Important

Non-homogeneous linear least squares summary

(the part you need to know)

You should be able to set up

$$U\mathbf{x} = \mathbf{y}$$

You should know that it is solved by

$\mathbf{x} = U^{\dagger}\mathbf{y}$ where $U^{\dagger}$ is the pseudoinverse of U

You can assume that you can look up

$$U^{\dagger} = (U^T U)^{-1} U^T$$

*You should also keep in mind that for numerical stability, one may want to use a different approach to solve

$$U^T U\mathbf{x} = U^T\mathbf{y}$$

without matrix inversion.

Important

Non-homogeneous linear least squares (example two---naïve line fitting)

Can write y=mx + b as:

(x 1)*(m b) = y

So form

a matrix U with rows (x_i 1)

a vector **y** with elements y_i

a vector of unknowns **x=**(a,b)

and use the formula to solve U**x=y**

Quick "derivation" of formula for linear least squares

$U\mathbf{x} \cong \mathbf{b}$ (U has more rows than columns)

$U^T U\mathbf{x} \cong U^T\mathbf{b}$ (Multiply both sides by U^T)

$U^T U$ is likely to be robustly invertable

So, $\mathbf{x} \cong \left(U^T U\right)^{-1} U^T\mathbf{b} = U^{\dagger}\mathbf{b}$

Supplementary (optional) material.

Linear Least Squares (§3.1)

Problem statement. Find **x** that minimizes E where

$E = |\mathbf{e}|^{\mathbf{2}} = \mathbf{e}^T \mathbf{e}$ where $\mathbf{e} = U\mathbf{x} - \mathbf{y}$

For a minimum, $\dfrac{\delta E}{\delta x_i} = 0$, $\forall x_i$

(given no boundary conditions)

$$E = \sum_j e_j^2$$

$$\frac{\delta E}{\delta x_i} = 2\sum_j \frac{\delta e_j}{\delta x_i} \bullet e_j = 2\frac{\delta \mathbf{e}^T}{\delta x_i}\mathbf{e}$$

Supplementary (optional) material.

Linear Least Squares (§3.1)

$$\frac{\delta E}{\delta x_i} = 2\frac{\delta \mathbf{e}^T}{\delta x_i}\mathbf{e} = 0 \quad \text{(for minimum)}$$

This is true for all components, x_i, so we get:

$$\begin{pmatrix} \cdots \\ \\ \dfrac{\delta \mathbf{e}^T}{\delta x_i} \\ \\ \cdots \end{pmatrix}\mathbf{e} = 0$$

Supplementary (optional) material.

Linear Least Squares (§3.1)

The next step then is to evaluate $\frac{\delta \mathbf{e}^T}{\delta x_i}$

to get each row of a matrix, A, where A**e**=0

$$\frac{\delta \mathbf{e}^T}{\delta x_i} = \left(\frac{\delta \mathbf{e}}{\delta x_i}\right)^T = \left(\frac{\delta}{\delta x_i}(U\mathbf{x} - \mathbf{y})\right)^T = \left(\frac{\delta}{\delta x_i}U\mathbf{x}\right)^T$$

Supplementary (optional) material.

Linear Least Squares (§3.1)

Each row of A is $\left(\frac{\delta}{\delta x_i} U\mathbf{x}\right)^T$

$$(U\mathbf{x})_k = \sum_j U_{kj} x_j$$ (Let's study the k'th element of $U\mathbf{x}$)

$$\frac{\delta}{\delta x_i}(U\mathbf{x})_k = U_{ki}$$

Supplementary (optional) material.

Linear Least Squares (§3.1)

$$\frac{\delta}{\delta x_i}(U\mathbf{x})_k = U_{ki}$$ (k'th element of i'th column of U)

So $\frac{\delta}{\delta x_i}(U\mathbf{x})$ is the i'th column of U

And so $\frac{\delta \mathbf{e}^T}{\delta x_i} = \left(\frac{\delta}{\delta x_i} U\mathbf{x}\right)^T$ is the i'th row of U^T

So, the matrix referred to as A before, is U^T

Supplementary (optional) material.

Linear Least Squares (§3.1)

$$\frac{\delta \mathbf{e}^T}{\delta x_i} = \left(\frac{\delta}{\delta x_i} U\mathbf{x} \right)^T$$ is the i'th row of U^T

So $$\begin{pmatrix} \cdots \\ \frac{\delta \mathbf{e}^T}{\delta x_i} \\ \cdots \end{pmatrix} \mathbf{e} = 0$$ becomes $$U^T (U\mathbf{x} - \mathbf{y}) = 0$$

Supplementary (optional) material.

Linear Least Squares (§3.1)

From the previous slide our condition is $U^T(U\mathbf{x}-\mathbf{y})=0$

Or $U^T U\mathbf{x} = U^T\mathbf{y}$ (same as we got with our psuedo derivation)

So $\mathbf{x} = (U^T U)^{-1} U^T \mathbf{y}$

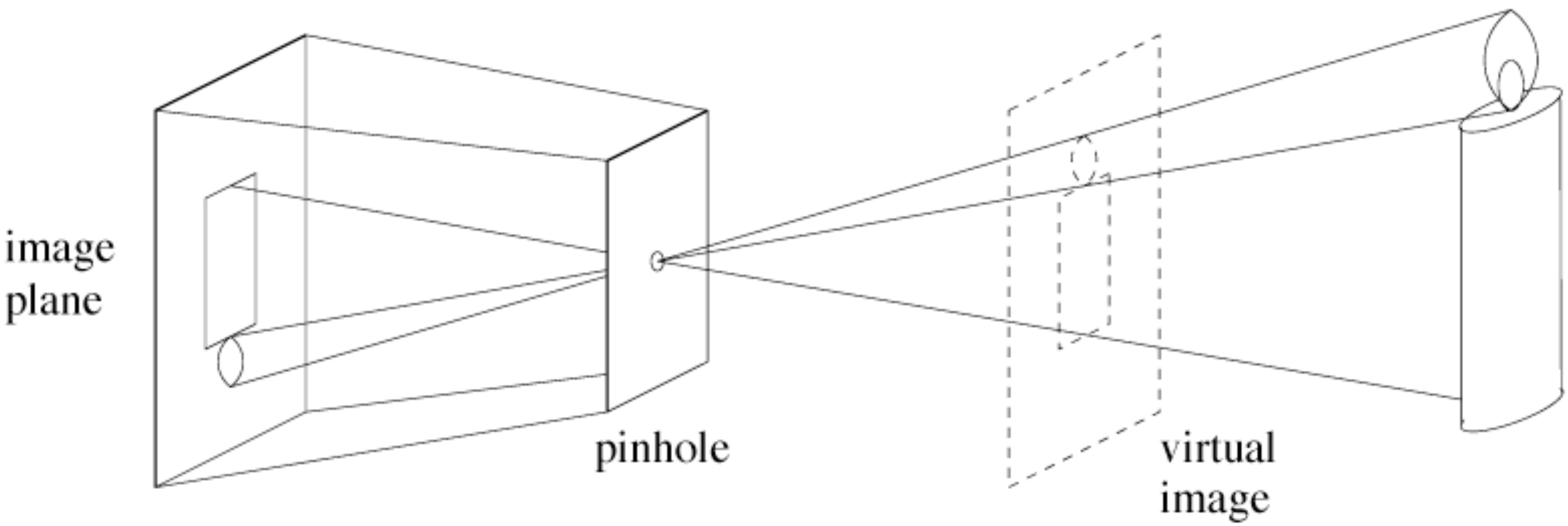
Thus

$\mathbf{x} = U^{\dagger}\mathbf{y}$ where $U^{\dagger} = (U^T U)^{-1} U^T$ is the pseudoinverse of U

Image Formation (Geometric)

Pinhole cameras

- Abstract camera model--box with a small hole in it
- Pinhole cameras work for deriving algorithms--a real camera needs a lens



Distant objects are smaller

Slide courtesy
Frank Dellaert

Size Constancy

Object size vs. object depth

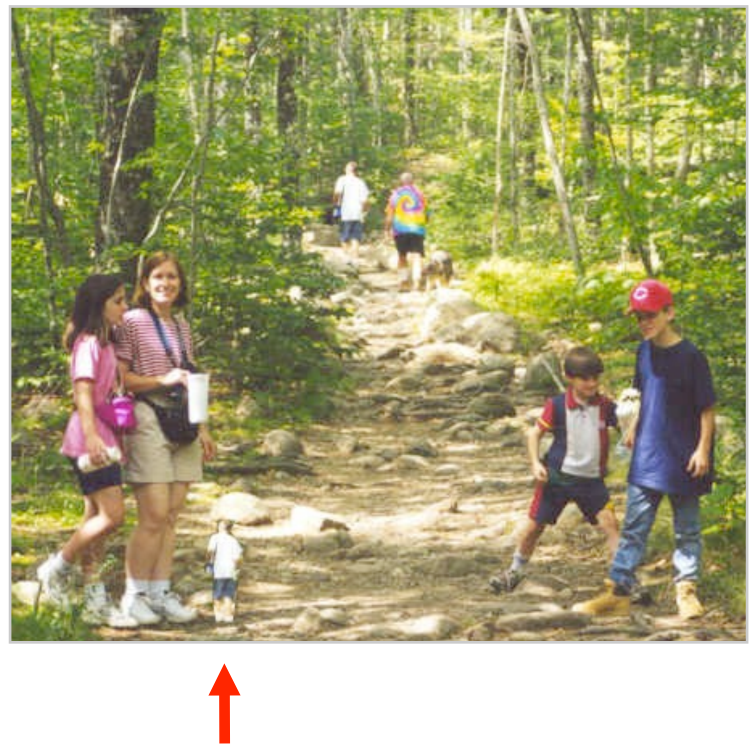
(Images copyright John H. Kranz, 1999)

Slide courtesy
Frank Dellaert

Size Constancy

Object size vs. object depth

(Images copyright John H. Kranz, 1999)



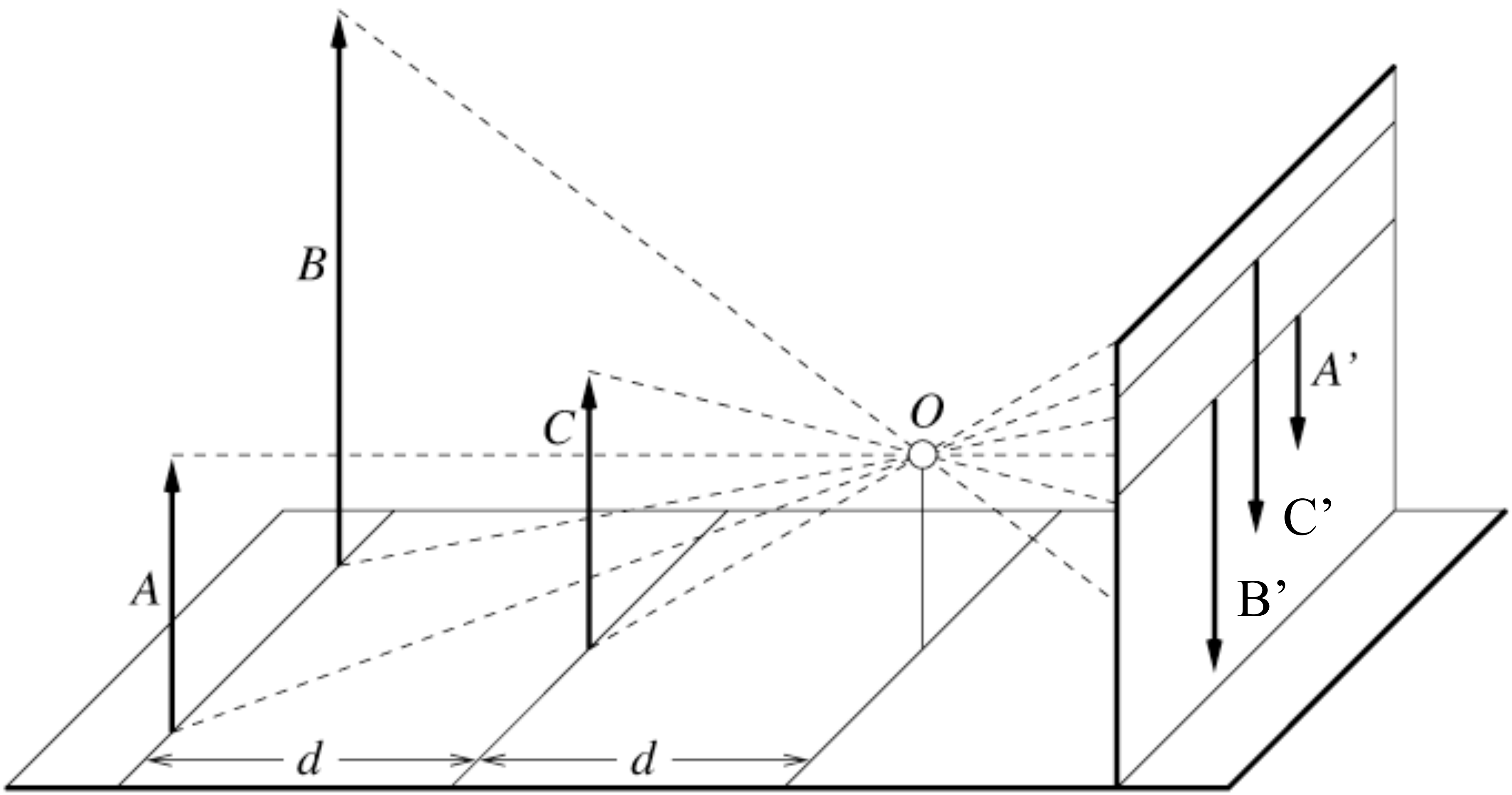
Slide courtesy
Frank Dellaert

Size Constancy

Object size vs. object depth

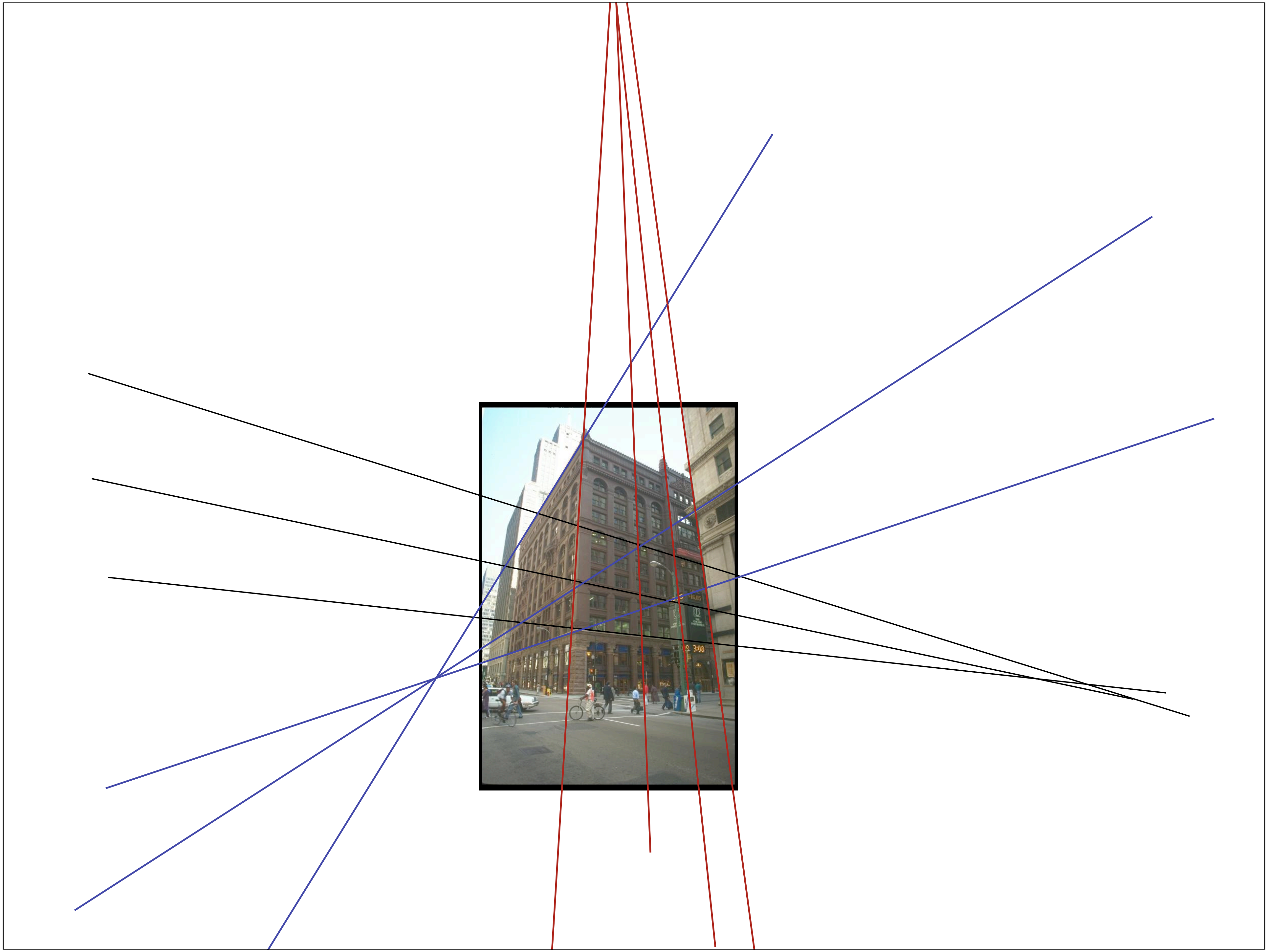
(Images copyright John H. Kranz, 1999)

Distant objects are smaller



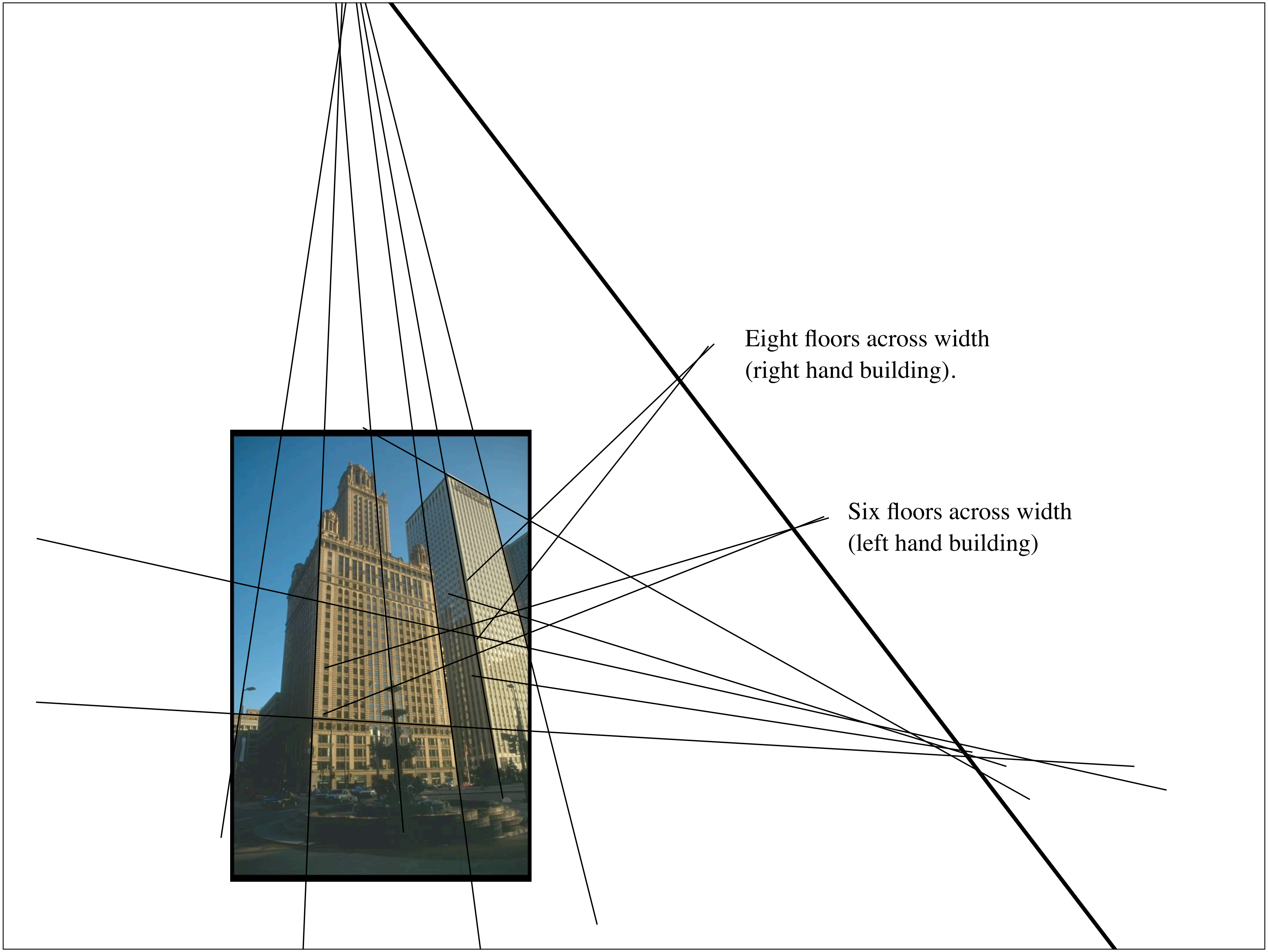
Vanishing points

- Each set of parallel lines (=direction) meets at a different point
  - The *vanishing point* for this direction



Vanishing points (cont)

- Sets of parallel lines on the same plane lead to *collinear* vanishing points.
  - The line is called the *horizon* for that plane
  - Standard horizon is the horizon of the ground plane.

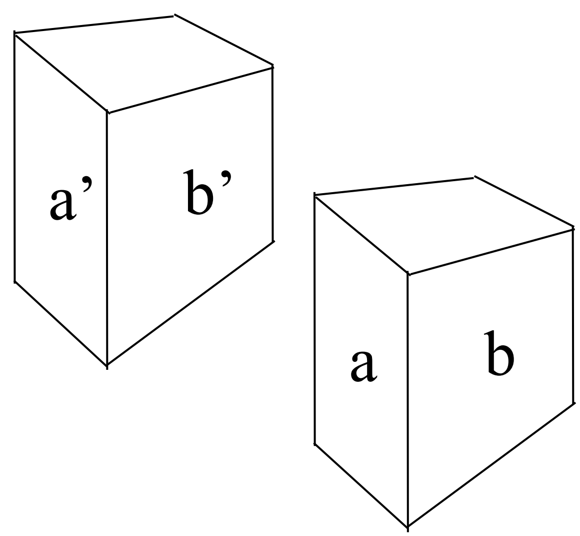

Eight floors across width
(right hand building).
Six floors across width
(left hand building)

Is the picture a fake?

- If scale and perspective don't work correctly, perhaps the image is a fake!

- We can check if:
  - Each set of parallel lines (=direction) meets at a different point
  - Sets of parallel lines on the same plane lead to *collinear* vanishing points.

Example: The figure below is claimed to provide a perspective view of two identical cubes, with faces a and a', and faces b and b' being parallel. Provide reasons why this could not be a real perspective drawing of the geometry described, marking any needed explanatory lines on the figure.



Geometric properties of projection

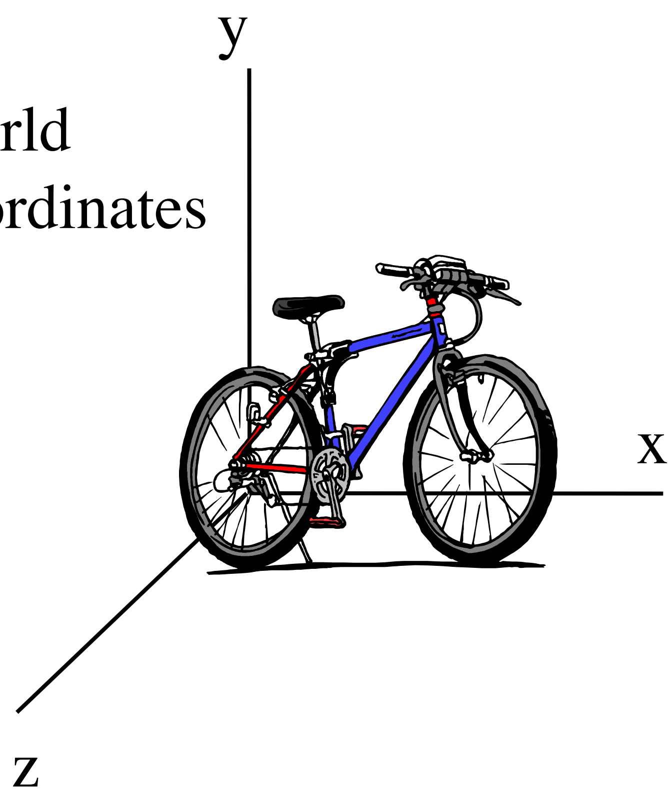
- Points go to points
- Lines go to lines
- Polygons go to polygons
- Degenerate cases
  - line through focal point projects to a point
  - plane through focal point projects to a line

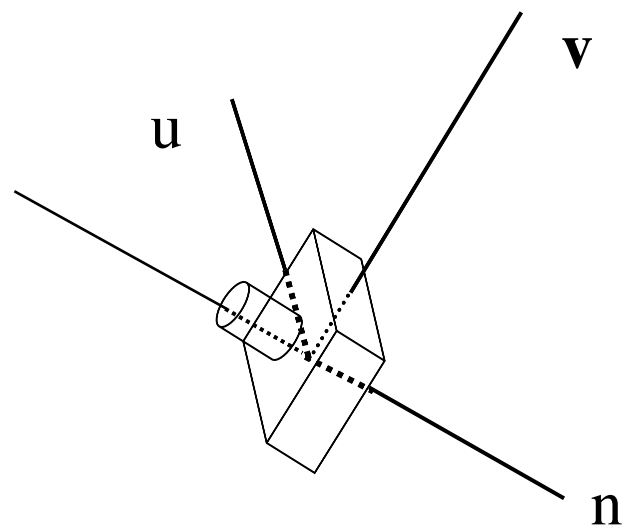
Geometric Camera Model

- Let **P**=(X,Y,Z) be a point in space.
- Let (u,v) be image coordinates.
- A geometric camera model, G, tells us where P goes in the image.
- (u,v) = G(**P**)

World and camera coordinates

World
coordinates





Geometric Camera Model

- Transform world coordinates to standard camera coordinates
  - (Extrinsic parameters)
- Project onto standard camera plane
  - (3D becomes 2D)
- Transform into pixel locations
  - (Intrinsic camera parameters)

Math aside, #2

Representing Transformations

- Need mathematical representation for mapping points from the world to an image (and later, from an image taken by one camera to another).
- Represent linear transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.
- To transform line segments, transform endpoints
- To transform polygons, transform vertices

2D Transformations

- Represent **linear** transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.
- Recall the definition of matrix times vector:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{pmatrix}$$

Matrix multiplication is linear

- In particular, if we define f(**x**)=M • **x**, where M is a matrix and **x** is a vector, then

$$
\begin{aligned}
f(a\mathbf{x}+b\mathbf{y}) &= M(a\mathbf{x}+b\mathbf{y}) \\
&= aM\mathbf{x}+bM\mathbf{y} \\
&= af(\mathbf{x})+bf(\mathbf{y})
\end{aligned}
$$

- Where the middle step can be verified using algebra (supplementary slide and/or homework)

Supplemental material

Proof that matrix multiplication is linear

$$M(a\mathbf{x}+b\mathbf{y}) = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} ax_1 + by_1 \\ ax_2 + by_2 \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}ax_1 + a_{11}by_1 + a_{12}ax_2 + a_{12}by_2 \\ a_{21}ax_1 + a_{21}by_1 + a_{22}ax_2 + a_{22}by_2 \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}ax_1 + a_{12}ax_2 + a_{11}by_1 + a_{12}by_2 \\ a_{21}ax_1 + a_{22}ax_2 + a_{21}by_1 + a_{22}by_2 \end{pmatrix}$$

$$= a\begin{pmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{pmatrix} + b\begin{pmatrix} a_{11}y_1 + a_{12}y_2 \\ a_{21}y_1 + a_{22}y_2 \end{pmatrix}$$

$$= aM\mathbf{x} + bM\mathbf{y}$$

Composition of Transformations

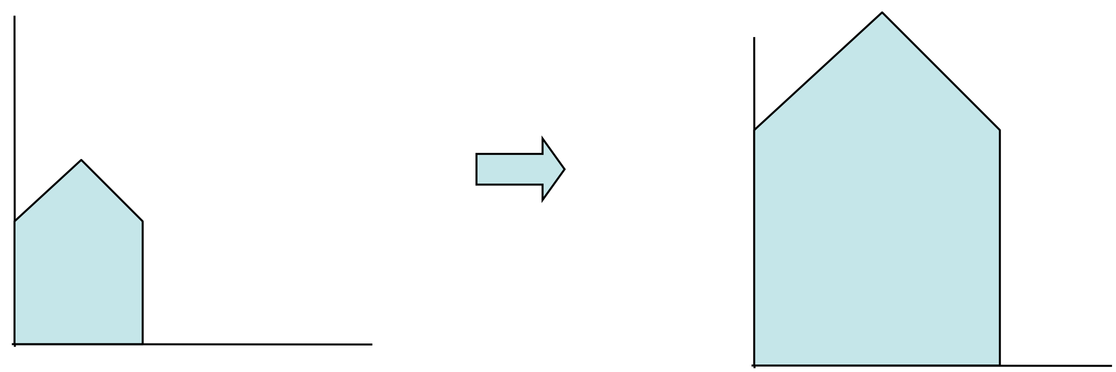
- If we use one matrix, M_1 for one transform and another matrix, M_2 for a second transform, then the matrix for the first transform followed by the second transform is simply $M_2 M_1$

- This follows from the associativity of matrix multiplication
  - $M_2 (M_1\mathbf{x}) = (M_2 M_1)\mathbf{x}$

- This generalizes to any number of transforms

Transformation examples in 2D

- Scale (stretch) by a factor of k

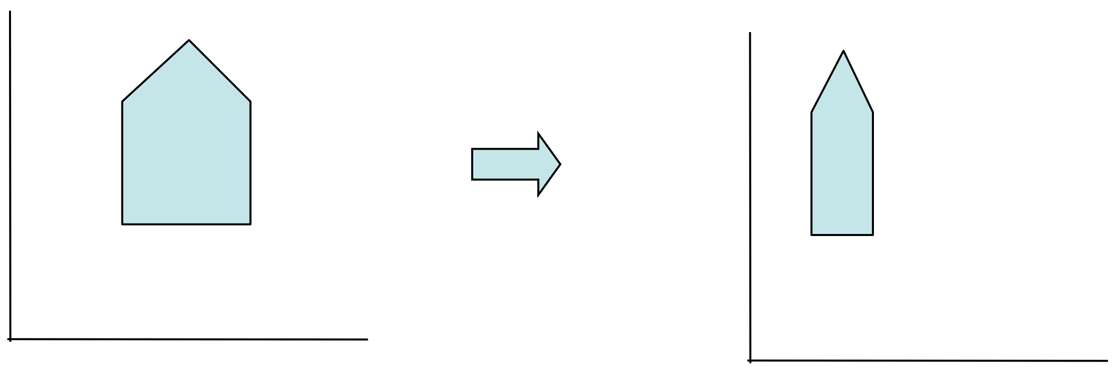


$$M = \begin{vmatrix} k & 0 \\ 0 & k \end{vmatrix}$$

(k = 2 in the example)

Transformation examples in 2D

- Scale by a factor of (S_x, S_y)



$$M = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$

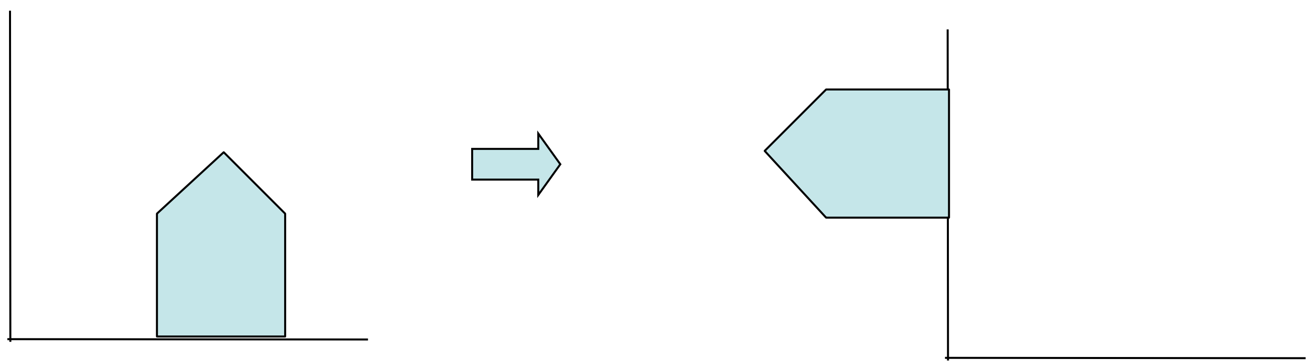
(Above, $S_x = 1/2$, $S_y = 1$)

Orthogonal Transformations

- Orthogonal transformations are defined by $O^T O = I$
- Also have $|\det(O)| = 1$
- Rigid body rotations and mirror “flip”

Transformation examples in 2D

- Rotate around origin by θ (Orthogonal)

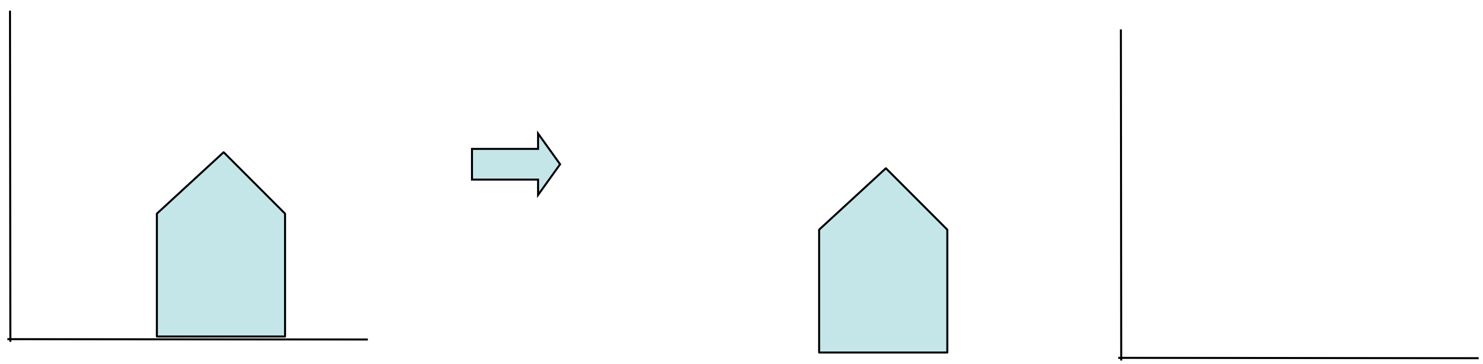


$$M = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$

(Above, $\theta = 90^o$)

Transformation examples in 2D

- Mirror flip through y axis (Orthogonal)

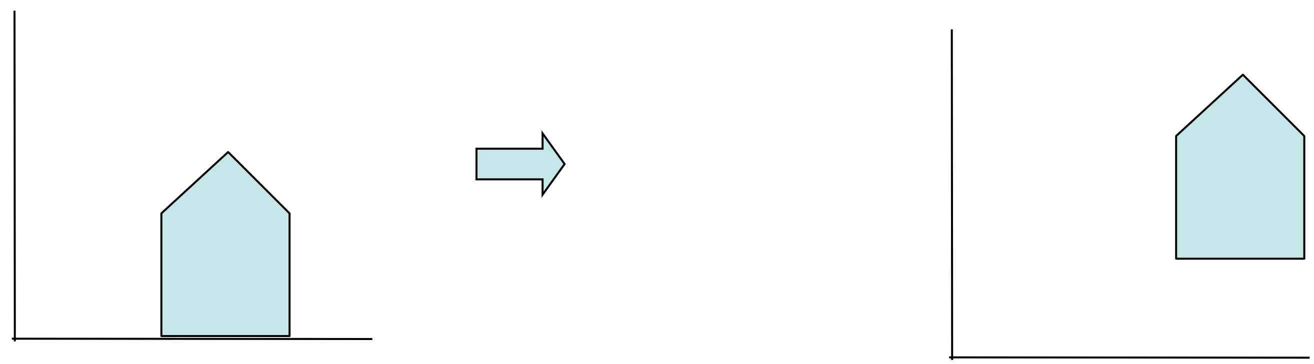


$$M = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$$

Flip over x axis is ?

2D Transformations

- Translation $(\mathbf{P}_{new} = \mathbf{P} + \mathbf{T})$



M =?

Homogenous Coordinates

- Represent 2D points by 3D vectors
- (x,y)-->(x,y,1)
- Now a multitude of 3D points (x,y,W) represent the same 2D point, (x/W, y/W, 1)
- Represent 2D transforms with 3 by 3 matrices
- Can now represent translations by matrix multiplications