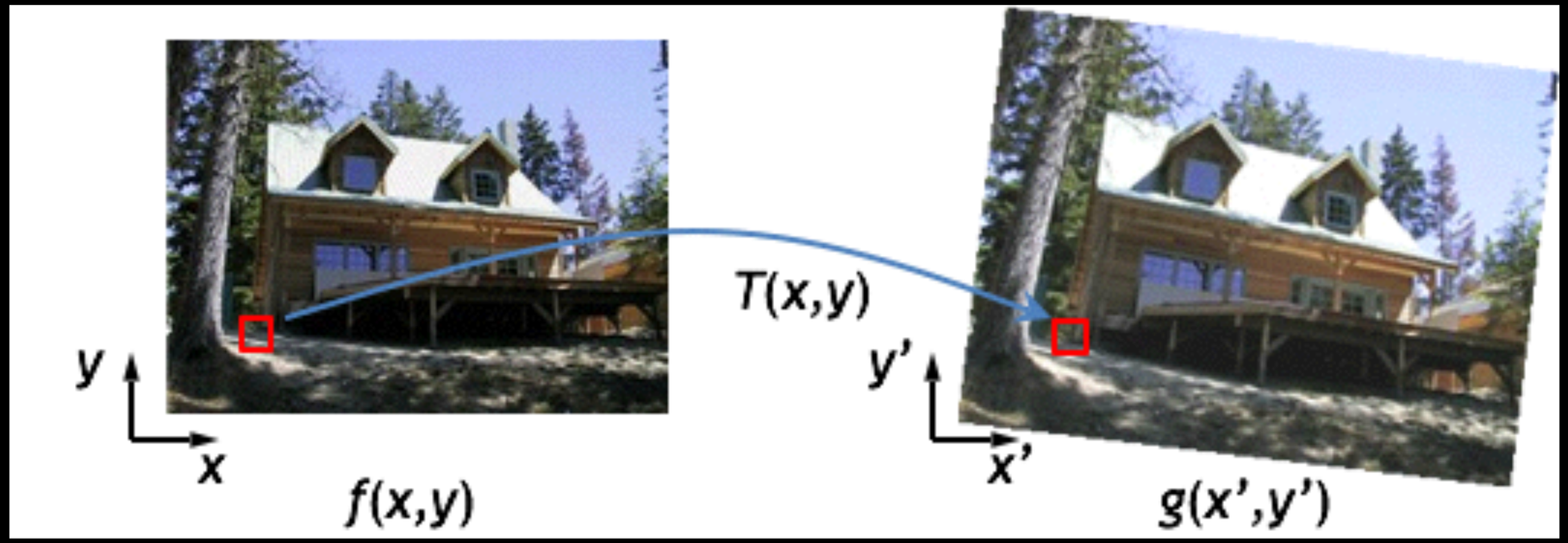
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

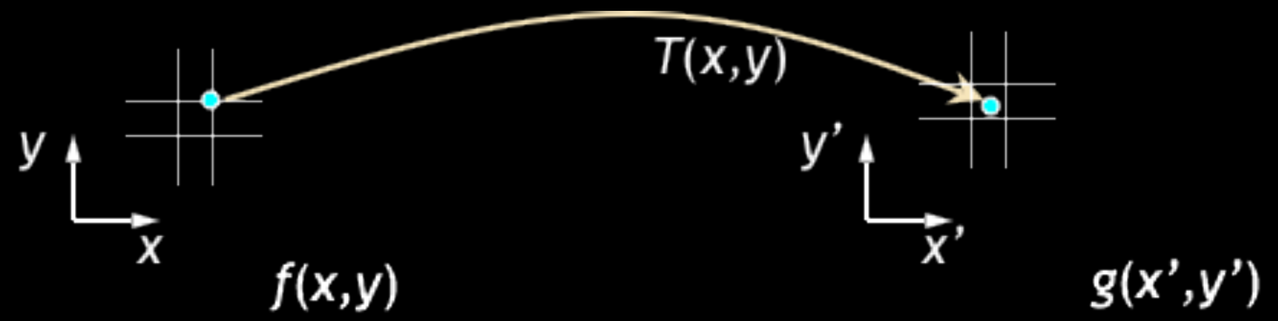
2D TRANSFORMATIONS



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

2D TRANSFORMATIONS





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

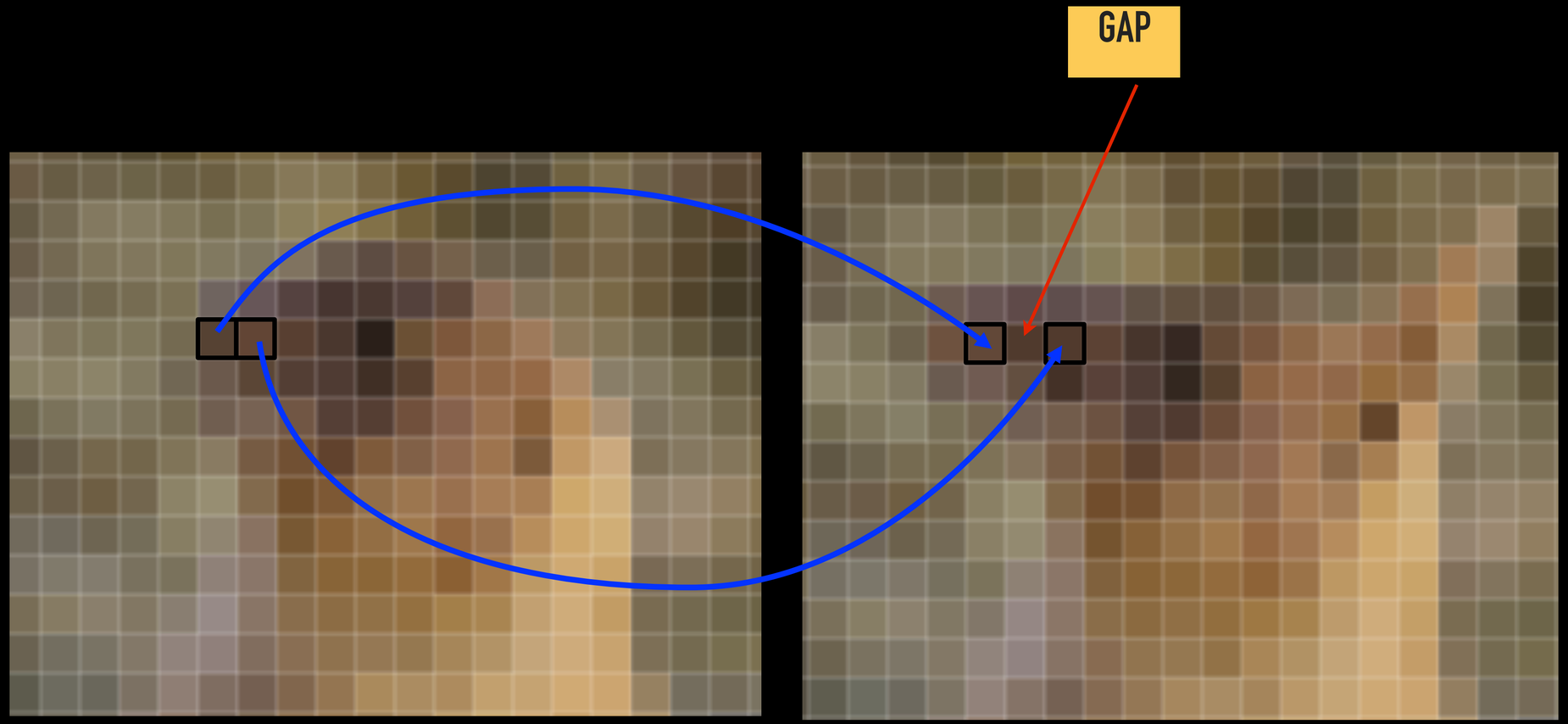
$$(x',y') = T(x,y) \text{ in the second image}$$

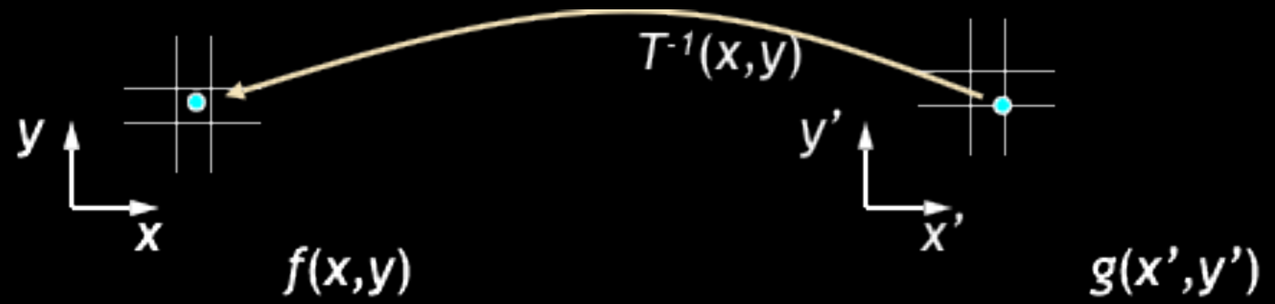
Q: what if pixel lands "between" two pixels?

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

- Known as "splatting"

2D TRANSFORMATIONS





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

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

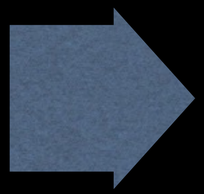
Q: what if pixel comes from "between" two pixels?

A: *Interpolate* color value from neighbors

- nearest neighbor, bilinear, Gaussian, bicubic

NAIVE SCALING

- Take all the pixels in input and transform them to their output location
  - im[y, x] => out[k*y, k*x]



USE THE INVERSE TRANSFORM!

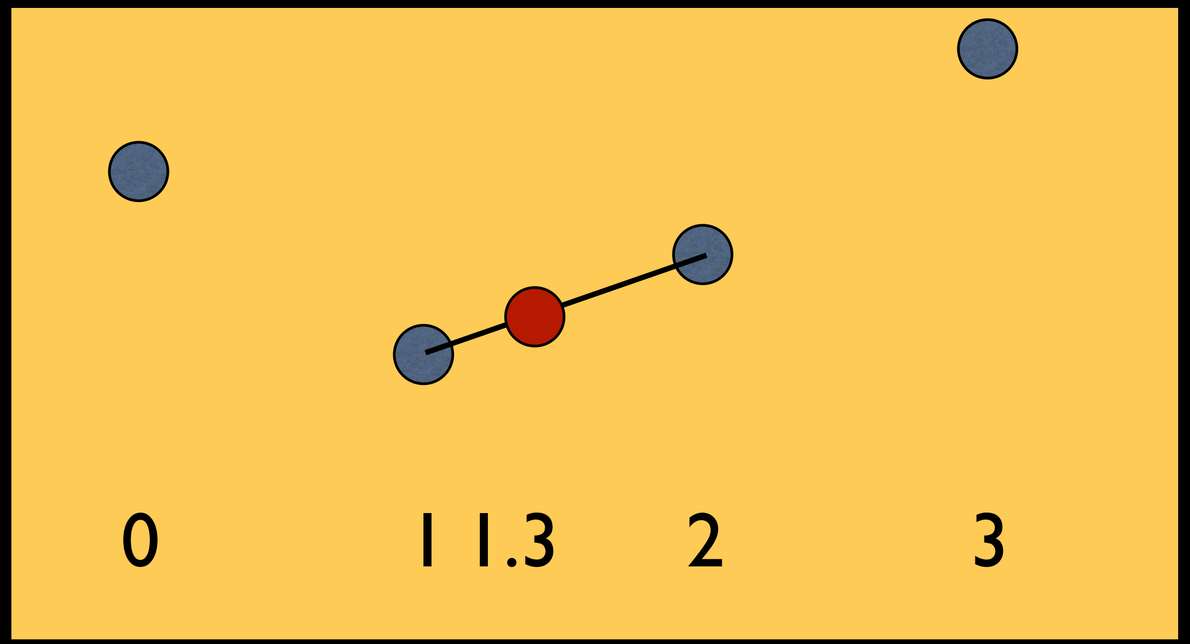
- Main loop on **output** pixels
  - out[y, x] <=im[y/k, x/k]

REMAINING PROBLEM

- A little too “blocky”
- Because we round to the nearest integer pixel coordinates
  - called nearest neighbor reconstruction

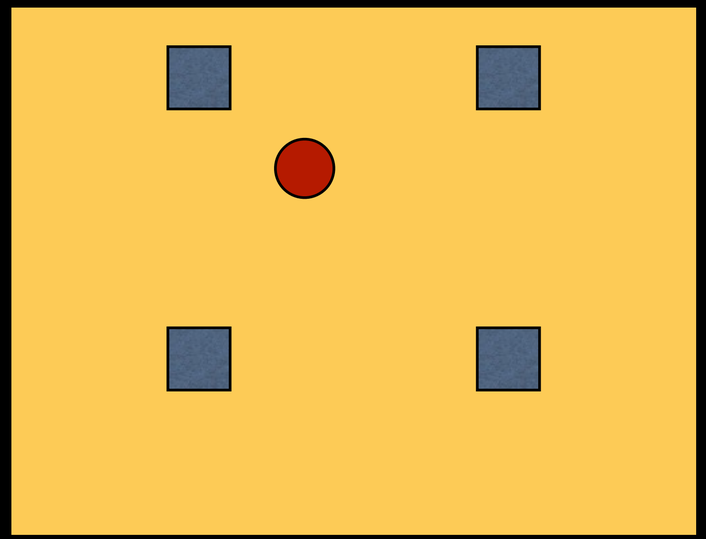
LINEAR RECONSTRUCTION

- im is now a 1D array along x
- reconstruct im[1.3]
- =0.7*im[1]+0.3*im[2]

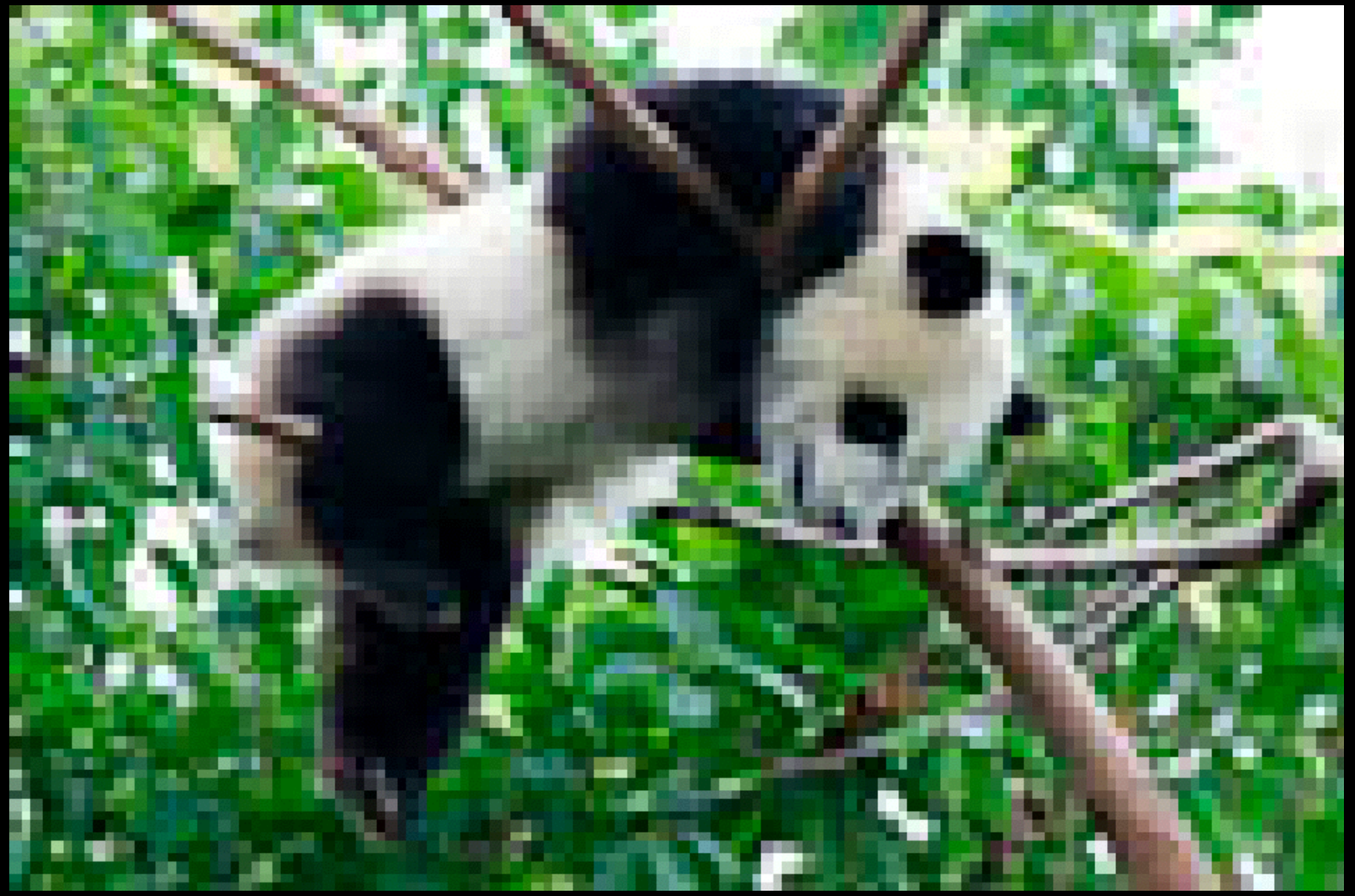


BILINEAR RECONSTRUCTION

- Take 4 nearest neighbors
- Weight according to x & y fractional coordinates
- Can be done using two 1D ;linear reconstructions along x then y (or y then x)



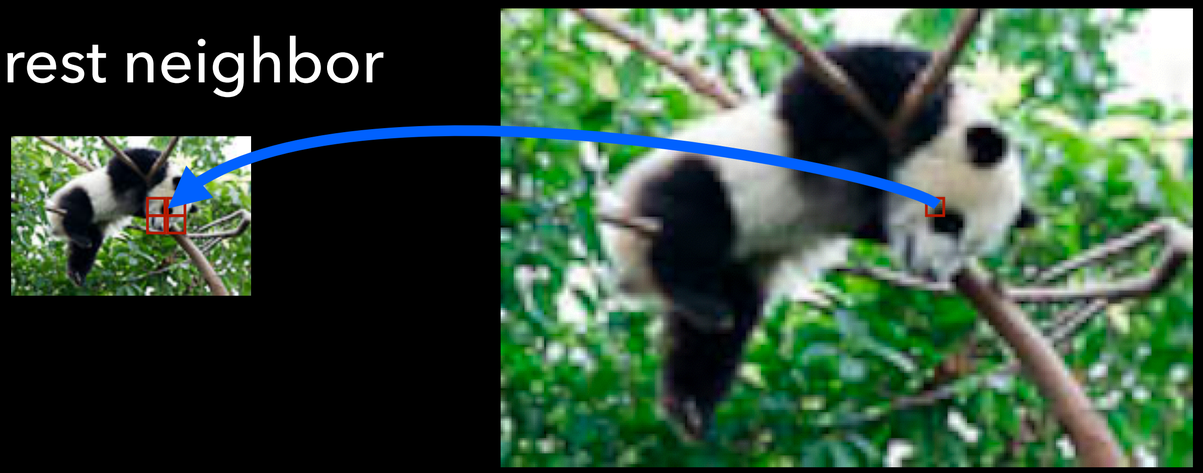
RECALL NEAREST NEIGHBOR



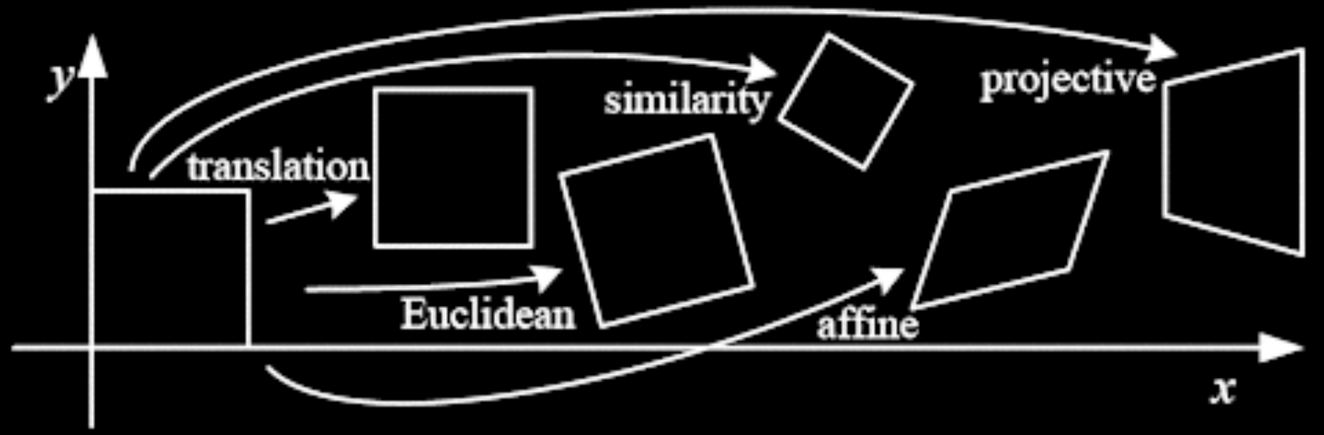
BILINEAR

TAKE HOME MESSAGES

- Main loop over OUTPUT pixels
  - Makes sure you cover all of them
- Use inverse transform
- Reconstruction makes a difference
  - Linear much better than nearest neighbor



2D TRANSFORMATIONS



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

HOMOGRAPHIES

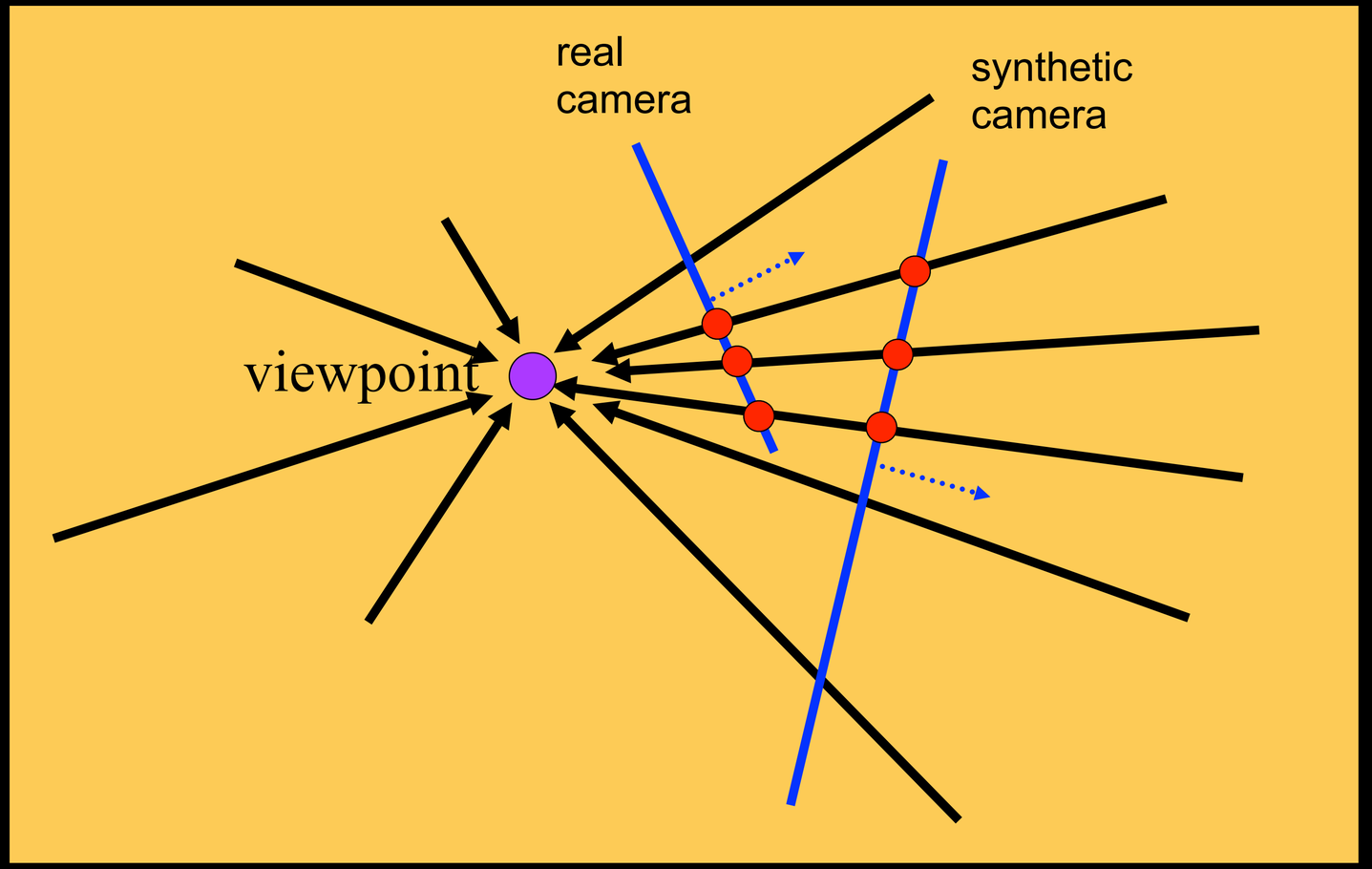
MOTIVATION

- It’s all a 3x3 matrix

MOTIVATION

A PENCIL OF RAYS CONTAINS ALL VIEWS

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



OTHER INTERPRETATION

- Depth does not matter
- We can pretend that each pixel is at a convenient depth

Three convenient depth distributions:

spherical

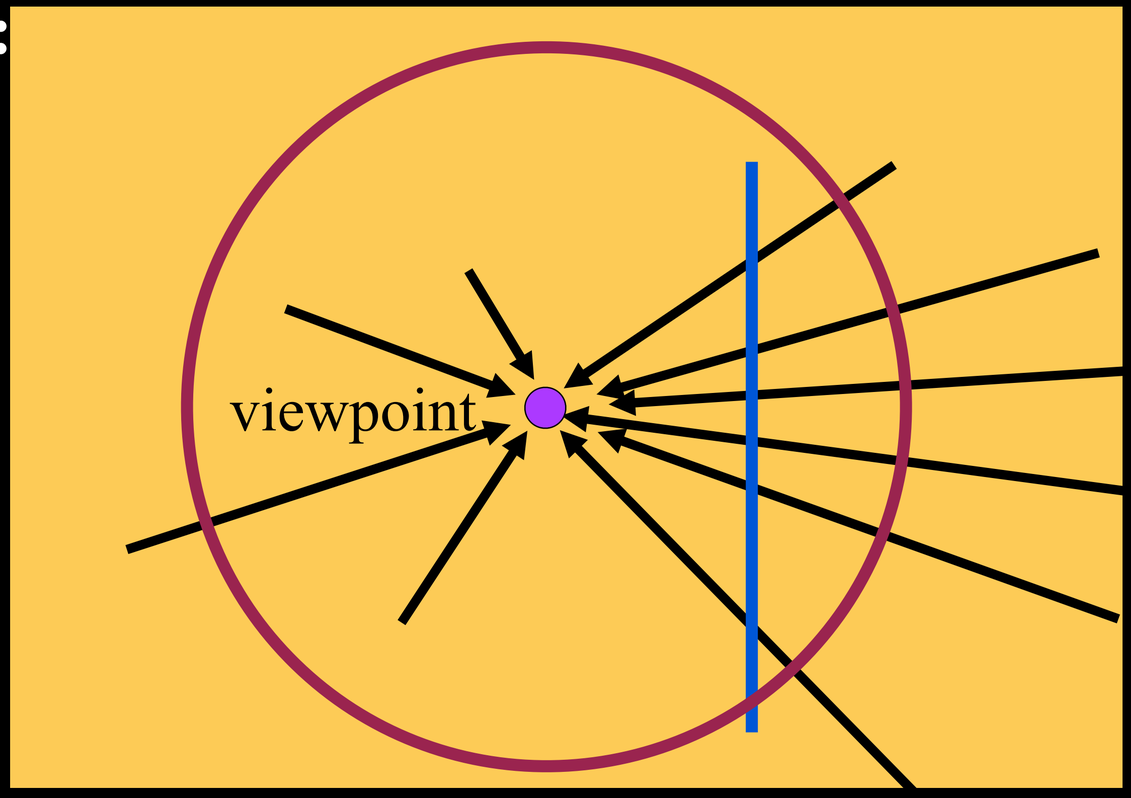
planar

cylindrical

We focus on planar

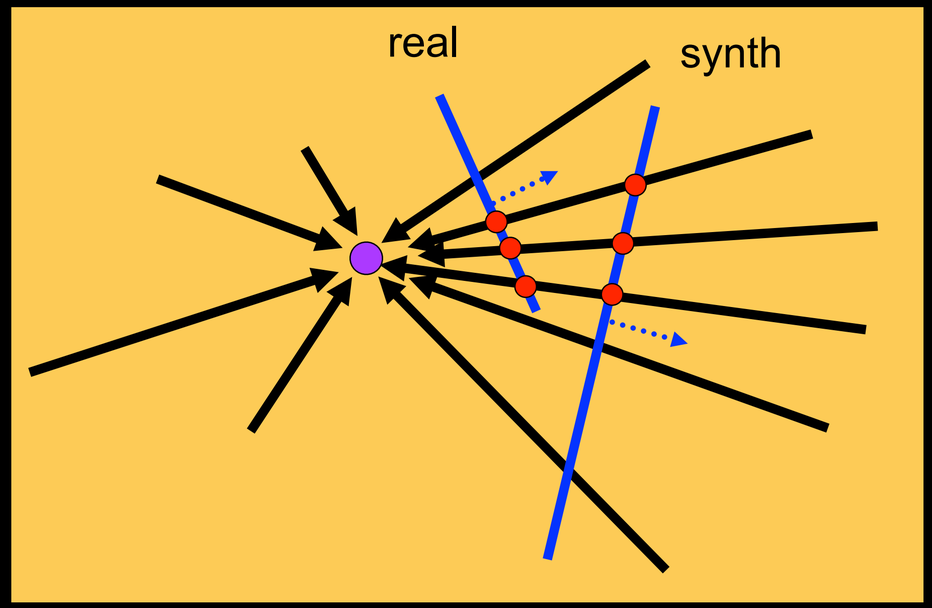
it makes life more linear

Still useful for spherical panos



RECAP

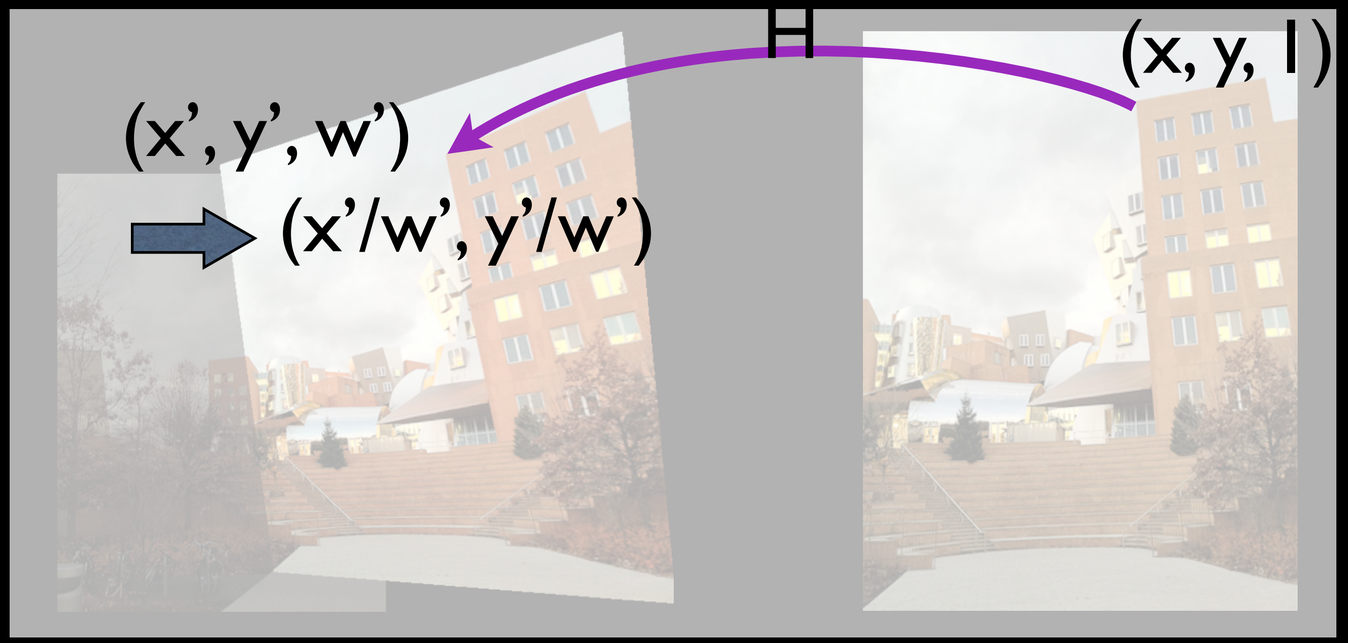
- When we only rotate the camera (around nodal point) depth does not matter
- It only performs a 2D warp
  - one-to-one mapping of the 2D plane
  - plus of course reveals stuff that was outside the field of view



- Now we just need to figure out this mapping

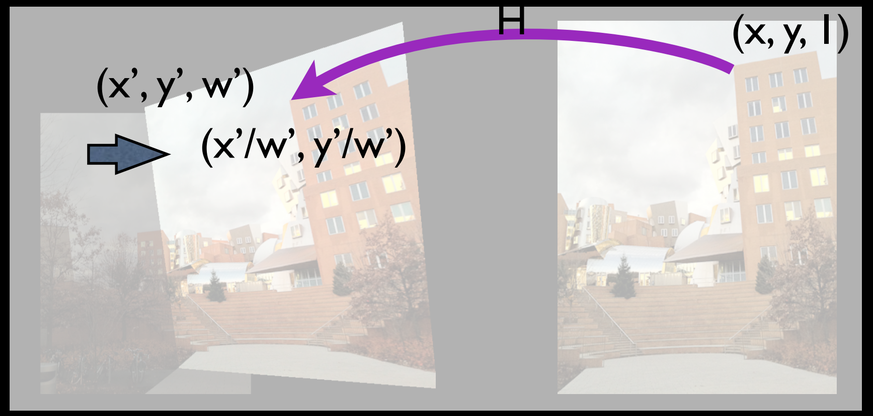
2D HOMOGENOUS VERSION

- 2D points represented as homogenous coordinates (x, y, 1)
- Mapped into the other image by a 3x3 matrix H, into homogenous coordinates (x′, y′, w′)
- get Euclidean coordinates by dividing by w′



3D VS. HOMOGENOUS: THE SAME

- 3 coordinates because viewed as plane in 3D
- 3x3 matrix because rotation
- Divide because perspective
- weird extra coordinate
- 3x3 homography matrix
- divide because it’s the rule



WRAPPING IT UP: HOMOGRAPHY

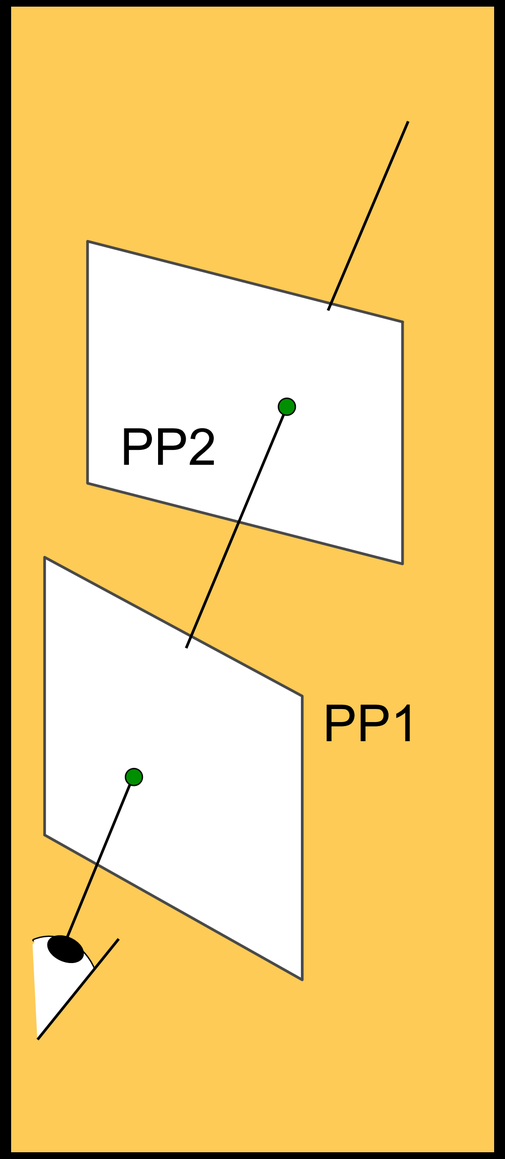
- Projective mapping between any two planes
- represented as 3x3 matrix in homogenous coordinates
  - corresponds to the 3D rotation

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

To apply a homography **H**

Compute **p'** = **Hp** (regular matrix multiply)

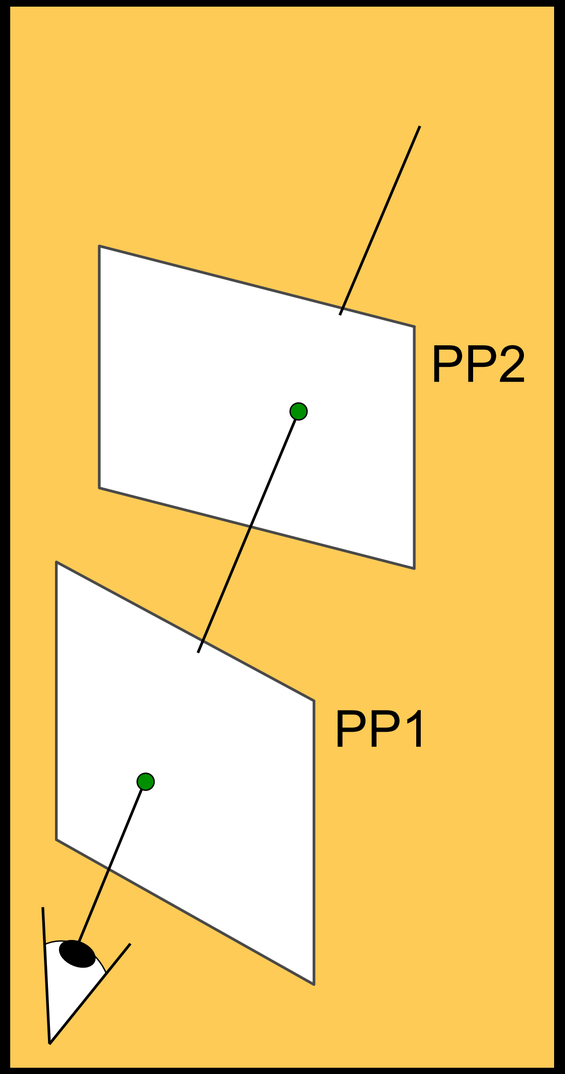
Convert **p'** from homogeneous to image coordinates (divide by w)



WRAPPING IT UP: HOMOGRAPHY

- rectangles map to arbitrary quadrilateral
- parallel lines aren’t parallel anymore
- but straight lines remain straight

- same as: project, rotate, reproject

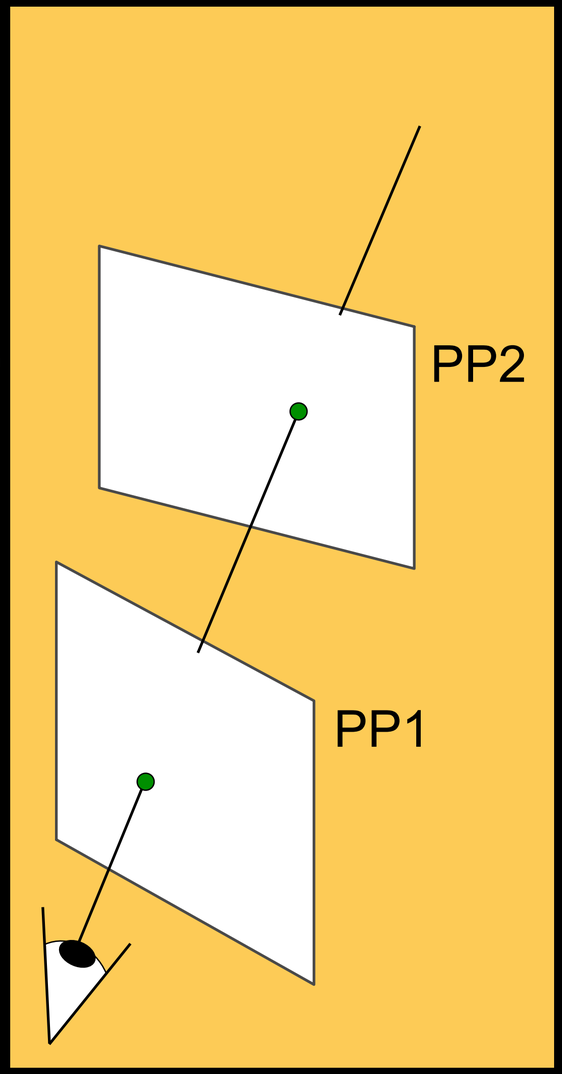


RECAP

- Reprojection = homography
- 3x3 matrix in homogeneous coordinate
  - (the matrix can be constrained to be a rotation)

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

$\mathbf{p}'$ $\mathbf{H}$ $\mathbf{p}$



SCALE

- Homographies are defined up to a scale
- H and kH represent the same 2D transformation
- because (w'x', w'y', w') and (kw'x', kw'y', kw') represent the same point

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

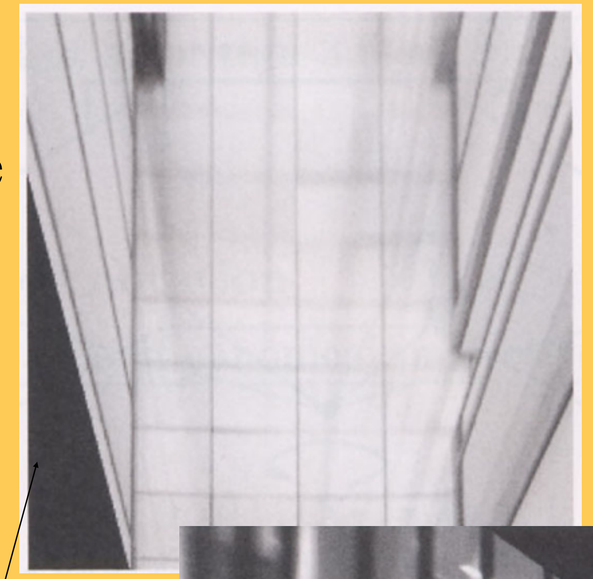
CORRECT PERSPECTIVE (PERSPECTIVE CROP)

- Recall from last time. Its a homography

TEXT

IMAGE WARPING WITH HOMOGRAPHIES

homography
so that image



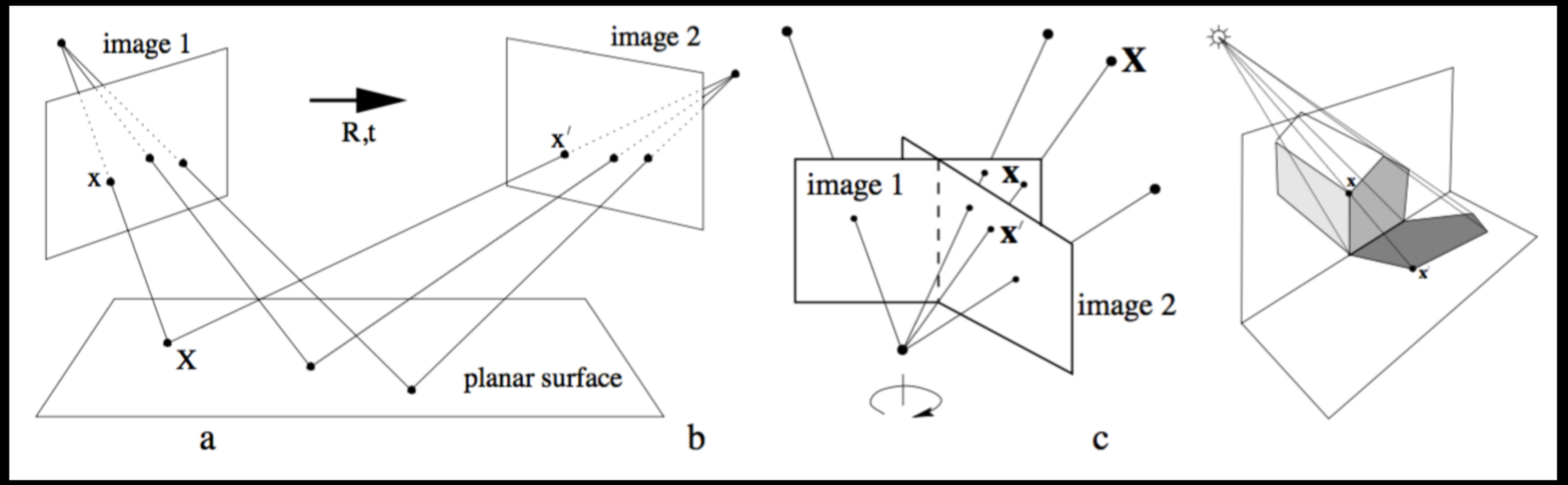
homography so that image is
parallel to right wall

ima

black area
where no pixel
maps to

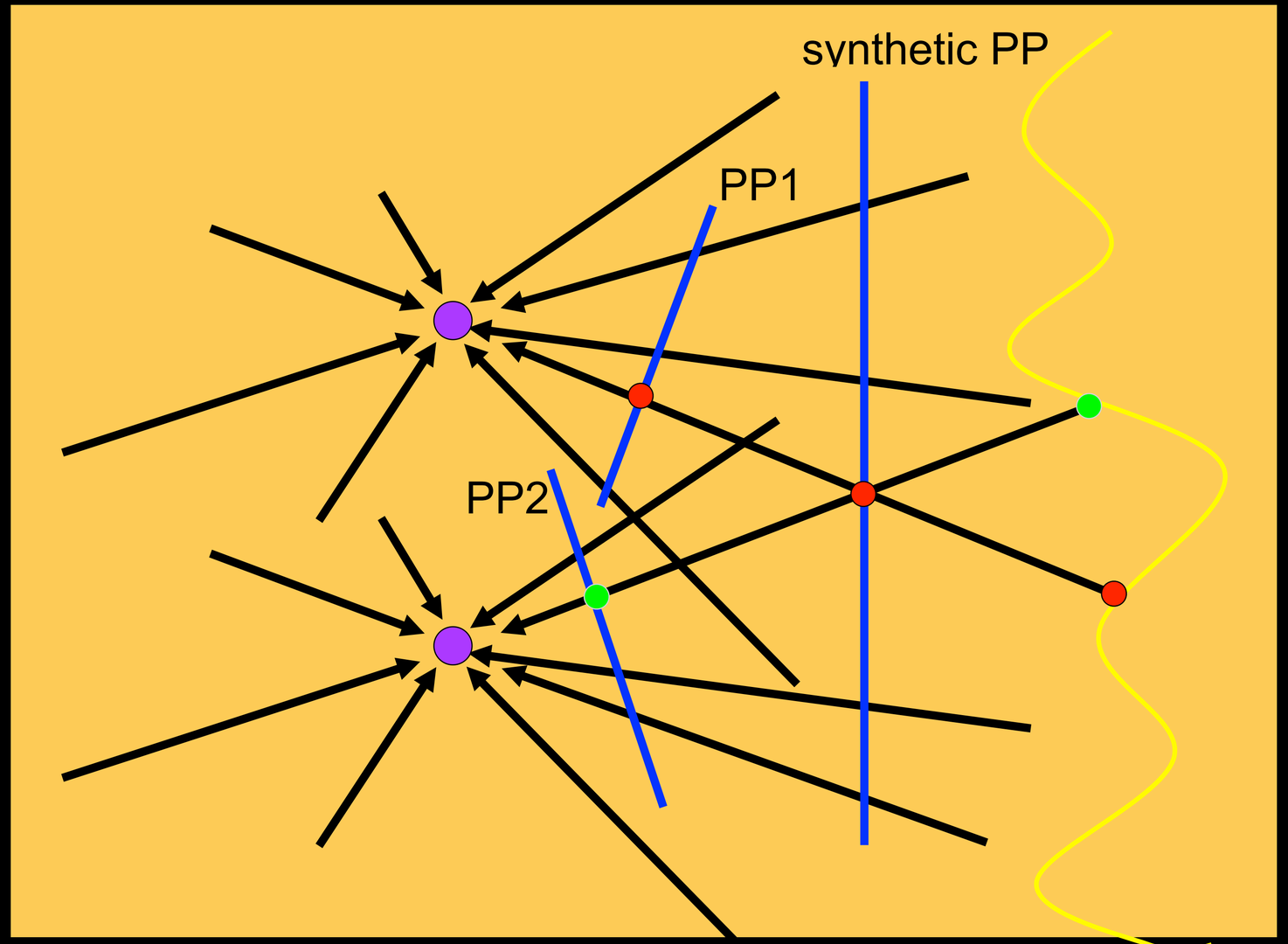
EXAMPLES OF HOMOGRAPHIES

- Rotation and Translation in 3D
- Rotation on camera center
- Shadows from a point light-source



CHANGING CAMERA CENTER

- Does it still work?
- Only for flat surfaces
- A more general relationship is the Fundamental matrix F
- Next week



WHEN WE WILL USE HOMOGRAPHIES?

- Rectification for stereo images
- Camera Calibration Lab 1
- Panorama Stitching in Computational Photography

CALCULATING AN HOMOGRAPHY

GOAL

- Given correspondences
- Find homography matrix H that maps the p_i to p_i'

HOMOGRAPHY EQUATION

- We are given pairs of corresponding points
  - x, y, x′, y′ are known
- Unknowns: matrix coefficients and w′
- (the order in Y and X is only for convenience when

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} y \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} y'w' \\ x'w' \\ w' \end{pmatrix}$$

HOMOGRAPHY EQUATION

- We are given pairs of corresponding points
  - x, y, x′, y′ are known
- Unknowns: matrix coefficients and w′
  - but w′ is easy to get:

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} y \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} y'w' \\ x'w' \\ w' \end{pmatrix}$$

$$w' = gy + hx + i$$

TEXT

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} y \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} y'w' \\ x'w' \\ w' \end{pmatrix}$$

$$w' = gy + hx + i$$

- For a pair of points (x, y)->(x’,y’) we have

$$ay + bx + c = y'(gy + hx + i)$$

$$dy + ex + f = x'(gy + hx + i)$$

TEXT

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} y \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} y'w' \\ x'w' \\ w' \end{pmatrix}$$

$$w' = gy + hx + i$$

- For a pair of points (x, y)->(x',y') we have

$$ay + bx + c = y'(gy + hx + i)$$

$$dy + ex + f = x'(gy + hx + i)$$

- Unknowns: a, b, c, d, e, f, g, h, i
  - Linear!

- Each correspondence pair gives us two equations

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} y \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} y'w' \\ x'w' \\ w' \end{pmatrix}$$

$$w' = gy + hx + i$$

$$ay + bx + c = y'(gy + hx + i)$$

$$dy + ex + f = x'(gy + hx + i)$$

- How many unknowns?
  - 9
  - but H is defined up to scale. Four pairs are enough!

FORMING THE LINEAR SYSTEM

- We have 4x2 linear equations in our 8 unknowns
- Represent as a matrix systemAx=B:
- Now we need to fill matrix A and vector B

$$\left(\quad A \quad \right) \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{pmatrix} = B$$

FORMING THE MATRIX

$$ay + bx + c = y'(gy + hx + i)$$

$$dy + ex + f = x'(gy + hx + i)$$

$$\overbrace{\begin{pmatrix} a & b & c & d & e & f & g & h & i \end{pmatrix}}^{} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{pmatrix} = \begin{matrix} \cdot \\ \cdot \\ \cdot \\ \\ \\ \cdot \\ \cdot \\ \cdot \end{matrix}$$

FORMING THE MATRIX

$$ay + bx + c = y'(gy + hx + i)$$

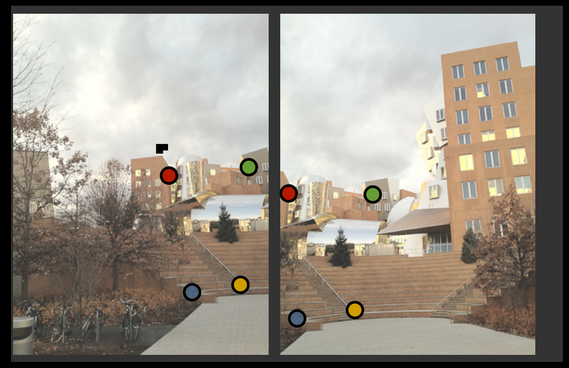
$$dy + ex + f = x'(gy + hx + i)$$

$$\begin{pmatrix} \overbrace{a \quad b \quad c \quad d \quad e \quad f \quad g \quad h \quad i} \\ \cdots\cdots \\ \text{y} \quad \text{x} \quad \text{1} \quad \text{0} \quad \text{0} \quad \text{0} \quad \text{-yy'} \quad \text{-xy'} \quad \text{-y'} \\ \cdots\cdots \\ \cdots\cdots \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{pmatrix} = \begin{matrix} \cdot \\ \cdot \\ \cdot \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{matrix}$$

I'll let you do the x case for pset 6

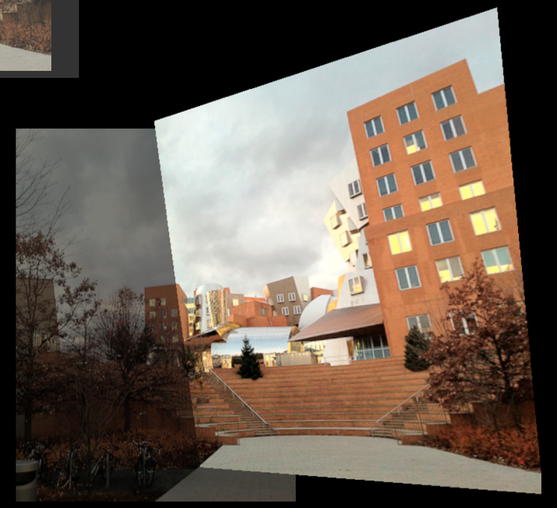
RECAP

- We have four pairs of points



- Looking for homography H

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} y \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} y'w' \\ x'w' \\ w' \end{pmatrix}$$



- Formed a big 8x9 linear system Ax=0
  - where x is the 9 homography coefficients

$$\begin{pmatrix} \cdots\cdots \\ y \;\; x \;\; 1 \;\; 0 \;\; 0 \;\; 0 \text{-}yy' \text{-}xy' \text{-}y' \\ \cdots\cdots \\ \cdots\cdots \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ f \\ g \\ h \\ i \end{pmatrix} = \begin{matrix} . \\ . \\ . \\ 0 \\ . \\ . \\ . \end{matrix}$$

SOLVE FOR SCALE INVARIANCE

- We know that there exists a full family of solutions kH, for any non-zero k
- i.e., if (a, b, c, d, e, f, g, h, i) is a solution,
  so is (ka, kb, kc, kd, ke, kf, kg, kh, ki)
- Adding more correspondences won’t help

DIRTY SOLUTION

- Hope that *i* is not 0
- Set it arbitrarily to 1
  - either create an 8x8 matrix
  - or add a last row that says i should be 1
- In practice i is rarely zero
  - But it’s still dirty
- You can use this solution for pset 6

CLEANER SOLUTION

- Use SVD
- The singular vector with singular value 0 is a solution
- See your favorite linear algebra textbook

DIRTY-CLEAN: THREE VERSIONS

- Dirty with i=1
  - 9x9:
    - RHS is zero almost everywhere, except last row
    - last row just says i=1
  - 8x8:
    - substitute i=1 from beginning
    - RHS is usually non-zero
- Clean with SVD: 8x9
  - Ax=0

SUMMARY

LAB 1
CAMERA CALIBRATION

TASK 1 : POSE ESTIMATION

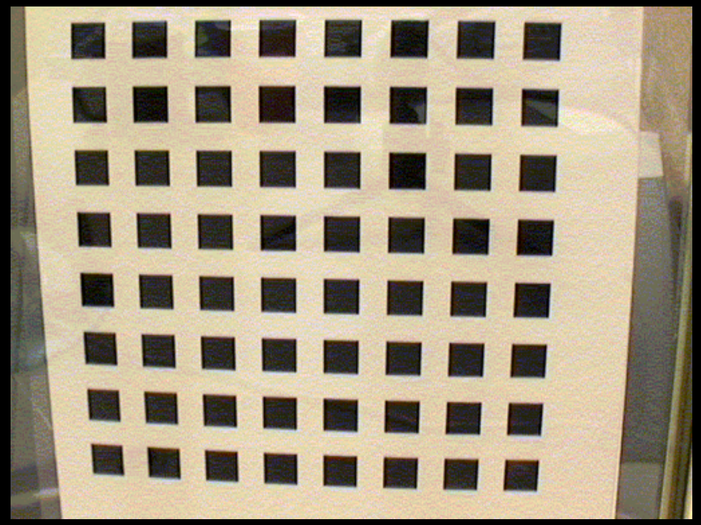
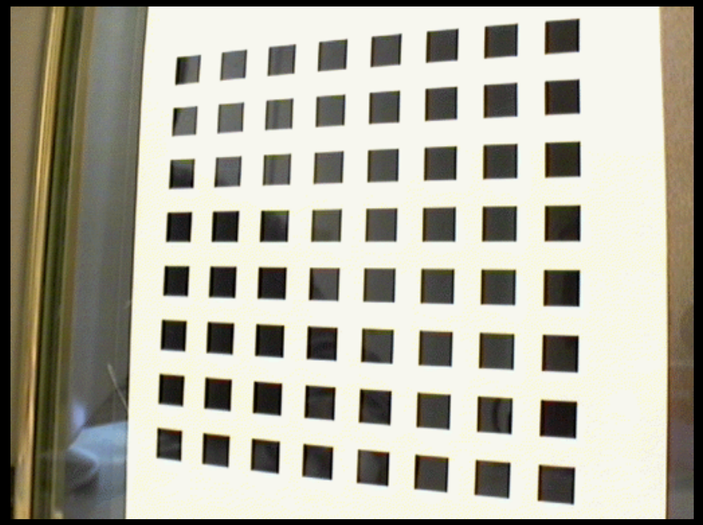
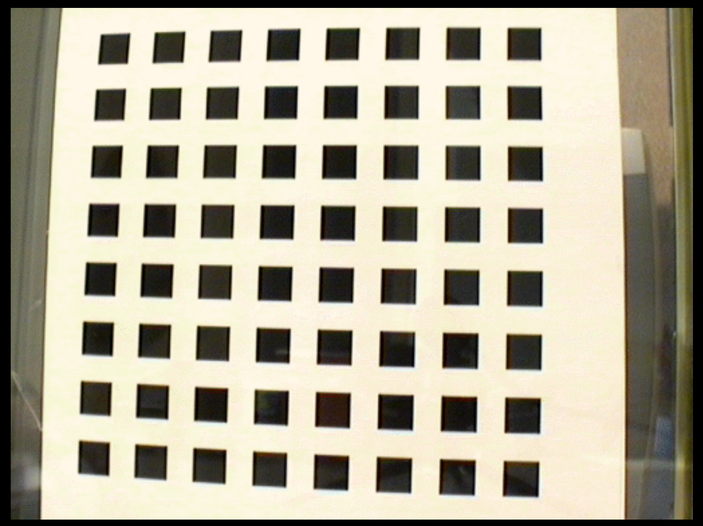
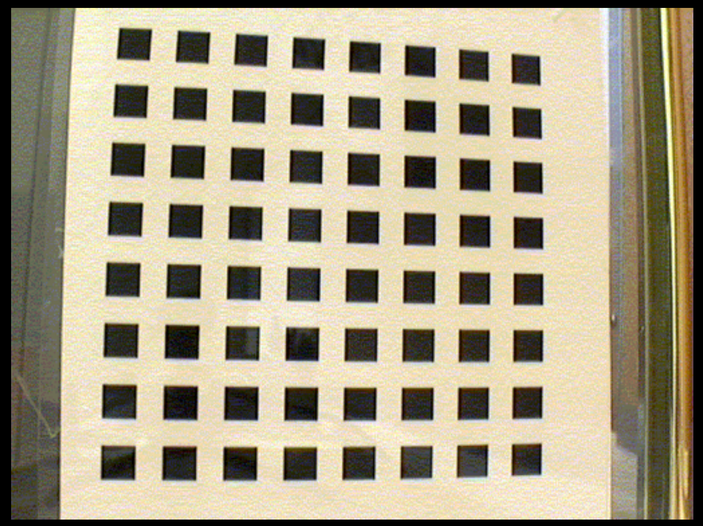
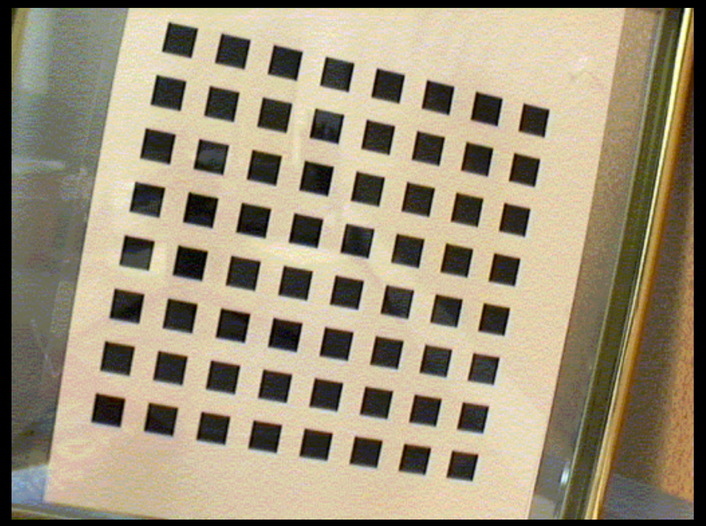
TASK 2 : DECOMPOSE P INTO K,R,T, AND C

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

5 6

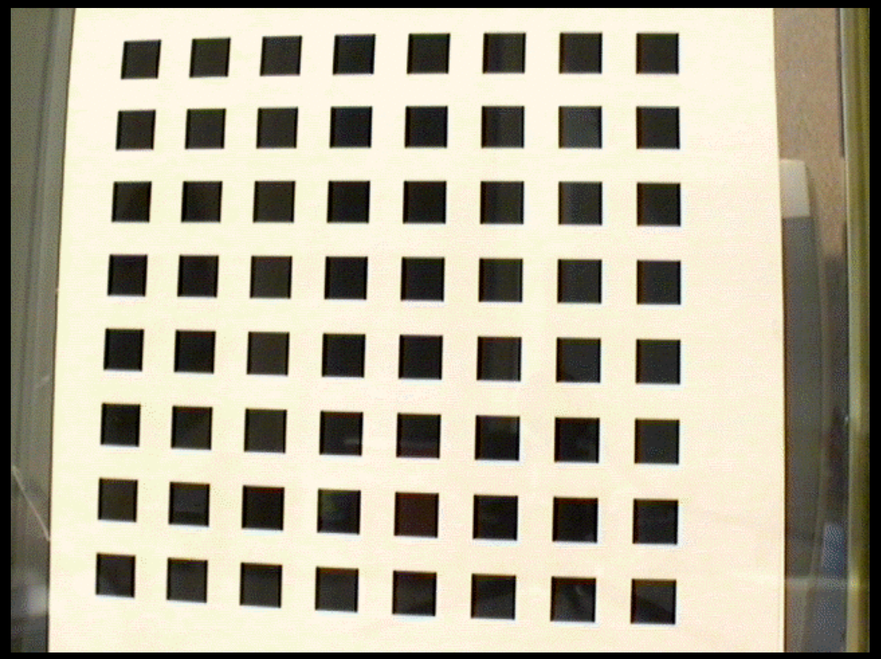
$$w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & s & u_0\\0 & \beta & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

TASK 3 : LENS DISTORTION PROJECTION

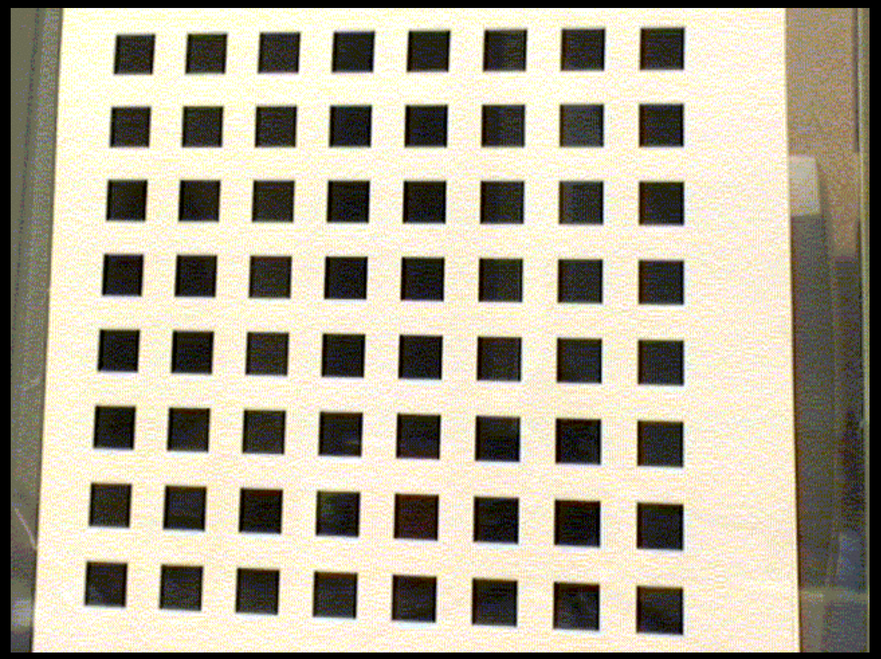






TASK 4 : LENS DISTORTION – CORRECTION

- captured



- undistorted



TEXT

- Compute P from 3D-2D correspondences
- Extract K,R,C from P
- Compute H from 4 corresponding points
- Compute lens distortion parameters

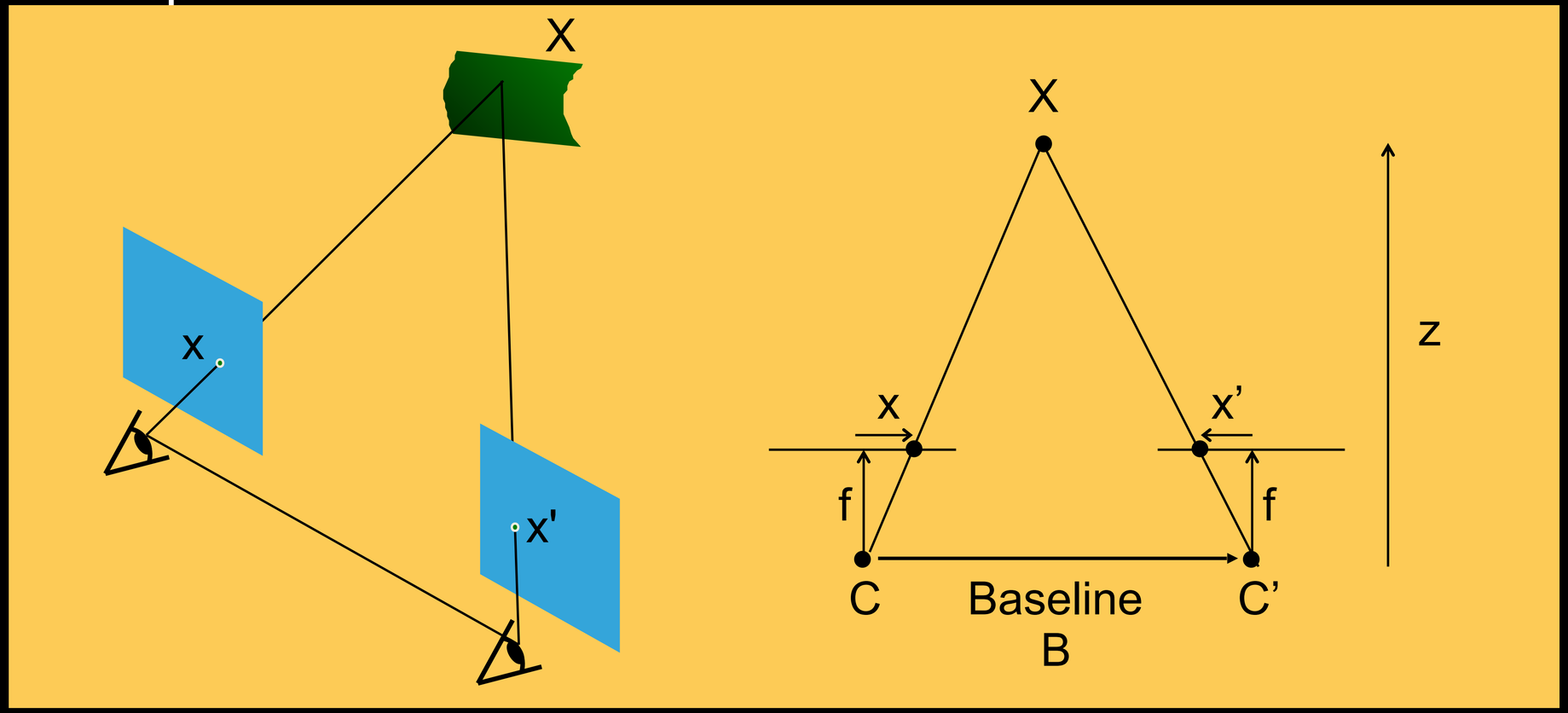
EPIPOLAR GEOMETRY

EPIPOLAR GEOMETRY

- Relates cameras from two positions

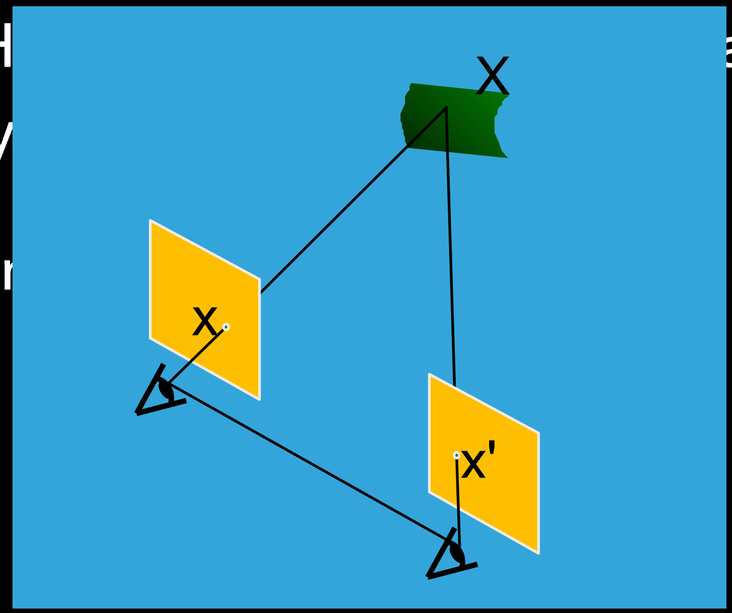
DEPTH FROM STEREO

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



DEPTH FROM STEREO

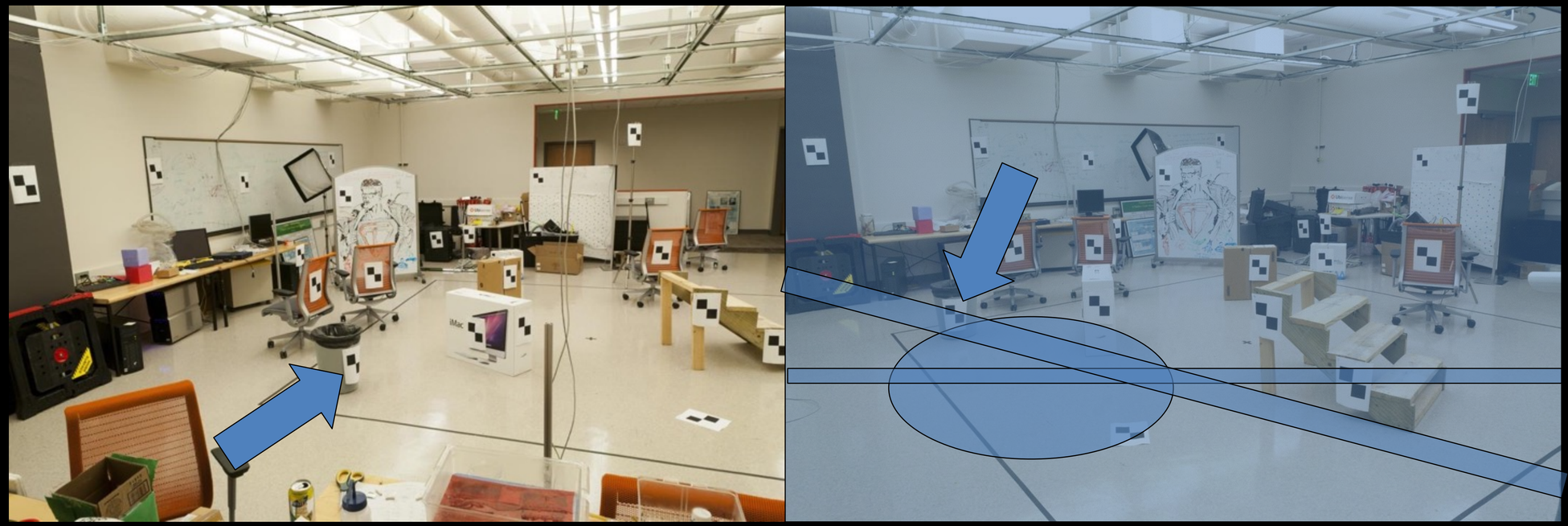
- Goal: recover depth by finding image coordinate x′ that corresponds to x
- Sub-Problems
  - Calibration: H ation of the cameras (if not already
  - Correspondenc r the matching point x′?



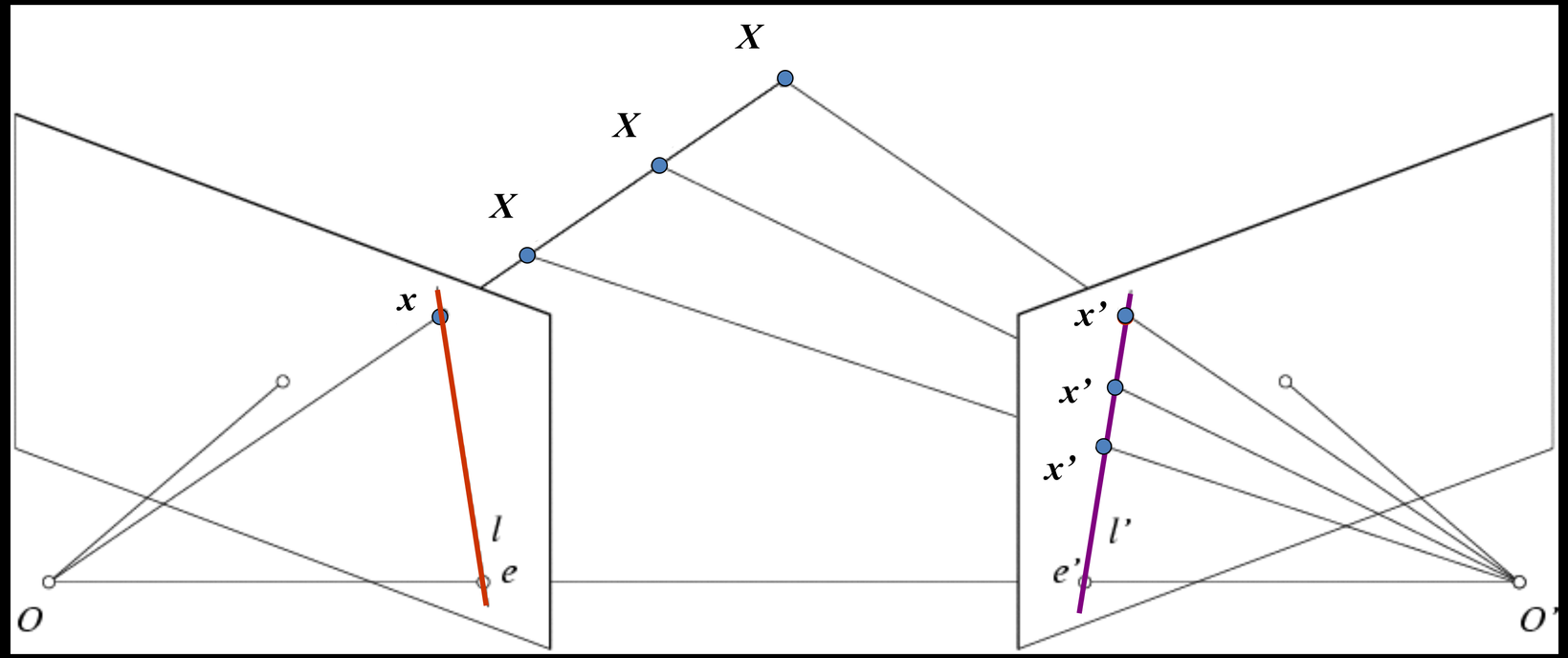
CORRESPONDENCE PROBLEM

- We have two images taken from cameras with different intrinsic and extrinsic parameters
- How do we match a point in the first image to a point in the second? How can we constrain our search?

WHERE DO WE NEED TO SEARCH?



KEY IDEA: EPIPOLAR CONSTRAINT

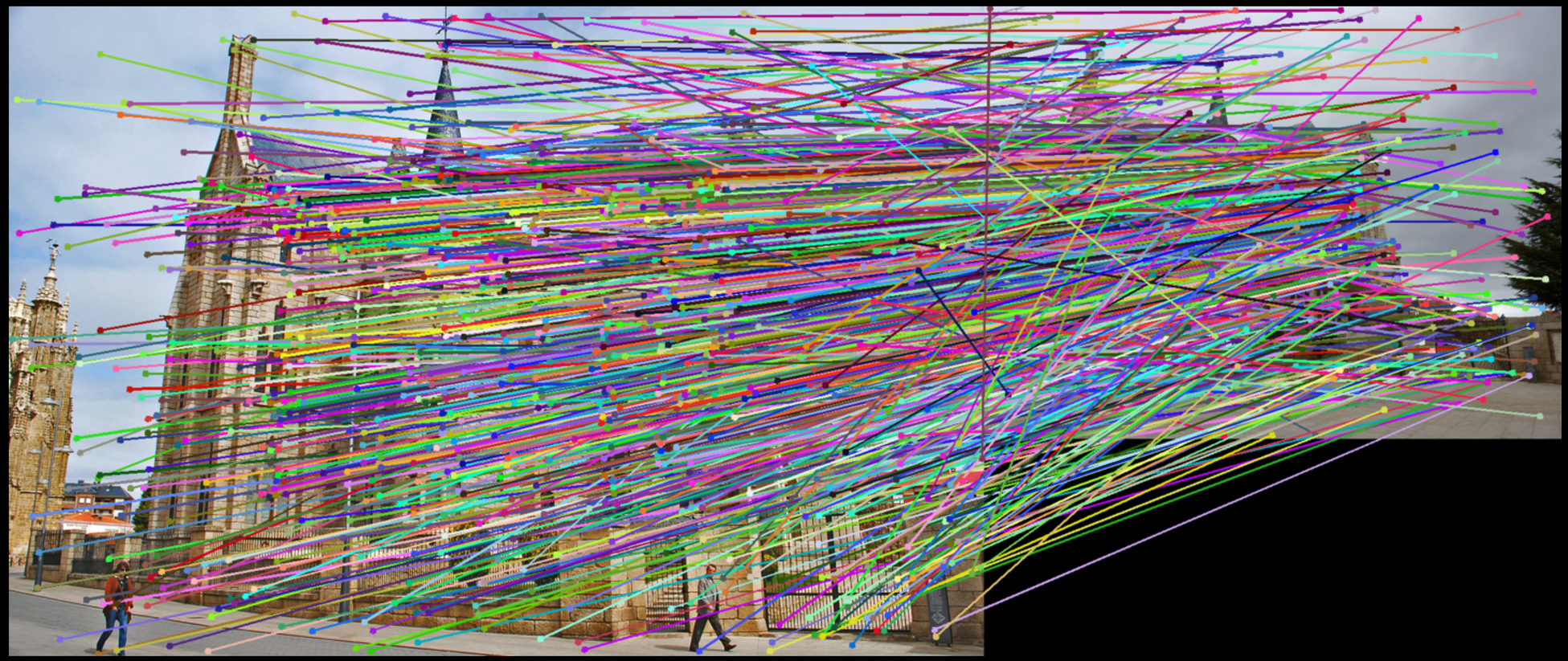


Potential matches for x have to lie on the corresponding line l′.

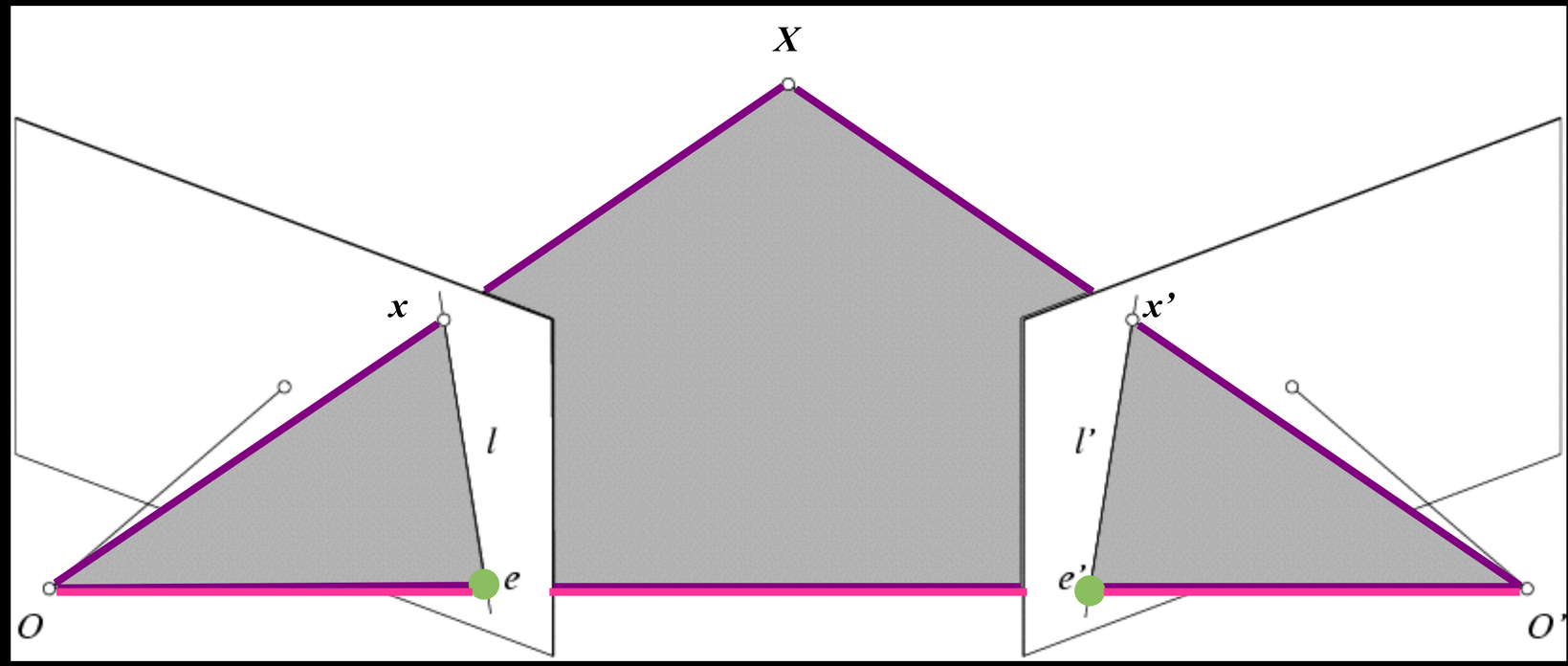
Potential matches for x′ have to lie on the corresponding line l.

WOULDN'T IT BE NICE TO KNOW WHERE MATCHES CAN LIVE? TO CONSTRAIN OUR 2D SEARCH TO 1D.

- VLFeat's 800 most confident matches among 10,000+ local features.



EPIPOLAR GEOMETRY: NOTATION



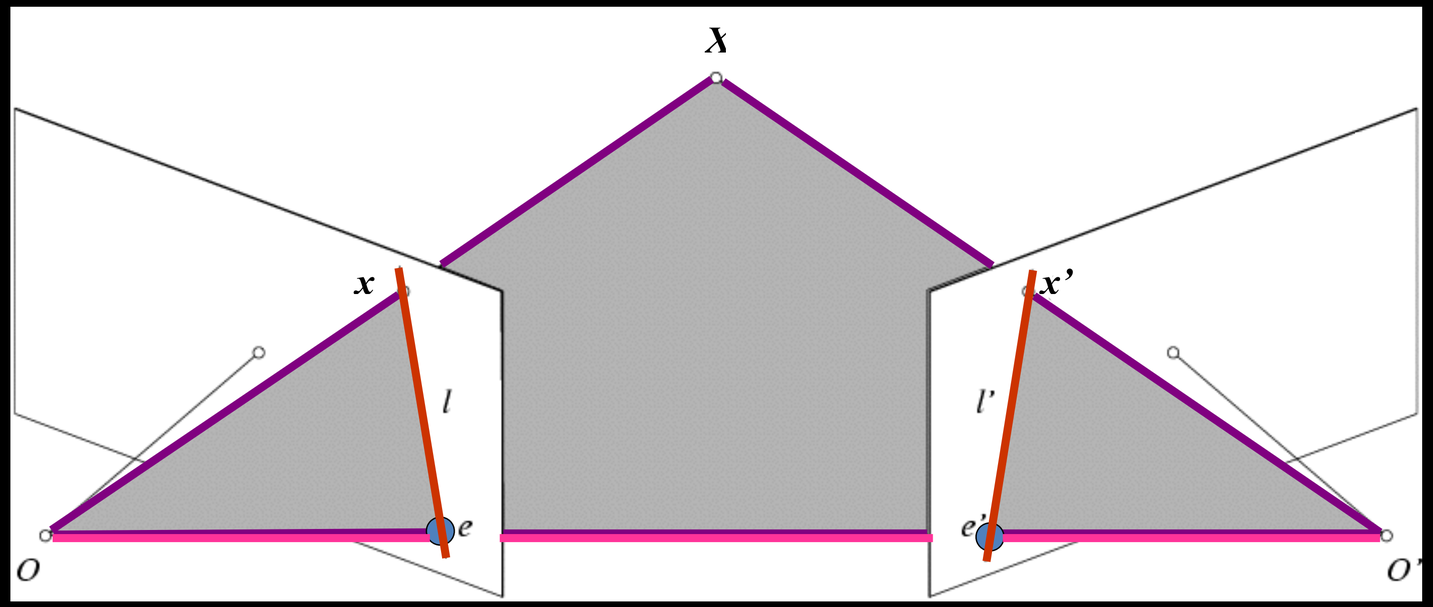
- **Baseline** – line connecting the two camera centers
- **Epipoles**

= intersections of baseline with image planes

= projections of the other camera center

- **Epipolar Plane** – plane containing baseline

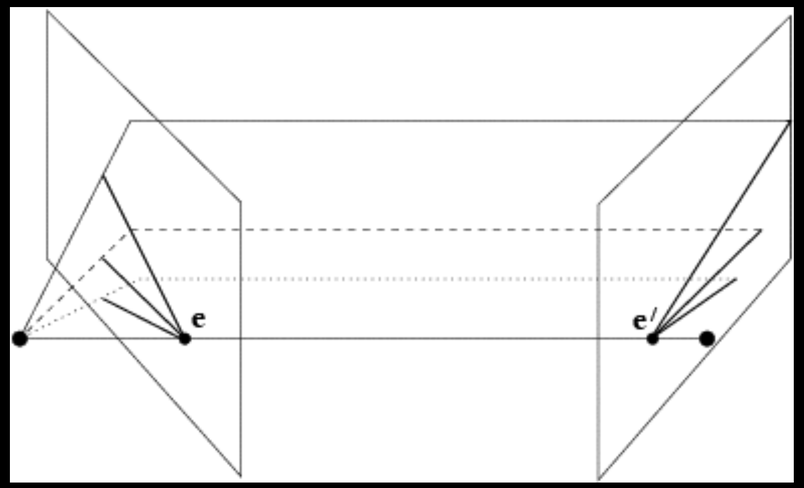
EPIPOLAR GEOMETRY: NOTATION

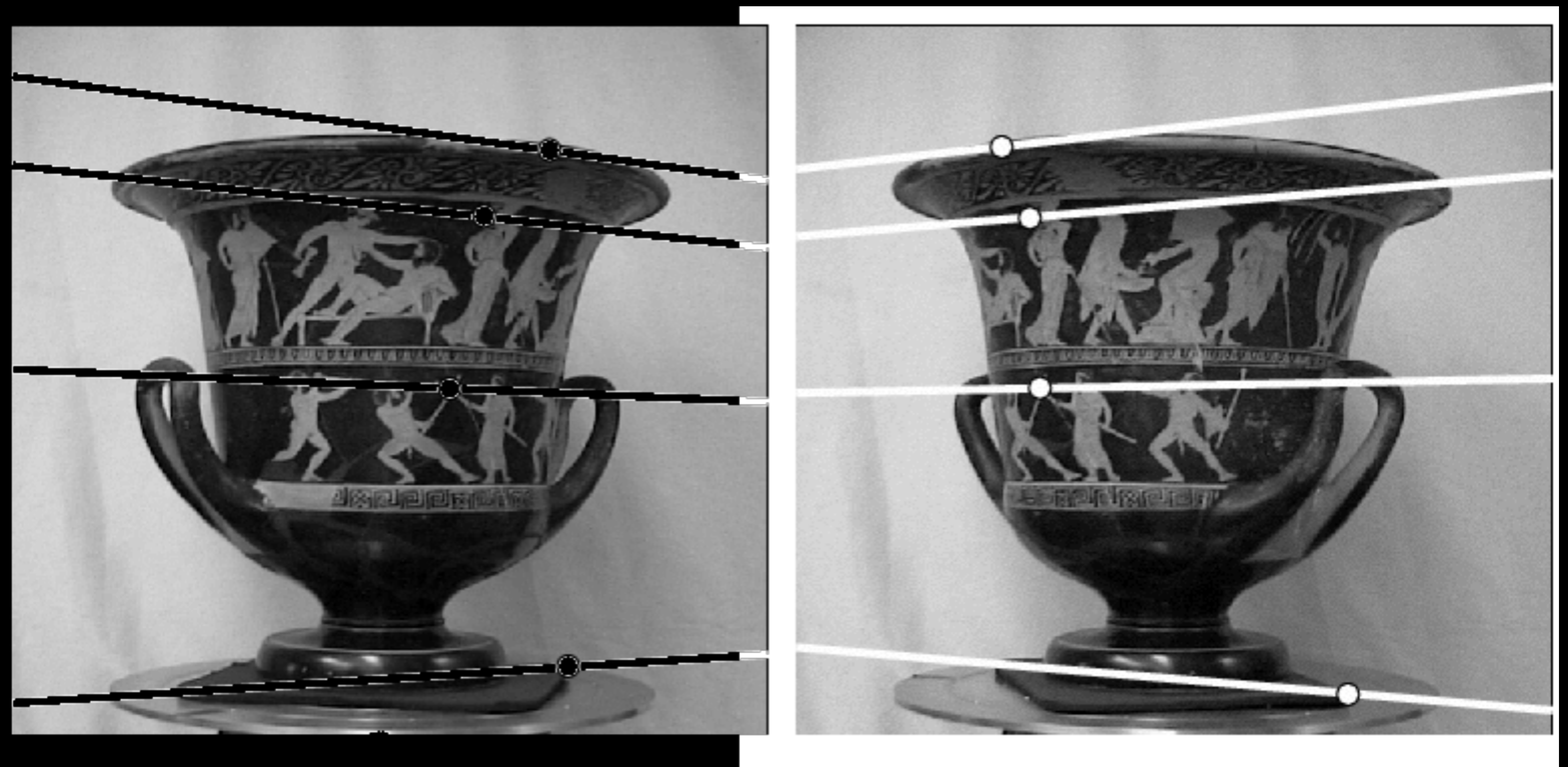


- **Baseline** – line connecting the two camera
- **Epipoles**
= intersections of baseline with image

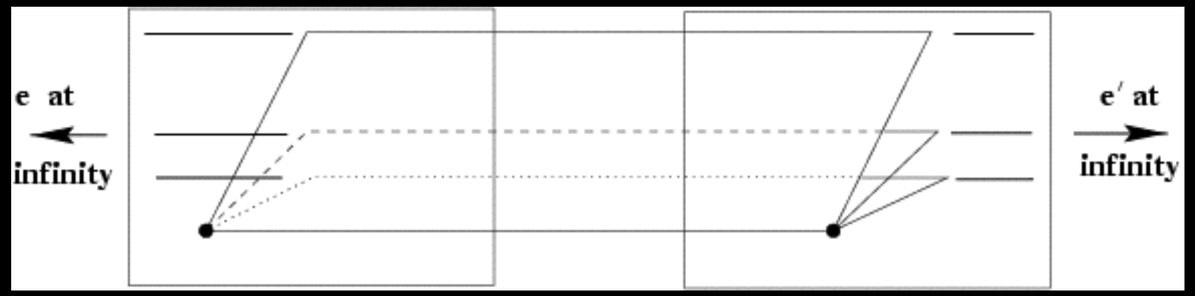
- **Epipolar Plane** – plane containing baseline
- **Epipolar Lines** - intersections of epipolar plane with image

EXAMPLE: CONVERGING CAMERAS





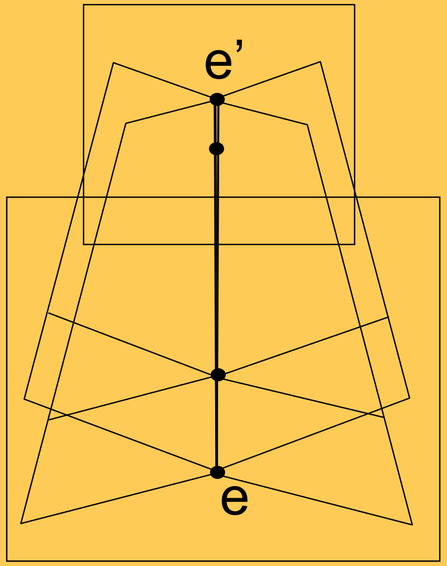
EXAMPLE: MOTION PARALLEL TO IMAGE PLANE



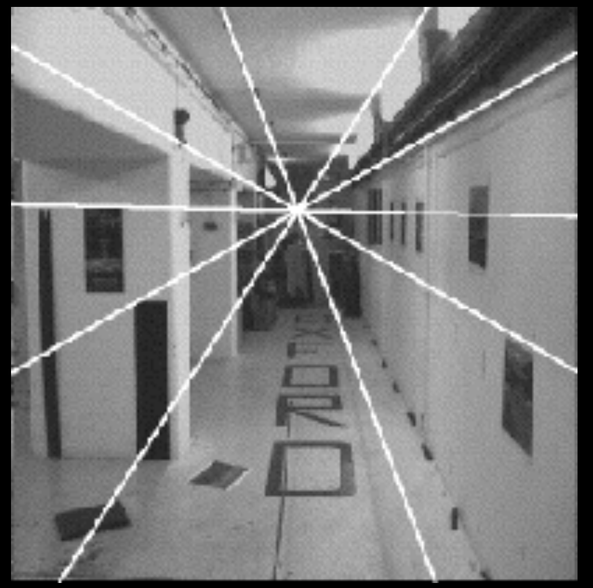
EXAMPLE: FORWARD MOTION

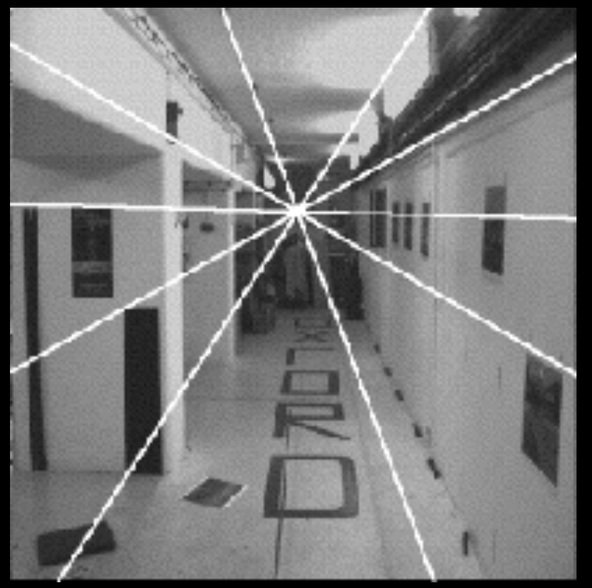
What would the epipolar lines look like if the camera moves directly forward?

EXAMPLE: FORWARD MOTION



Epipole has same coordinates in both images.
Points move along lines radiating from e:
“Focus of expansion”

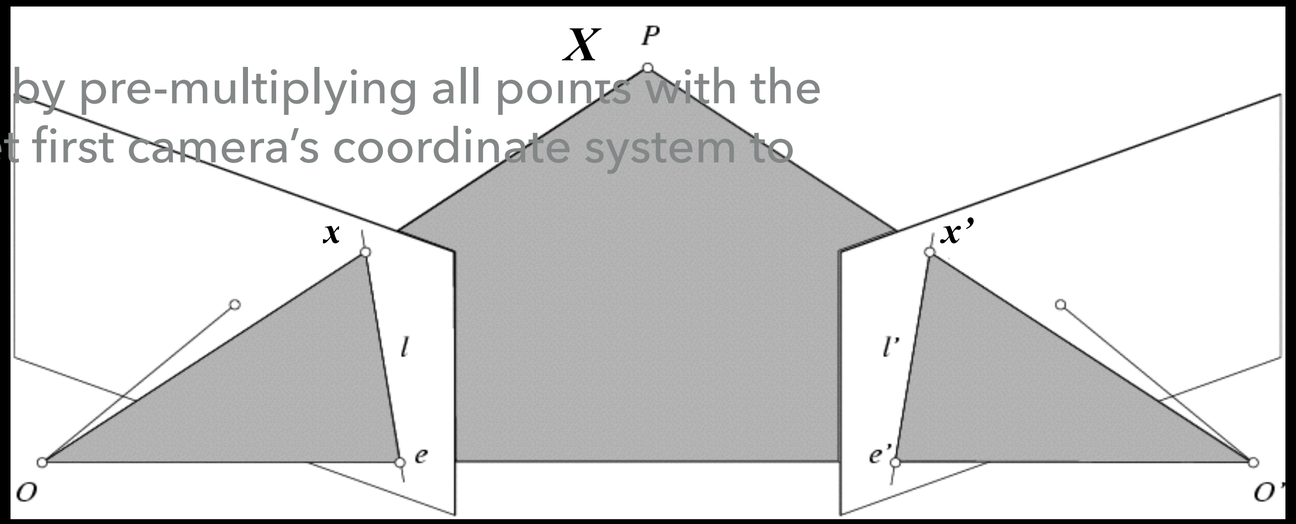




EPIPOLAR CONSTRAINT: CALIBRATED CASE

Given the intrinsic parameters of the cameras:

1. Convert to normalized coordinates by pre-multiplying all points with the inverse of the calibration matrix; set first camera's coordinate system to world coordinates



$$\hat{x} = K^{-1}x = X \qquad \hat{x}' = K'^{-1}x' = X'$$

Homogeneous 2d point (3D ray towards X)

2D pixel coordinate (homogeneous)

3D scene point

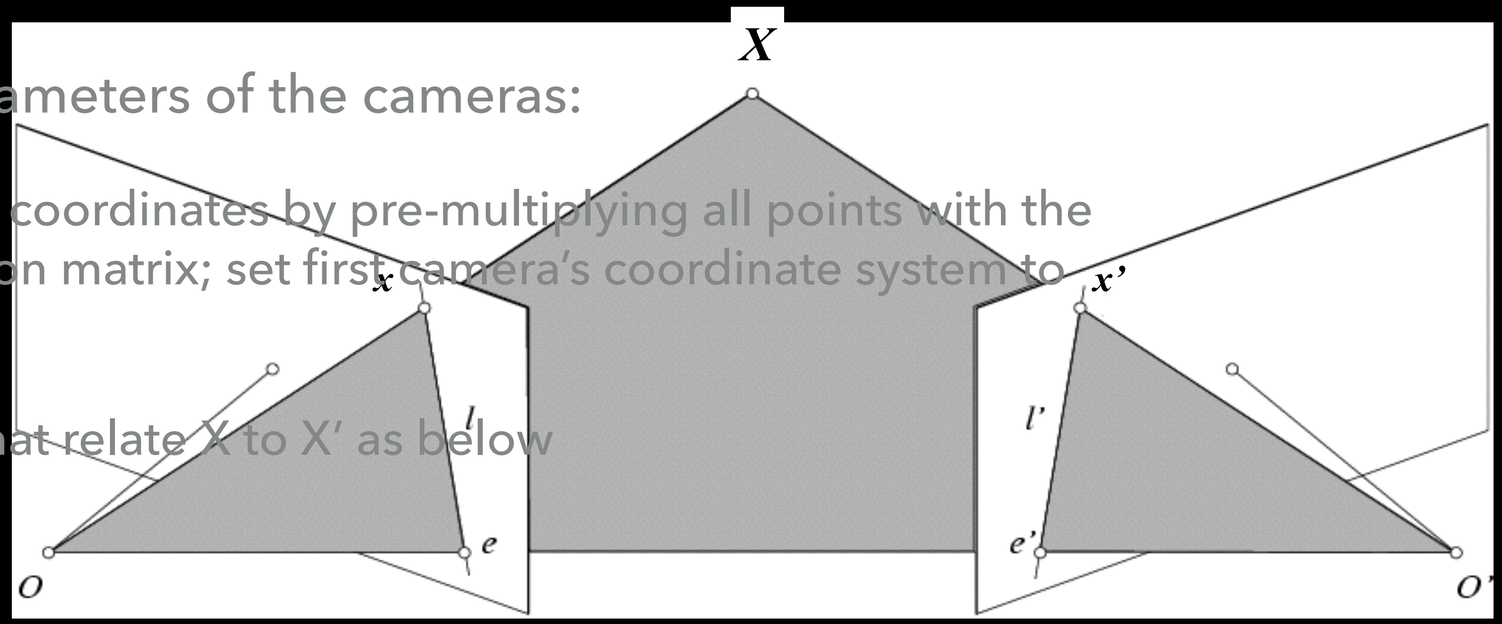
3D scene point in 2nd camera's 3D coordinates

TEXT

EPIPOLAR CONSTRAINT: CALIBRATED CASE

Given the intrinsic parameters of the cameras:

1. Convert to normalized coordinates by pre-multiplying all points with the inverse of the calibration matrix; set first camera's coordinate system to world coordinates
2. Define some *R* and *t* that relate X to X′ as below

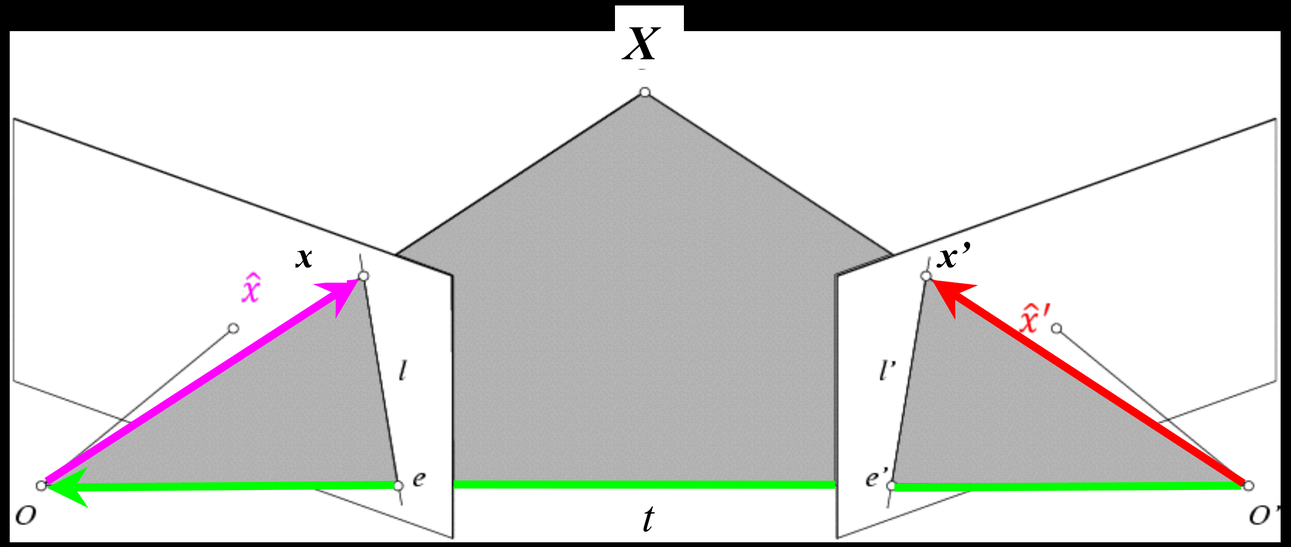


for some scale factor

$$\hat{x} = K^{-1}x = X \qquad \hat{x}' = K'^{-1}x' = X'$$

$$\hat{x} = R\hat{x}' + t$$

EPIPOLAR CONSTRAINT: CALIBRATED CASE

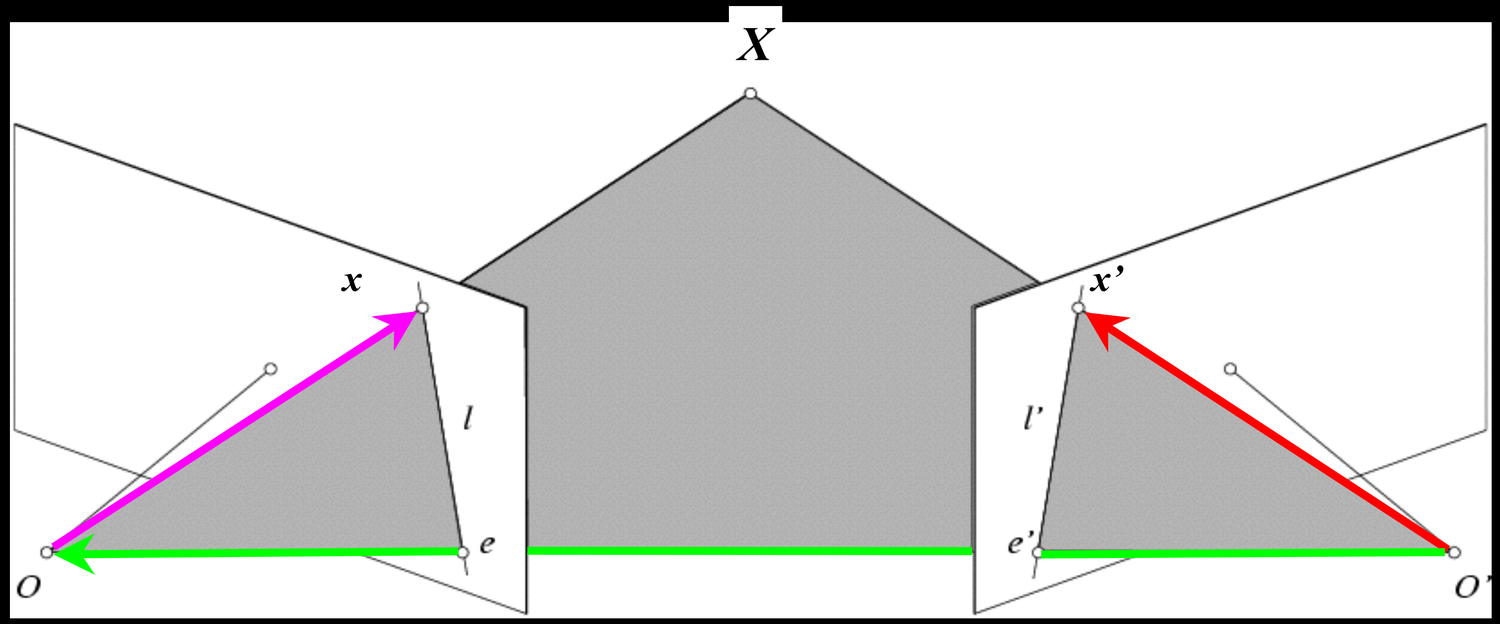


$$\hat{x} = K^{-1}x = X \qquad \hat{x}' = K'^{-1}x' = X'$$

$$\hat{x} = R\hat{x}' + t \quad \Rightarrow \quad \hat{x} \cdot [t \times (R\hat{x}')] = 0$$

(because $\hat{x}, R\hat{x}'$, and t are co-planar)

ESSENTIAL MATRIX

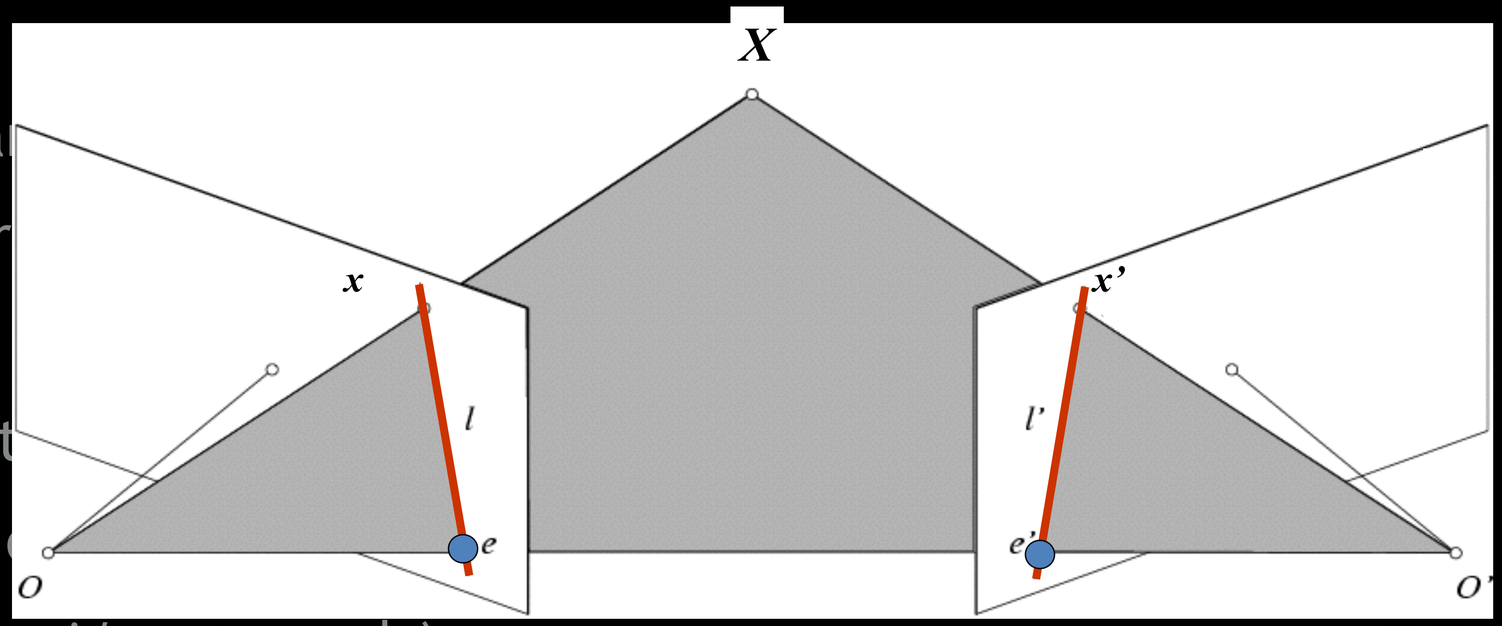


$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

Essential Matrix
(Longuet-Higgins, 1981)

PROPERTIES OF THE ESSENTIAL MATRIX

- $E\,x'$ is the epipolar
- $E^T x$ is the epipolar
- $E\,e' = 0$ and $E^T e$
- E is singular (rank t
- E has five degrees
  - (3 for R, 2 for t because it's up to a scale)



$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

Drop ^ below to simplify notation

Skew-symmetric matrix

THE FUNDAMENTAL MATRIX

Without knowing K and K′, we can define a similar relation using unknown normalized coordinates

$$\hat{x}^T E \hat{x}' = 0$$

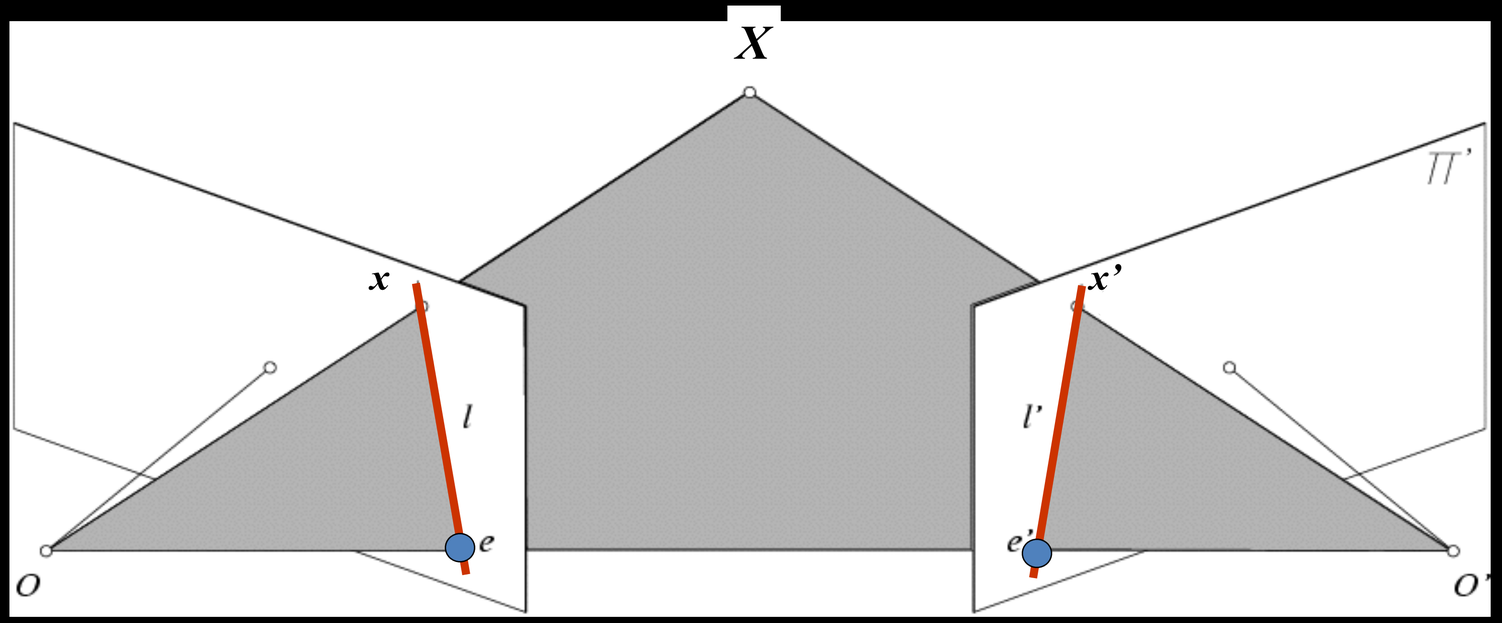
$$\hat{x} = K^{-1} x$$

$$\hat{x}' = K'^{-1} x'$$

$$\Rightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

Fundamental Matrix
(Faugeras and Luong, 1992)

PROPERTIES OF THE FUNDAMENTAL MATRIX



$$x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

F x′ is the epipolar line associated with x′ (l = F x′)

FTx is the epipolar line associated with x (l′ = FTx)

F e′ = 0 and FTe = 0

F is singular (rank two): det(F)=0

F has seven degrees of freedom: 9 entries but defined up to scale, det(F)=0