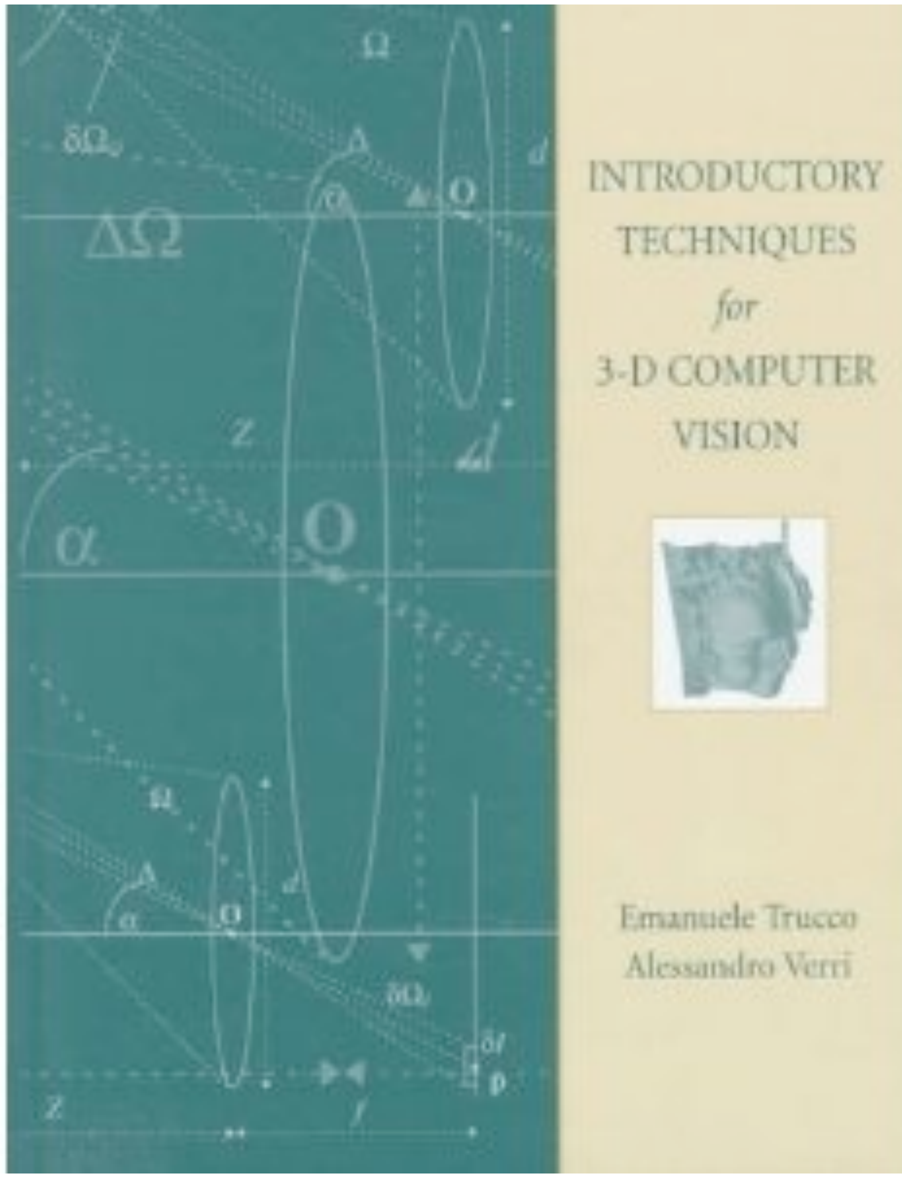
15-381
ARTIFICIAL INTELLIGENCE

LECTURE 15:
VISION II: GEOMETRY

FALL 2010

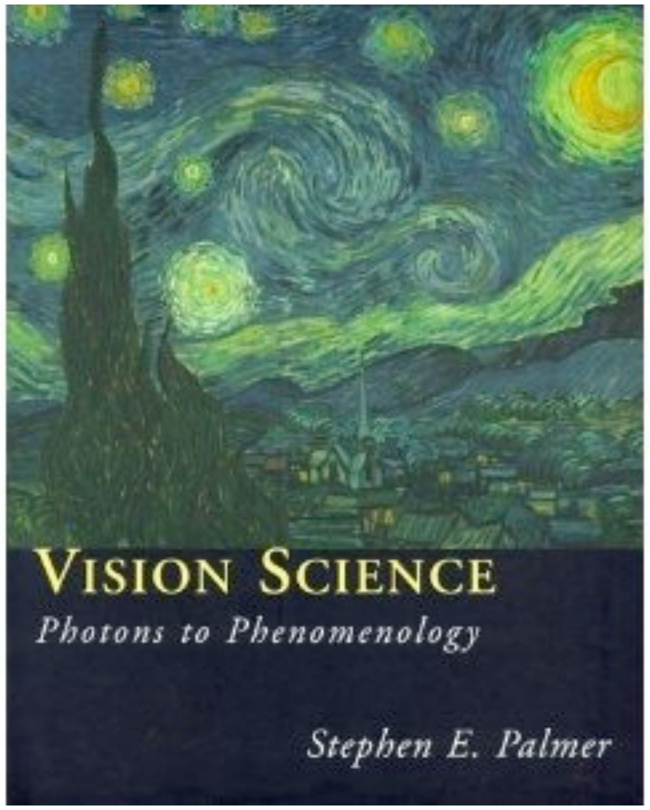
TEXTBOOK

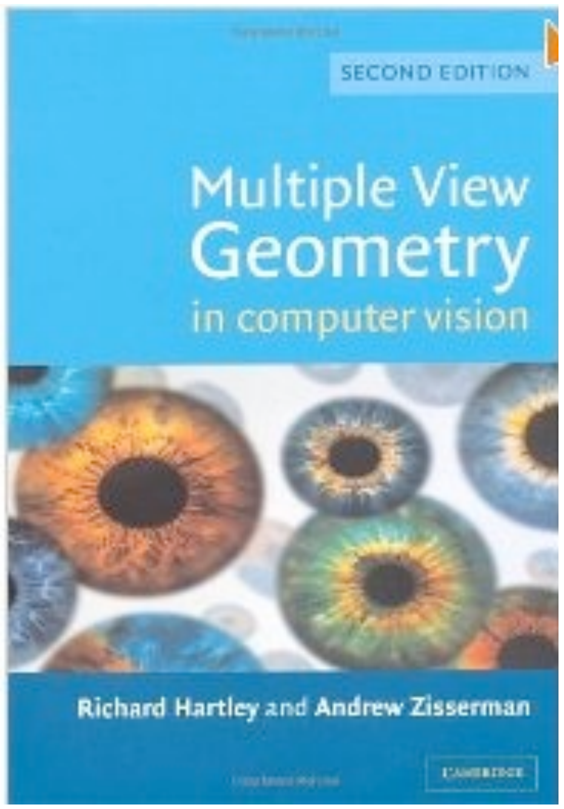


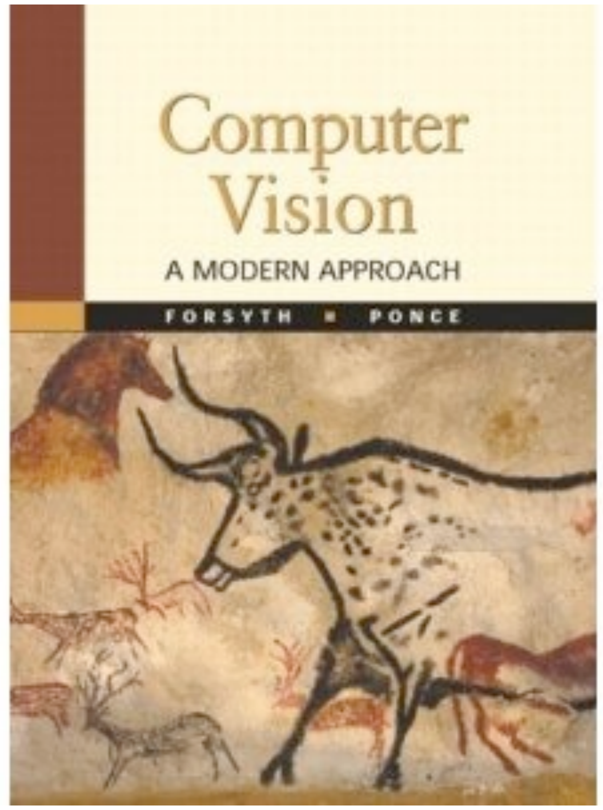
Introductory Technique for 3D Computer Vision

Trucco and Verri

VISION BOOKS







WHERE DOES A PIXEL COME FROM?

2048 x 3072 x 3

$$\begin{bmatrix} \text{red} \\ \text{green} \\ \text{blue} \\ x \\ y \end{bmatrix} = \begin{bmatrix} 178 \\ 172 \\ 158 \\ 545 \\ 1540 \end{bmatrix} \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \end{array}\right\} \begin{array}{l} \textbf{radiometry} \\ \text{last lecture} \end{array} \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \begin{array}{l} \textbf{geometry} \\ \text{this lecture} \end{array} \end{array}$$

SHAPE AND MOTION FROM IMAGE STREAMS UNDER ORTHOGRAPHY (1992)

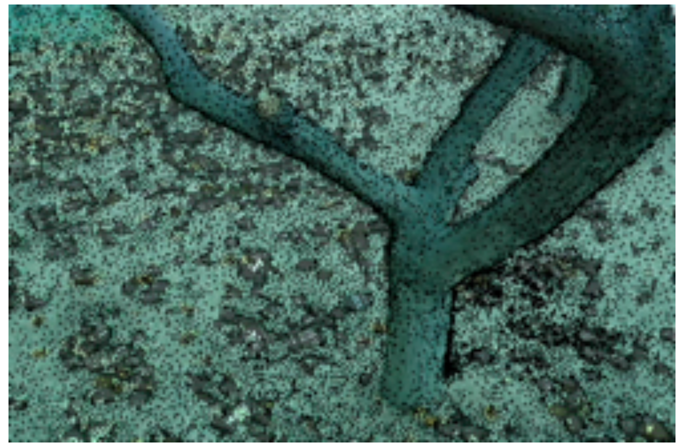
Presented by: Yaser Sheikh

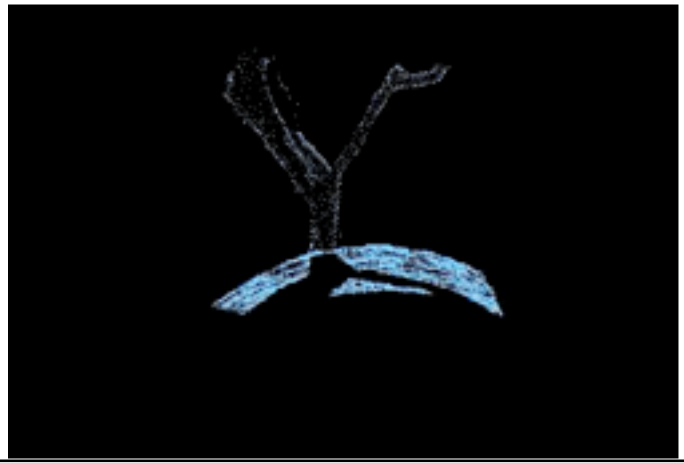
CARLO TOMASI AND TAKEO KANADE

INPUT: Points tracked across a video captured by a camera moving about an object

OUTPUT: 3D structure of the object and the camera motion

METHOD:





Core idea: Under orthographic projection, camera motion and object structure are "separable" from image measurements

- Despite having many measurements (100s of points in 100s of frames) the measurements were actually highly correlated.
- **A rank 3 (or rank 4) condition derived on the measurements**
- Singular Value Decomposition (SVD) was used to recover the camera motion and the 3D structure

WHY IS THIS PAPER INTERESTING?

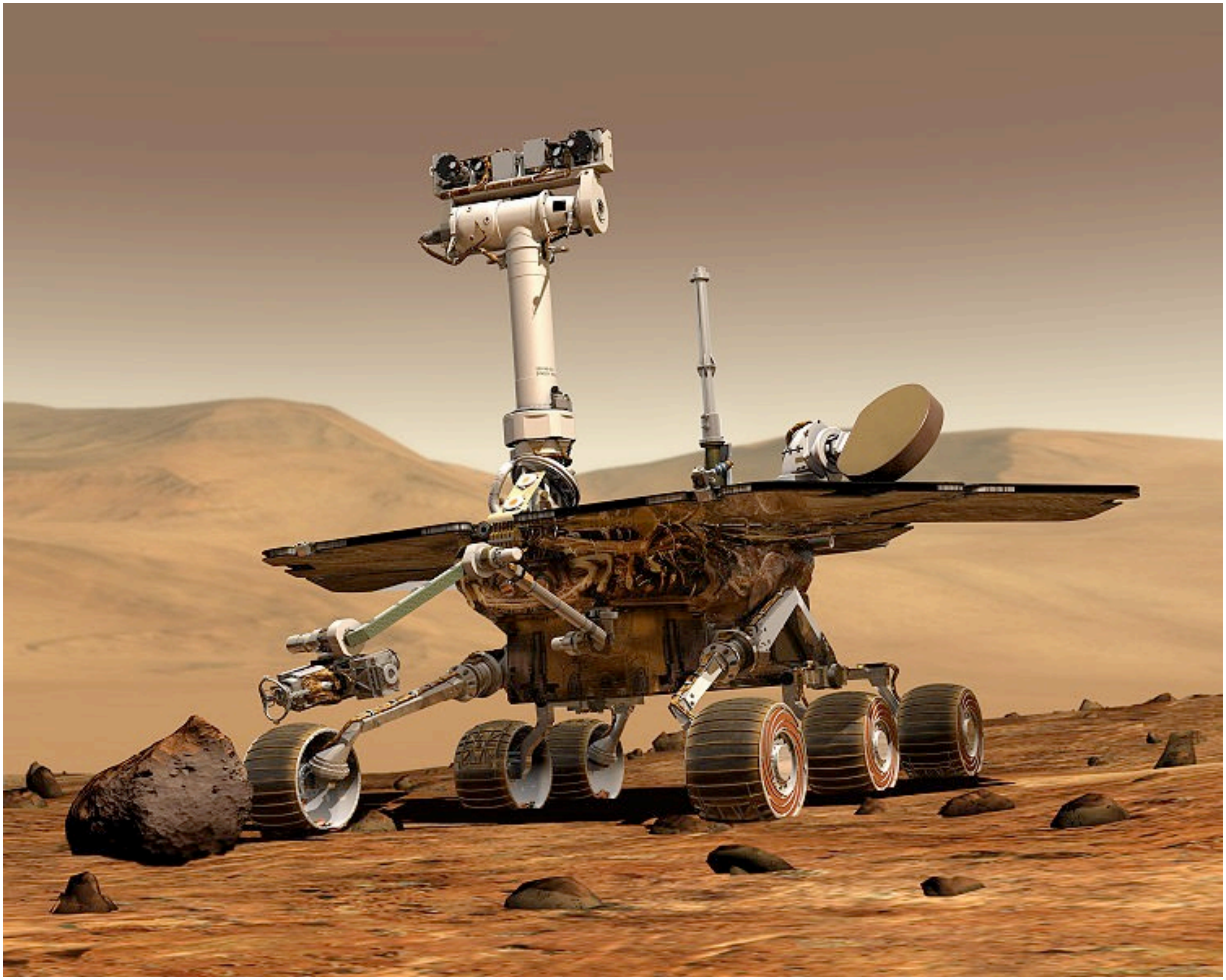
- 3D reconstruction allows 3D navigation for robots, match-moving for the movie industry, photogrammetry, visualization, etc.
- First stable algorithm to recover 3D structure from video. Spurred two decade of reconstruction research

HOW WOULD I IMPROVE THIS PAPER?

- The paper assumes an **orthographic** camera. Can we derive it for a perspective camera?
- The paper assumes a stationary object. Can we derive a similar algorithm for when the object moves during capture?

WHY RECOVER 3D?

Snavely et al. , SIGGRAPH, 2006

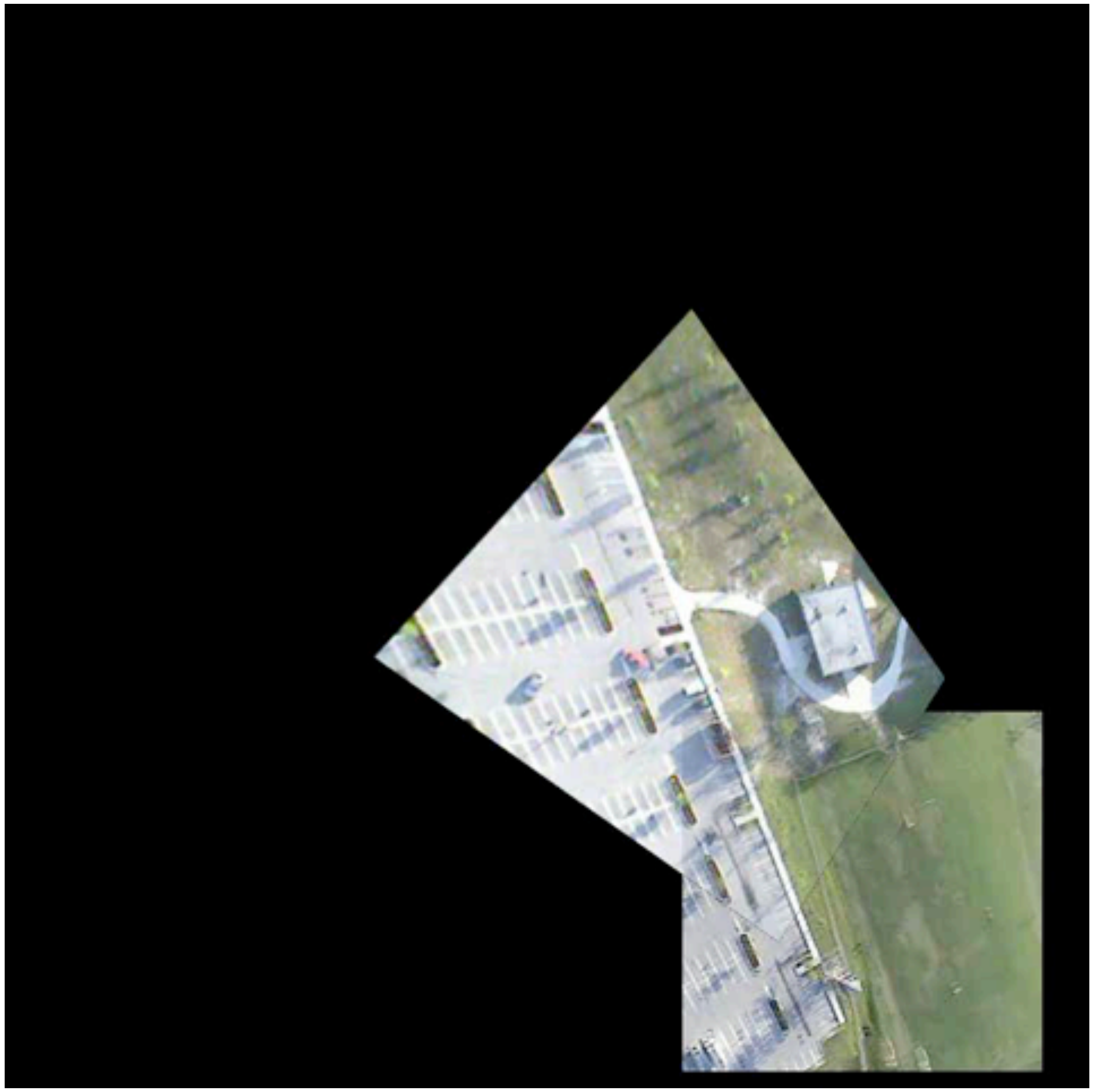


Barnum et al. , ISMAR, 2009

Hoeim et al. , SIGGRAPH, 2005

WHY RECOVER 3D?

Jain et al. , SCA, 2010



Sheikh et al. , TPAMI, 2008

PANORAMAS

LINEAR ALGEBRA PRIMER

MATRICES

$$\mathbf{A}_{3\times 3} = \begin{array}{c} \begin{array}{cc} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} & \text{3 rows} \end{array} \\ \text{3 columns} \end{array}$$

$$\mathbf{A}_{3\times 3} = \begin{bmatrix} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ | & | & | \end{bmatrix}$$

$$\mathbf{A}_{N\times M} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_M \\ | & | & & | \end{bmatrix} = \boxed{N \times M}$$

LINEAR ALGEBRA PRIMER

VECTOR TRANSFORMATIONS

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} ae + bf \\ ce + df \end{bmatrix} = \begin{bmatrix} e\begin{bmatrix} a \\ c \end{bmatrix} + f\begin{bmatrix} b \\ d \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_M \\ | & | & & | \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{bmatrix} = \begin{bmatrix} | & & | & & | \\ v_1\mathbf{a}_1 & + & v_2\mathbf{a}_2 & + \cdots & v_M\mathbf{a}_M \\ | & & | & & | \end{bmatrix}$$

LINEAR ALGEBRA PRIMER

RANK

Rank: Number of linearly independent rows or columns

$$\mathbf{A}_{3\times 3} = \begin{bmatrix} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ | & | & | \end{bmatrix}$$

9 values define **A**

$$\mathbf{a}_3 = b_1\mathbf{a}_1 + b_2\mathbf{a}_2$$

$$c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + c_3\mathbf{a}_3 = 0$$

$$\mathbf{A}_{3\times 3} = \begin{bmatrix} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & b_1\mathbf{a}_1 + b_2\mathbf{a}_2 \\ | & | & | \end{bmatrix}$$

8 values define **A**

LINEAR ALGEBRA PRIMER

RANK

$$\mathbf{A}_{1000\times1000} = \begin{array}{c} \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_{1000} \\ | & | & & | \end{bmatrix} \text{1000 rows} \\ \text{1000 columns} \end{array}$$

1,000,000 values define **A**

$$\mathbf{a}_{1000} = b^1_{1000}\mathbf{a}_1 + b^2_{1000}\mathbf{a}_2$$

$$\mathbf{a}_i = b^1_i\mathbf{a}_1 + b^2_i\mathbf{a}_2, \forall i \in \{1, \dots, 1000\}$$

$$\mathbf{A}_{1000\times1000} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & b^1_{1000}\mathbf{a}_1 + b^2_{1000}\mathbf{a}_2 \\ | & | & & | \end{bmatrix}$$

4000 values define **A** if **A** is rank 2

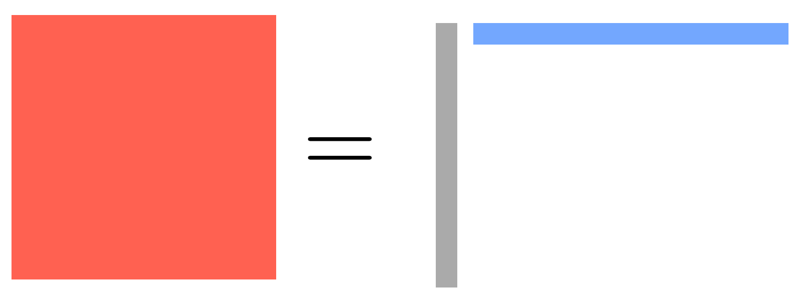
LINEAR ALGEBRA PRIMER

RANK

$$\mathbf{A}_{1000\times 1000} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & b^1_{1000}\mathbf{a}_1 + b^2_{1000}\mathbf{a}_2 \\ | & | & & | \end{bmatrix}$$

$$\mathbf{A}_{1000\times 1000} = \begin{bmatrix} | & | \\ \mathbf{a}_1 & \mathbf{a}_2 \\ | & | \end{bmatrix} \begin{bmatrix} -\mathbf{b}_1- \\ -\mathbf{b}_2- \end{bmatrix}$$

$$= \mathbf{A}_{1000\times 2}\mathbf{B}_{2\times 1000}$$

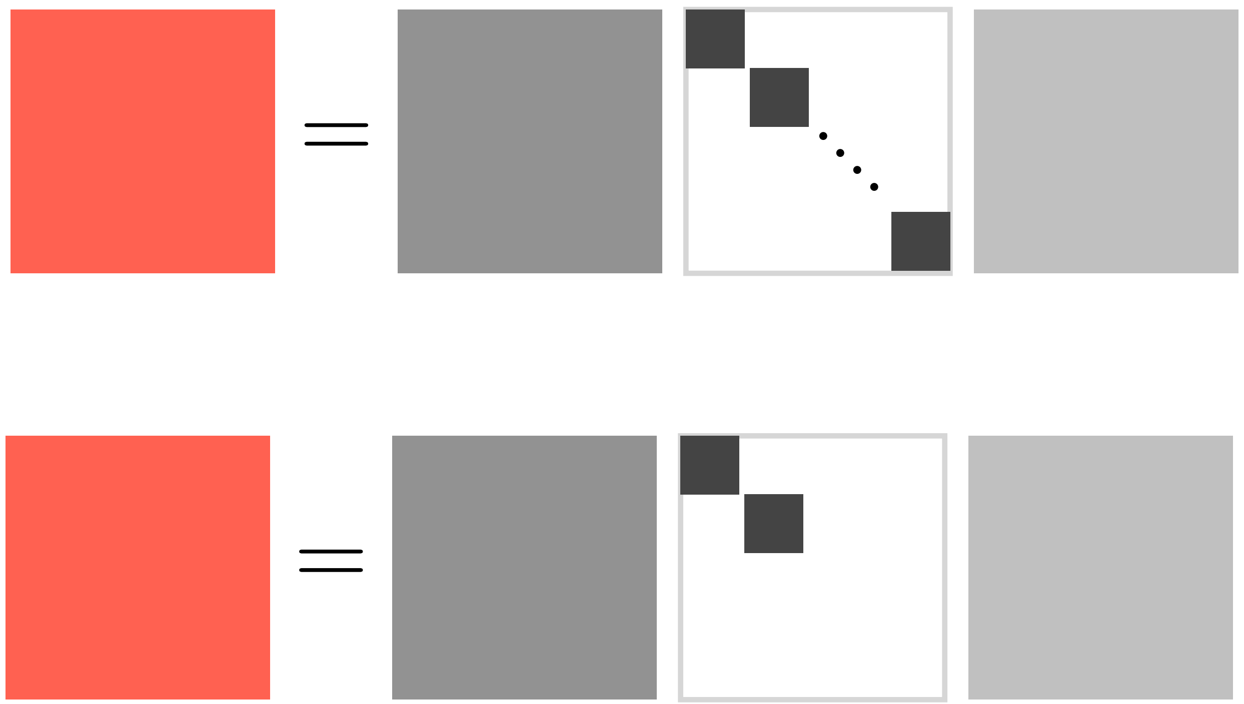


LINEAR ALGEBRA PRIMER

SINGULAR VALUE DECOMPOSITION

Any *m* x *n* matrix **A** can be written as a product of three matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$



Rank 2 means the diagonal has only two non-zero elements

LINEAR ALGEBRA PRIMER

SINGULAR VALUE DECOMPOSITION

$$\mathbf{A} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}^T$$

if **A** is rank 1 then

$$\mathbf{A} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}^T$$

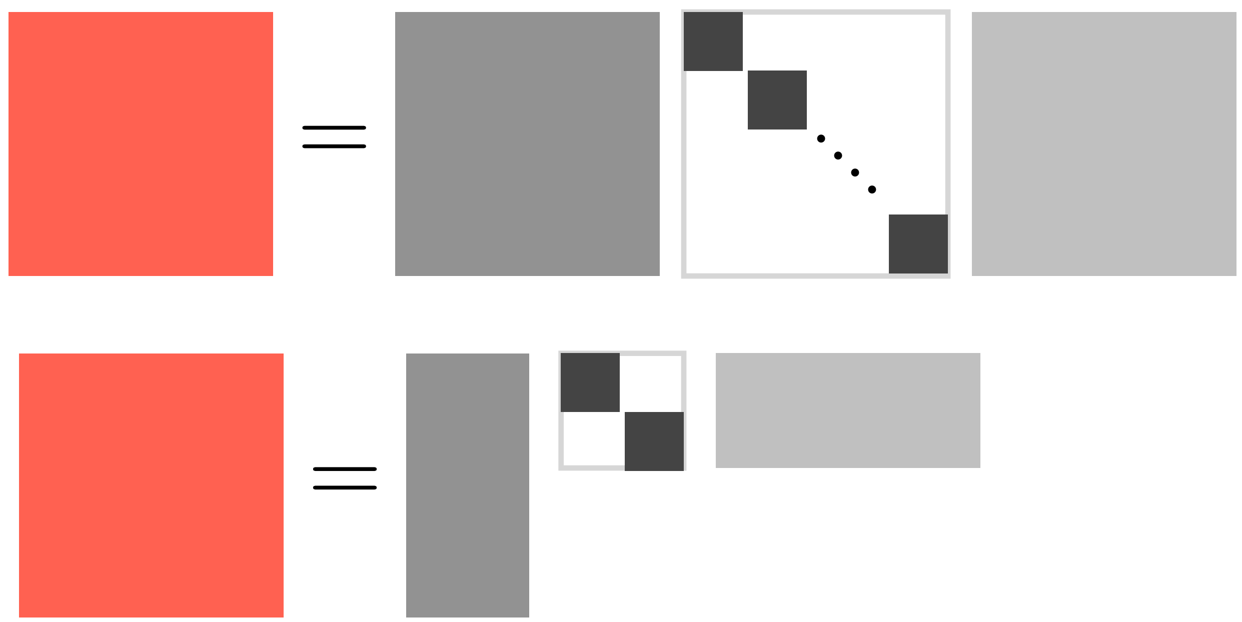
$$\mathbf{A} = \begin{bmatrix} u_{11} \\ u_{21} \\ u_{31} \end{bmatrix} \begin{bmatrix} s_1 \end{bmatrix} \begin{bmatrix} v_{11} & v_{21} & v_{31} \end{bmatrix}$$

LINEAR ALGEBRA PRIMER

SINGULAR VALUE DECOMPOSITION

Any *m* x *n* matrix **A** can be written as a product of three matrices:

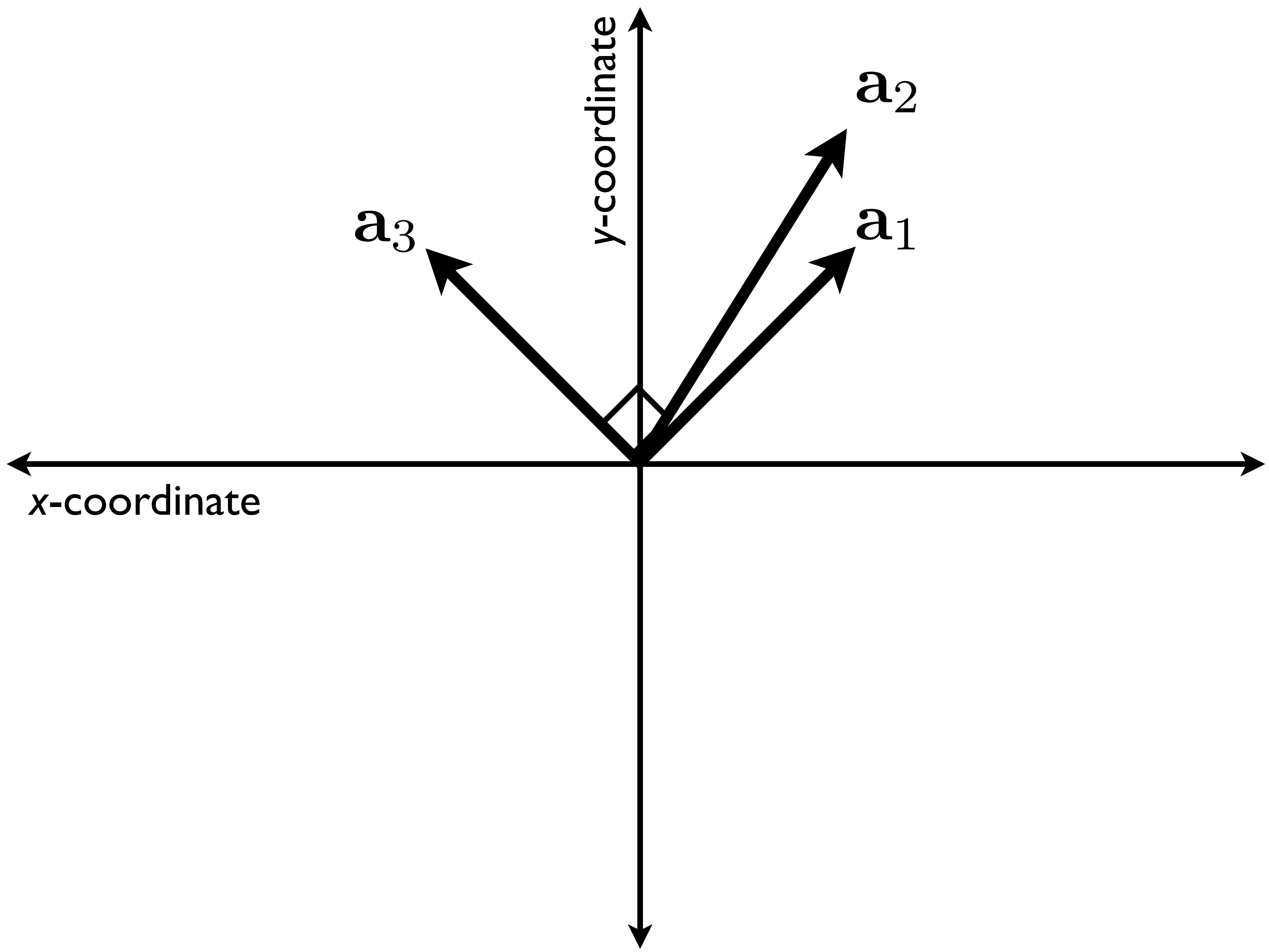
$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$



Note: SVD is <u>widely</u> used for many purposes. We're only interested in it for these property for now

LINEAR ALGEBRA PRIMER

ORTHOGONALITY



LINEAR ALGEBRA PRIMER

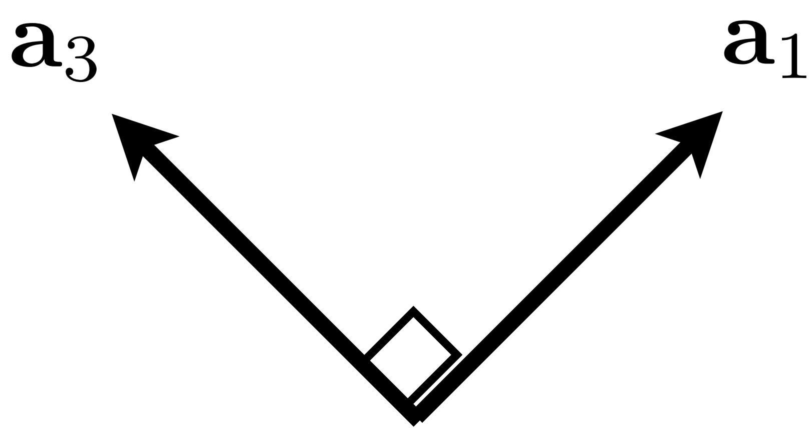
ORTHOGONALITY

Two vectors $\mathbf{a}_1$ and $\mathbf{a}_3$ are orthogonal if

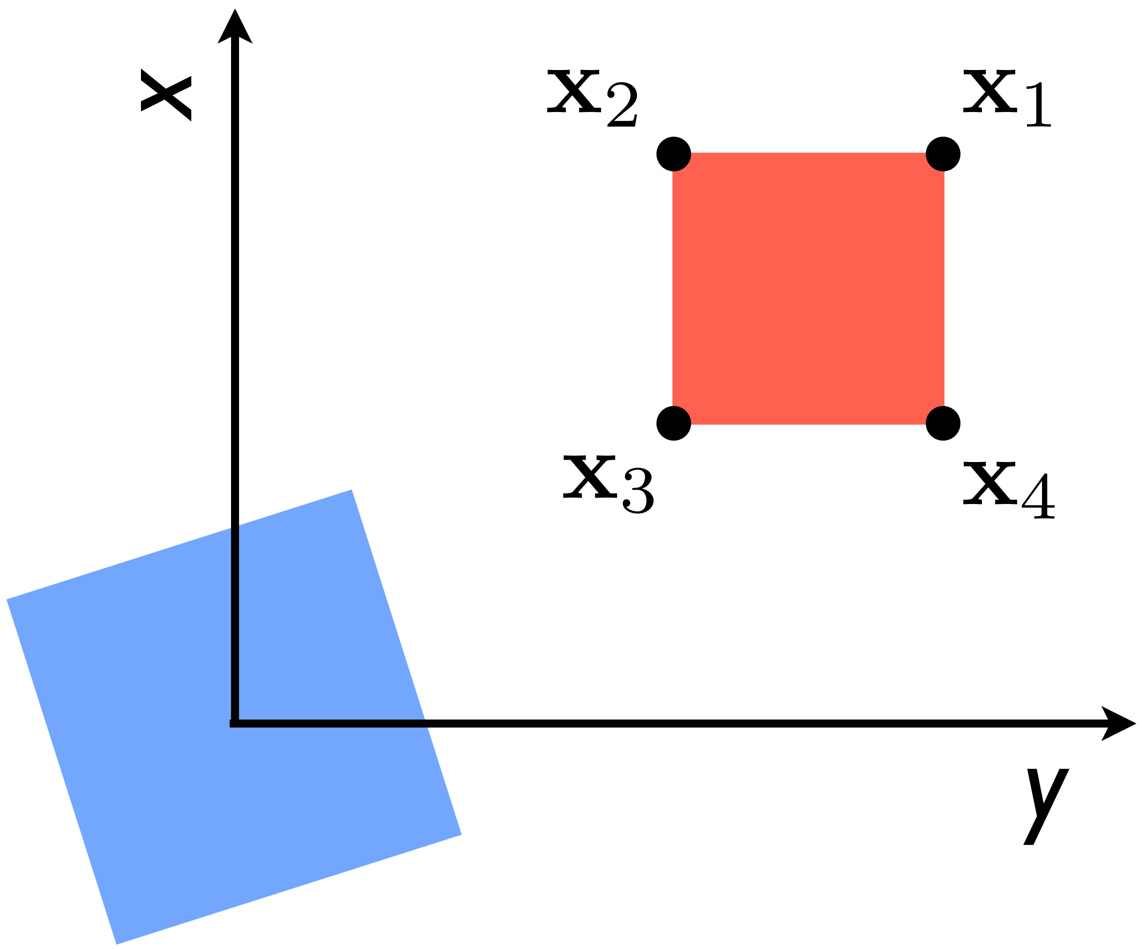
$$\mathbf{a}_1^T \mathbf{a}_3 = 0$$

$$\mathbf{a}_1^T \mathbf{a}_1 = 1$$

$$\mathbf{a}_3^T \mathbf{a}_3 = 1$$

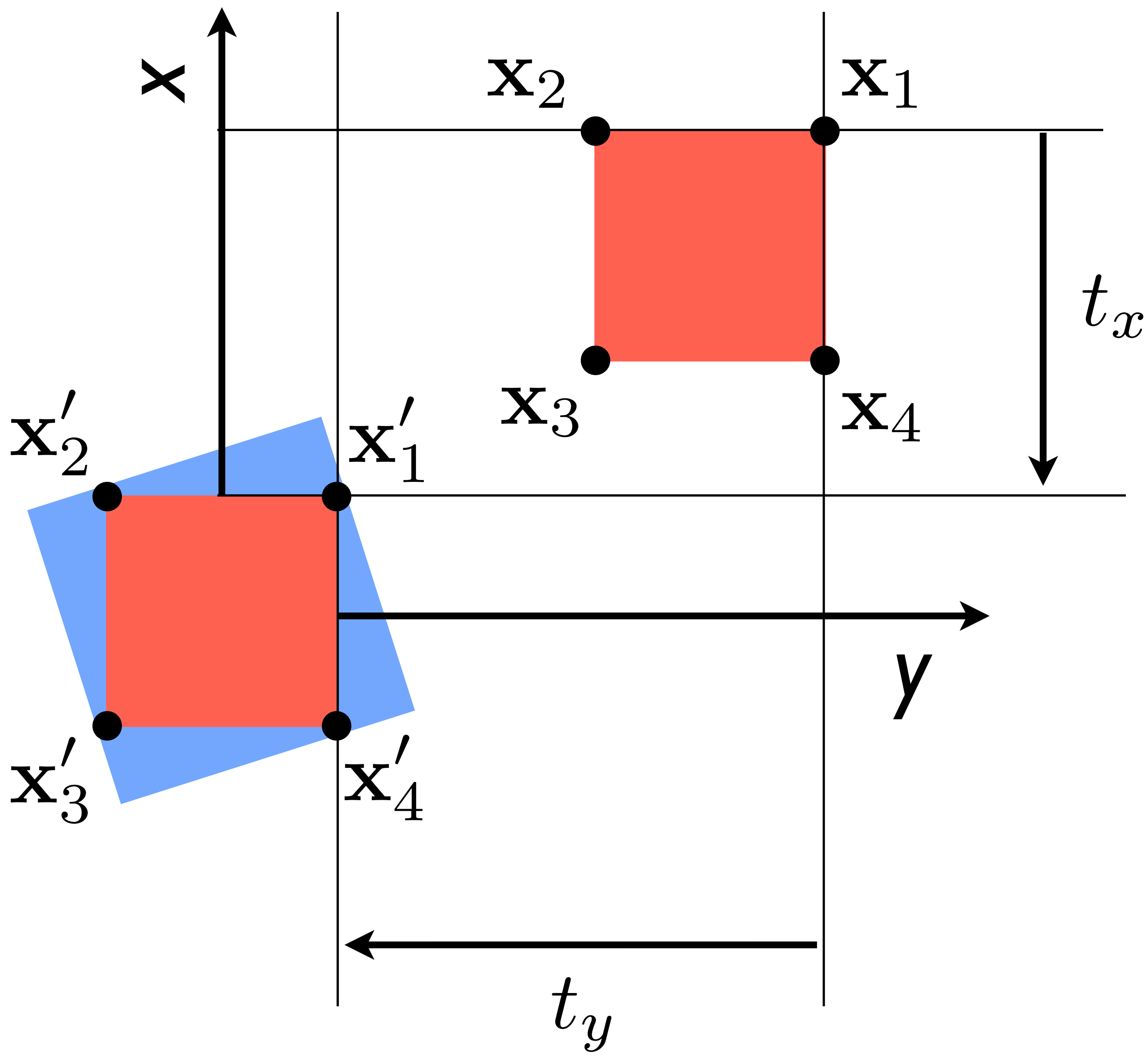


2D GEOMETRY



$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

2D TRANSLATION



$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

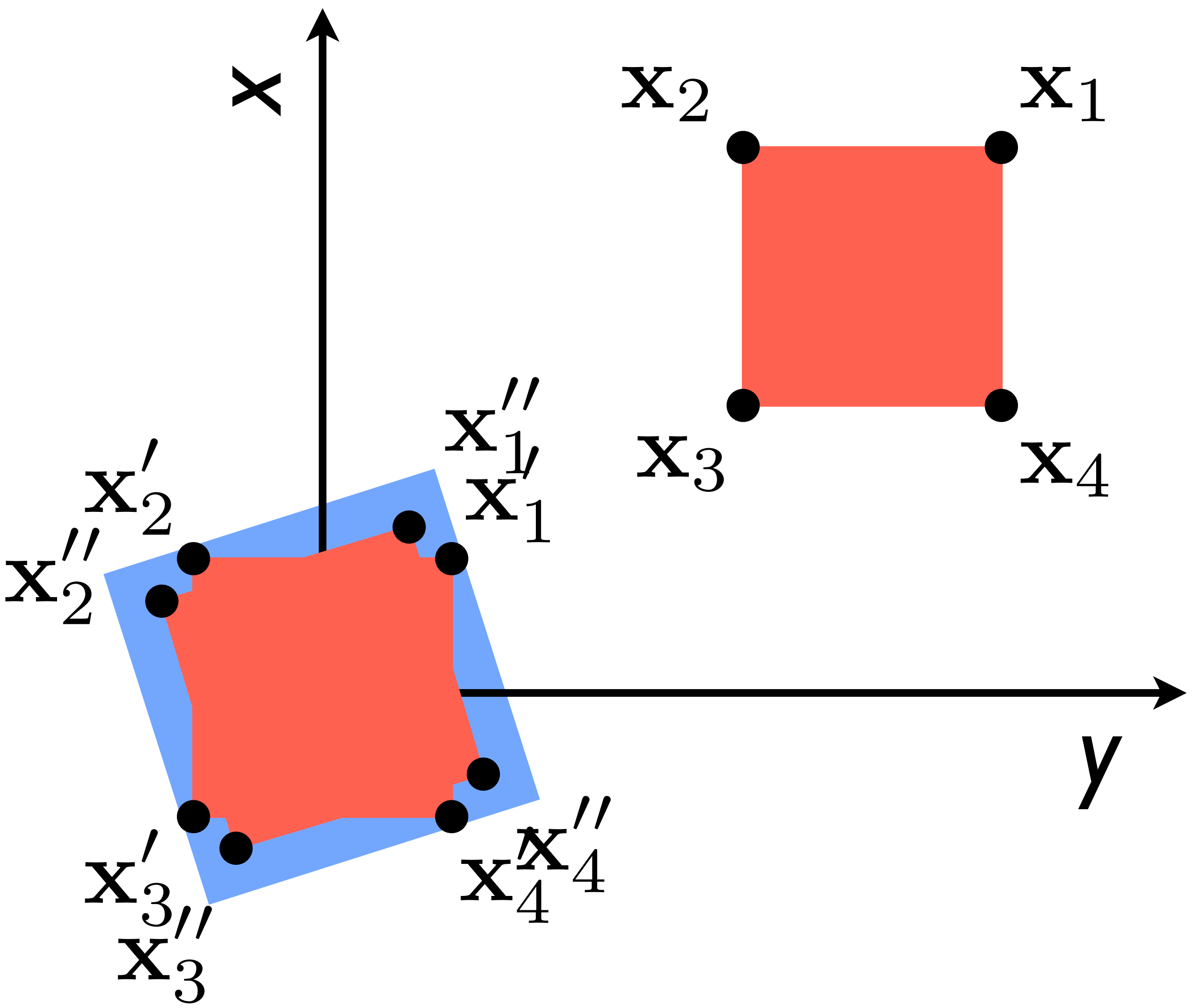
$$\mathbf{x}'_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} x'_1 \\ y'_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\hat{\mathbf{x}}'_1 = \mathbf{T}\hat{\mathbf{x}}_1$$

2D ROTATION



$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

$$\hat{\mathbf{x}}_1' = \mathbf{T}\hat{\mathbf{x}}_1$$

$$\begin{bmatrix} x_1'' \\ y_1'' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_1' \\ y_1' \end{bmatrix}$$

$$\begin{bmatrix} x_1'' \\ y_1'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix}$$

$$\hat{\mathbf{x}}_1'' = \mathbf{R}\hat{\mathbf{x}}_1'$$

$$\hat{\mathbf{x}}_1'' = \mathbf{R}\mathbf{T}\hat{\mathbf{x}}_1$$

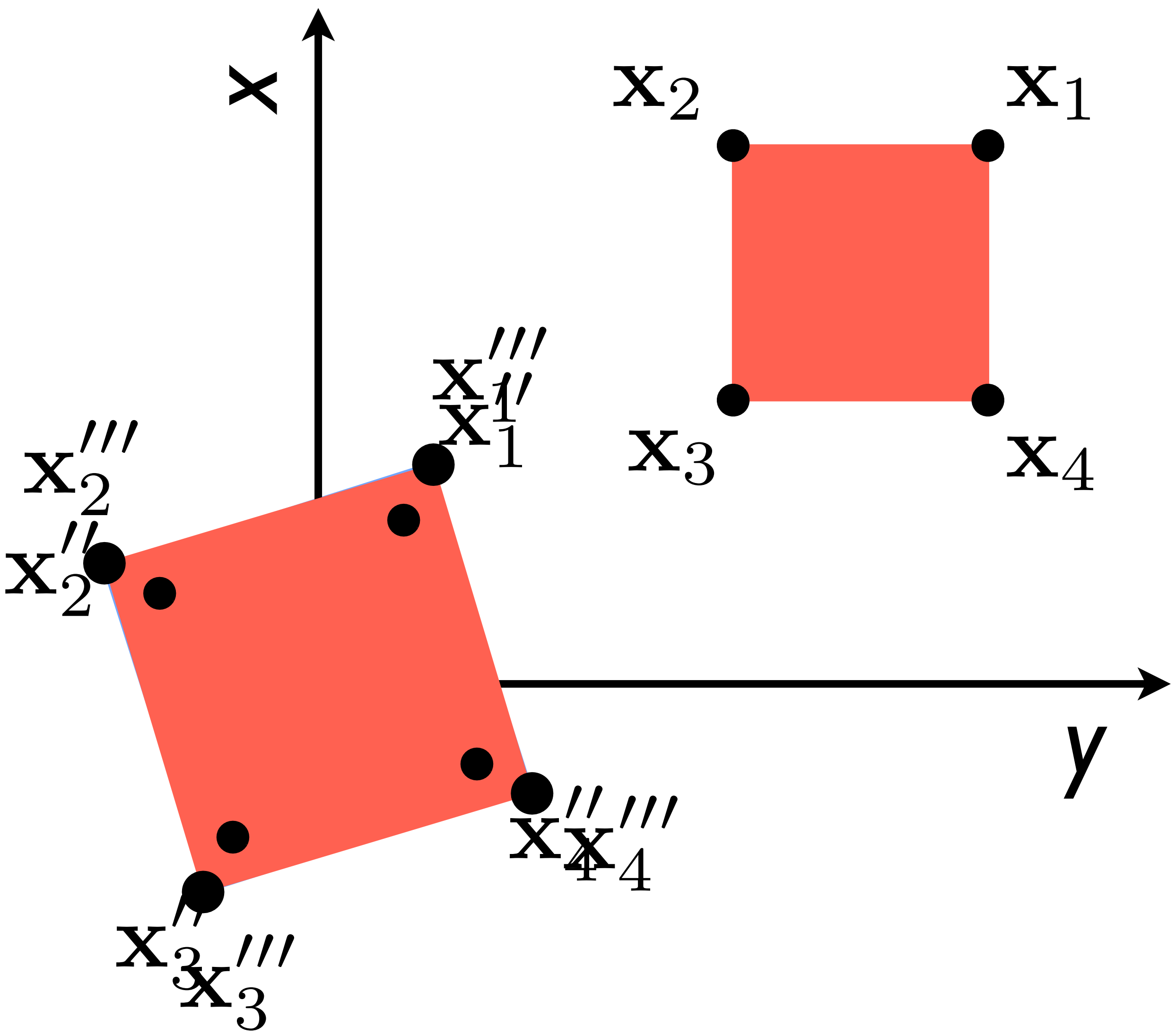
Rotations have to occur with respect to the origin. This example is a rotation about the origin. If you wish to rotate about an arbitrary point, you need three transformations: apply a translation taking the arbitrary point to the origin, apply the desired rotation, and finally apply a translation taking the point back to its original position.

ROTATION MATRICES ARE ORTHOGONAL

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}^T \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D SCALE



$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

$$\mathbf{x}_1' = \mathbf{T}\mathbf{x}_1$$

$$\mathbf{x}_1'' = \mathbf{R}\mathbf{T}\mathbf{x}_1$$

$$\begin{bmatrix} x_1''' \\ y_1''' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x_1'' \\ y_1'' \end{bmatrix}$$

$$\begin{bmatrix} x_1''' \\ y_1''' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1'' \\ y_1'' \\ 1 \end{bmatrix}$$

$$\mathbf{x}_1''' = \mathbf{S}\mathbf{R}\mathbf{T}\mathbf{x}_1$$

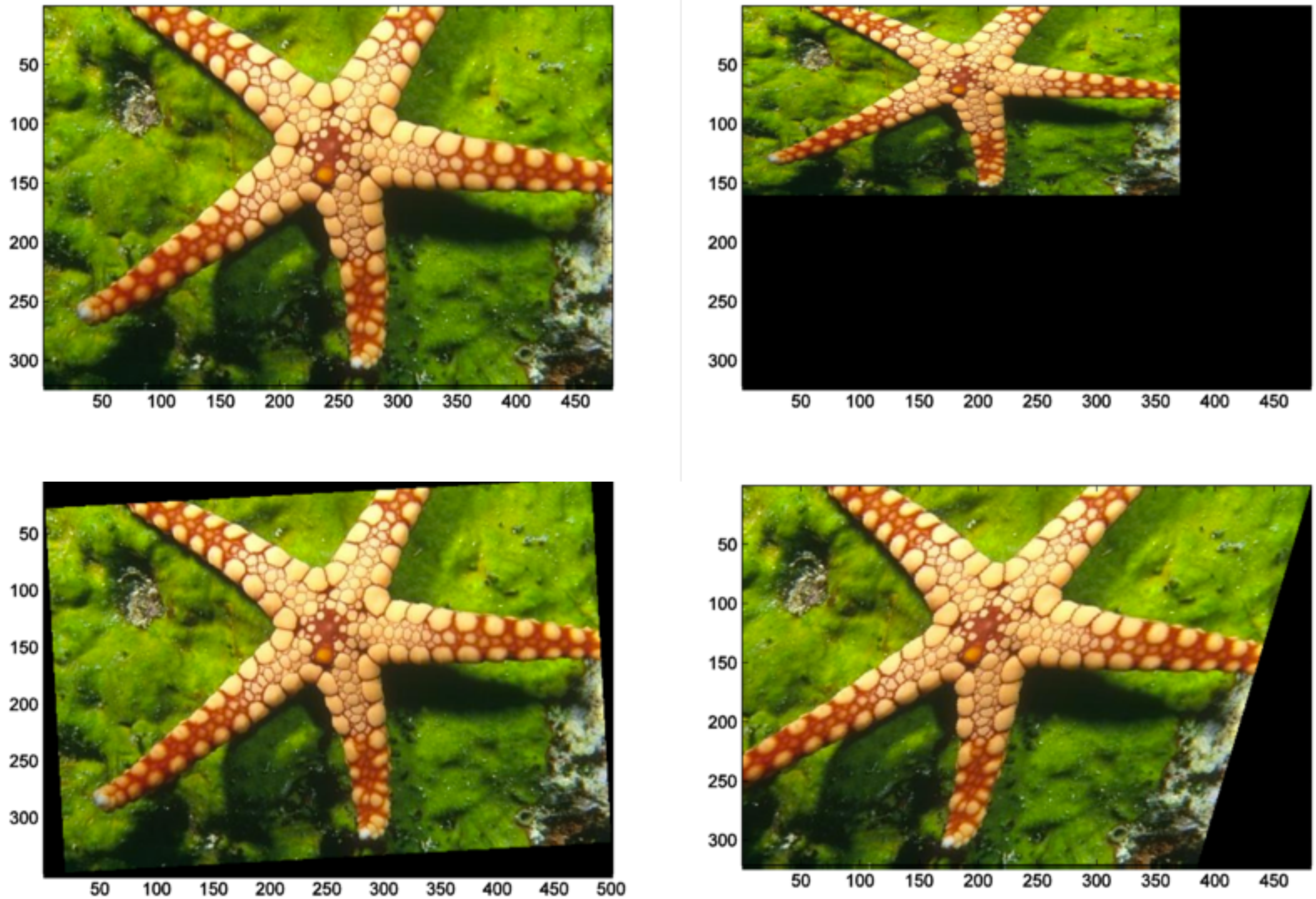
2D TRANSFORMATIONS

$$\mathbf{x}_1''' = \mathbf{SRT}\mathbf{x}_1$$

$$\begin{bmatrix} x_1''' \\ y_1''' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1''' \\ y_1''' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & S_y \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\mathbf{x}_1''' = \mathbf{A}\mathbf{x}_1$$


50 100 150 200 250 300
50 100 150 200 250 300 350 400 450
50 100 150 200 250 300
50 100 150 200 250 300 350 400 450
50 100 150 200 250 300
50 100 150 200 250 300 350 400 450 500
50 100 150 200 250 300
50 100 150 200 250 300 350 400 450

FORWARD WARPING ALGORITHM

- FOR EACH SOURCE PIXEL [*R*, *G*, *B*, *X*, *Y*]
    - APPLY TRANSFORM **A** TO *X*, *Y* TO GET *X'*, *Y'*
    - FOR THE TARGET ARRAY *X'*, *Y'* COPY *R*, *G*, *B*

BACKWARD WARPING ALGORITHM

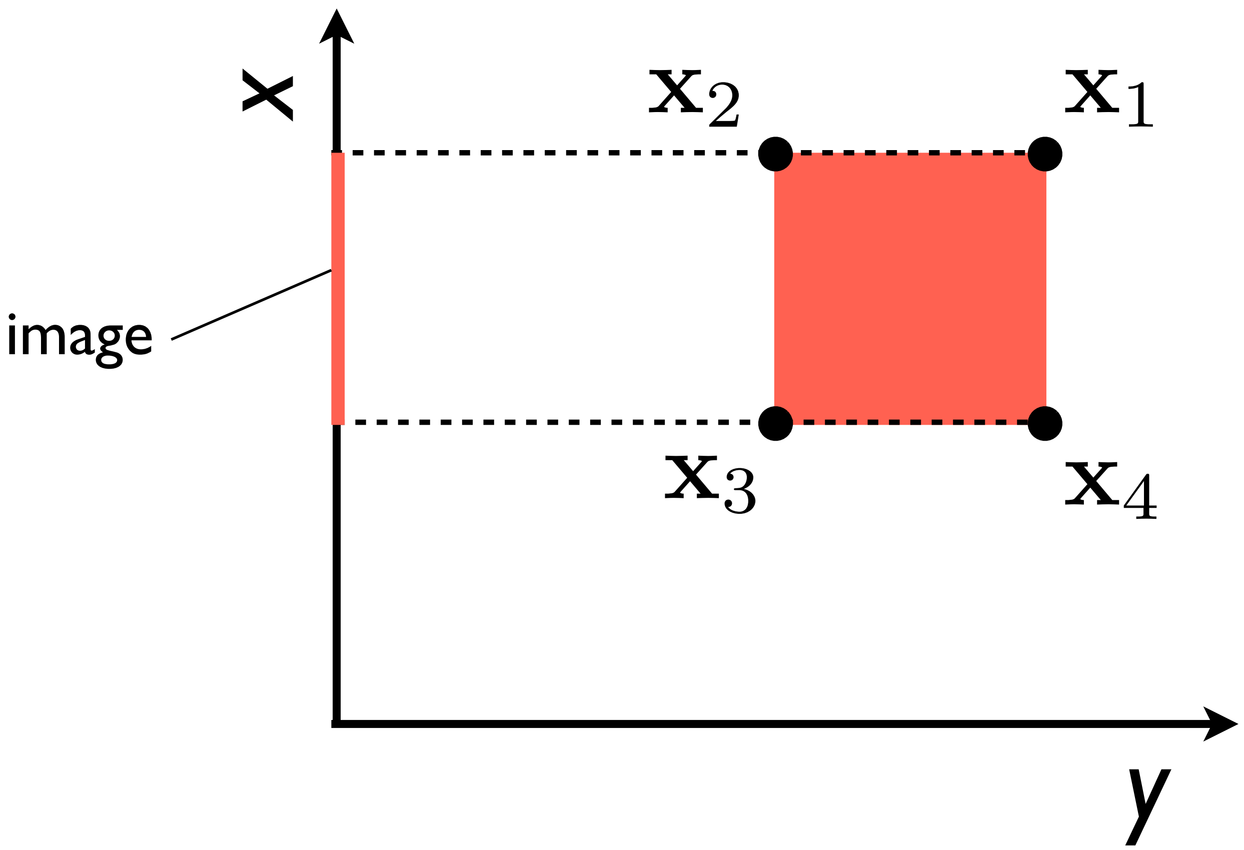
- FOR EACH TARGET PIXEL [*? ?, ?, X', Y'*]
  - APPLY TRANSFORM $\mathbf{A}^{-1}$ TO *X', Y'* TO GET *X, Y*
  - FOR THE TARGET ARRAY X', Y' COPY R, G, B

AFFINE TRANSFORMS

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

PROJECTION
3D to 2D

PROJECTION: 2D TO 1D

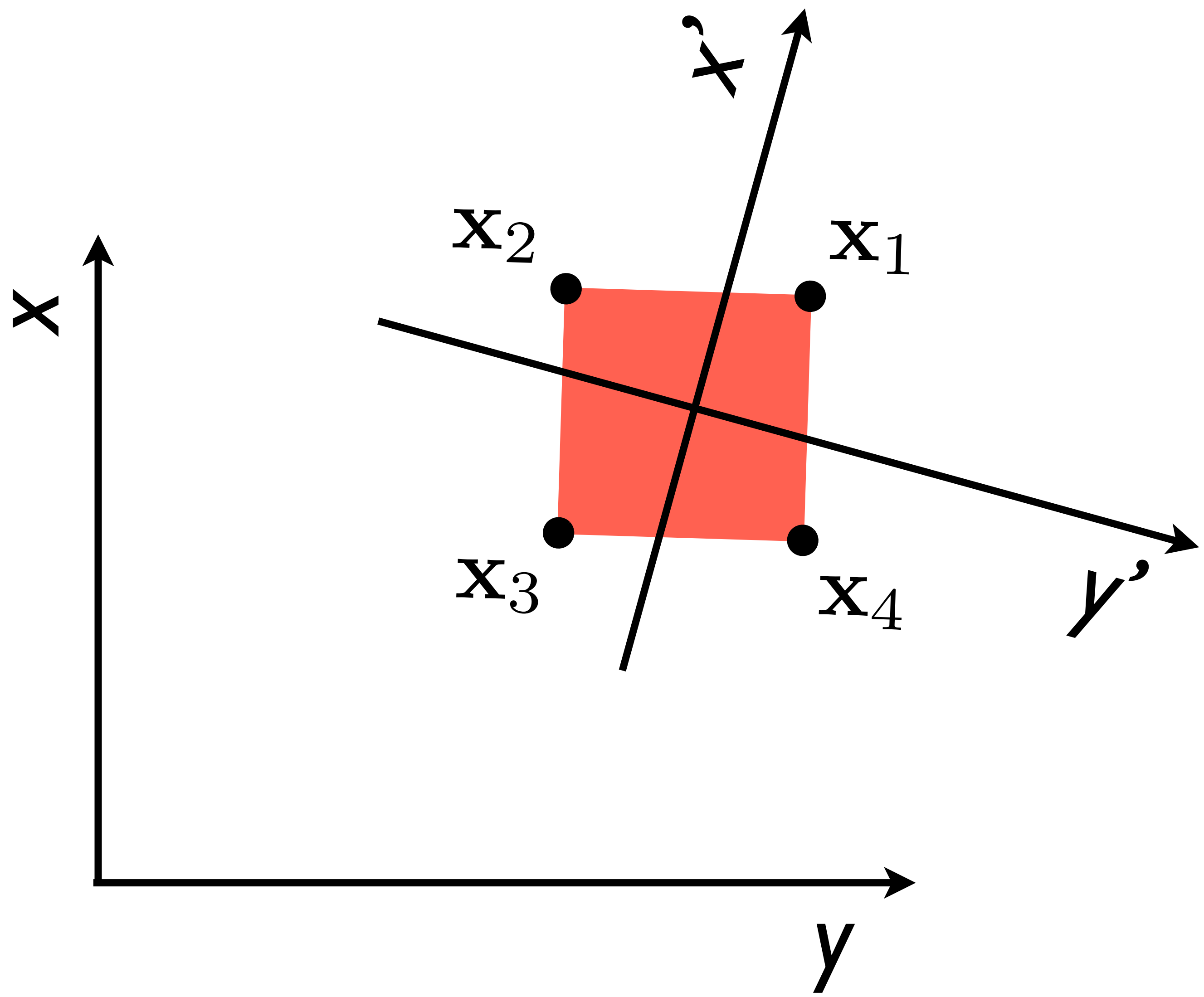


$$u_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$u_1 = \mathbf{K}\hat{\mathbf{x}}_1$$

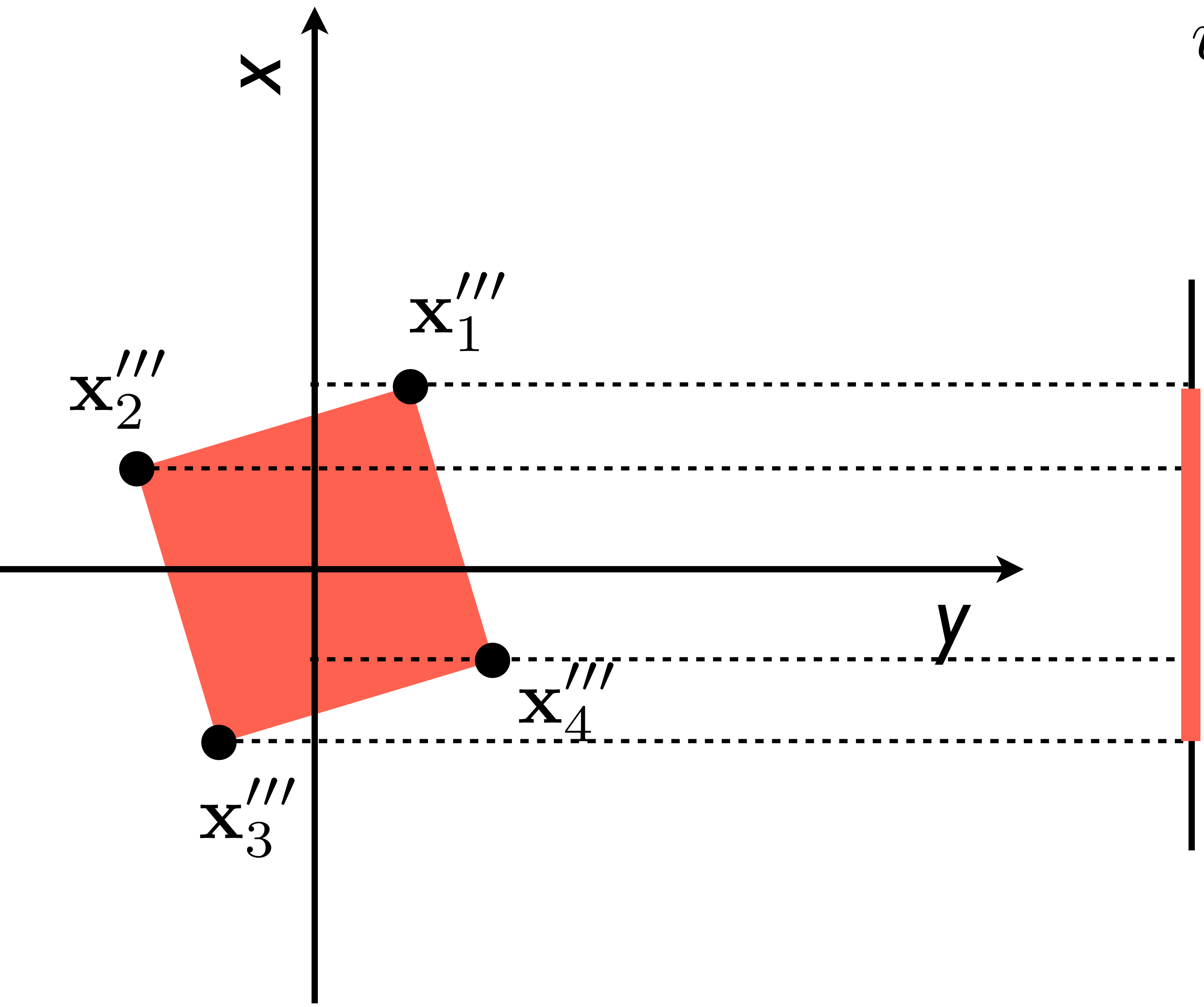
2 dimensional to 1 dimensional projection

PROJECTION: 2D TO 1D



How do we project to an arbitrary 1d 'camera'?

PROJECTION: 2D TO 1D

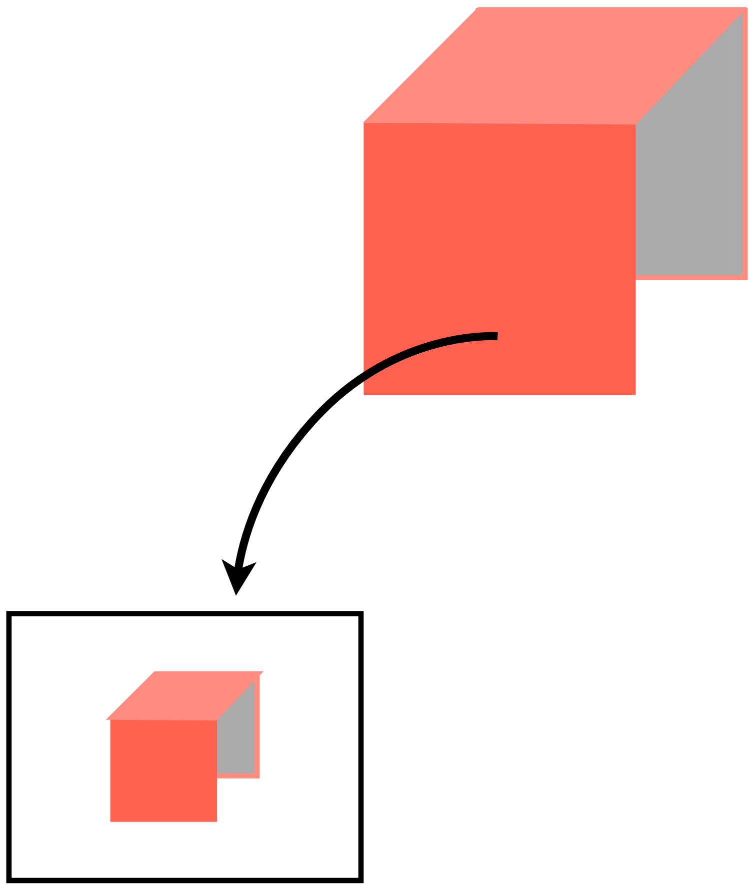


$$u_1''' = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1''' \\ y_1''' \\ 1 \end{bmatrix}$$

$$u_1''' = \mathbf{K}\hat{\mathbf{x}}_1'''$$

$$u_1''' = \mathbf{KSRT}\hat{\mathbf{x}}_1$$

PROJECTION: 3D TO 2D



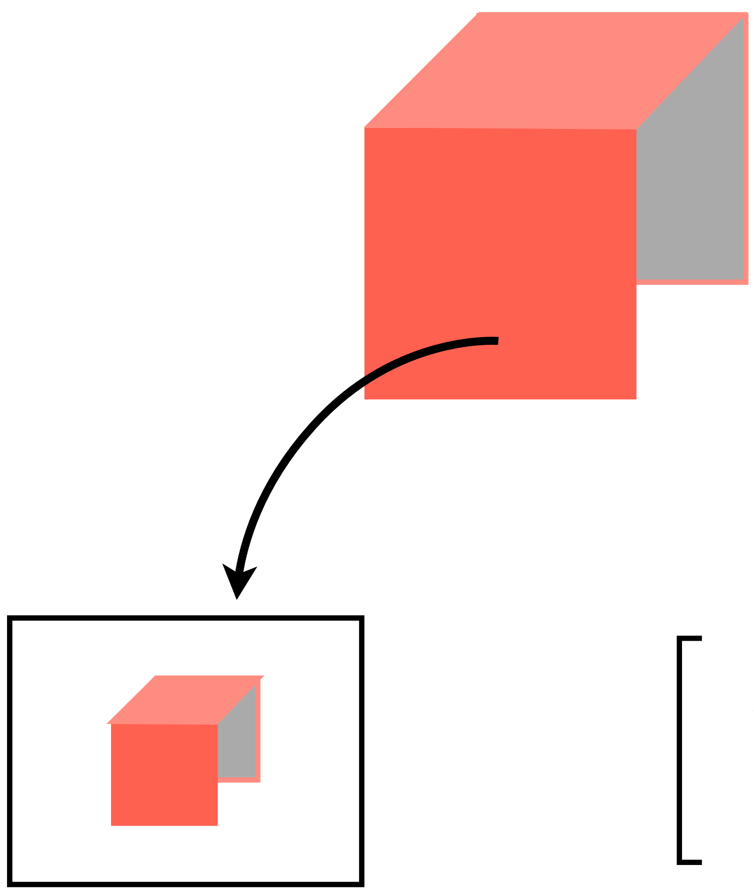
$$\hat{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\hat{\mathbf{X}} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$u_1''' = \mathbf{K}_{1\times 3}\mathbf{S}_{3\times 3}\mathbf{R}_{3\times 3}\mathbf{T}_{3\times 3}\hat{\mathbf{x}}_1$$

$$\mathbf{x}_1''' = \mathbf{K}_{2\times 4}\mathbf{S}_{4\times 4}\mathbf{R}_{4\times 4}\mathbf{T}_{4\times 4}\hat{\mathbf{X}}_1$$

PROJECTION: 3D TO 2D

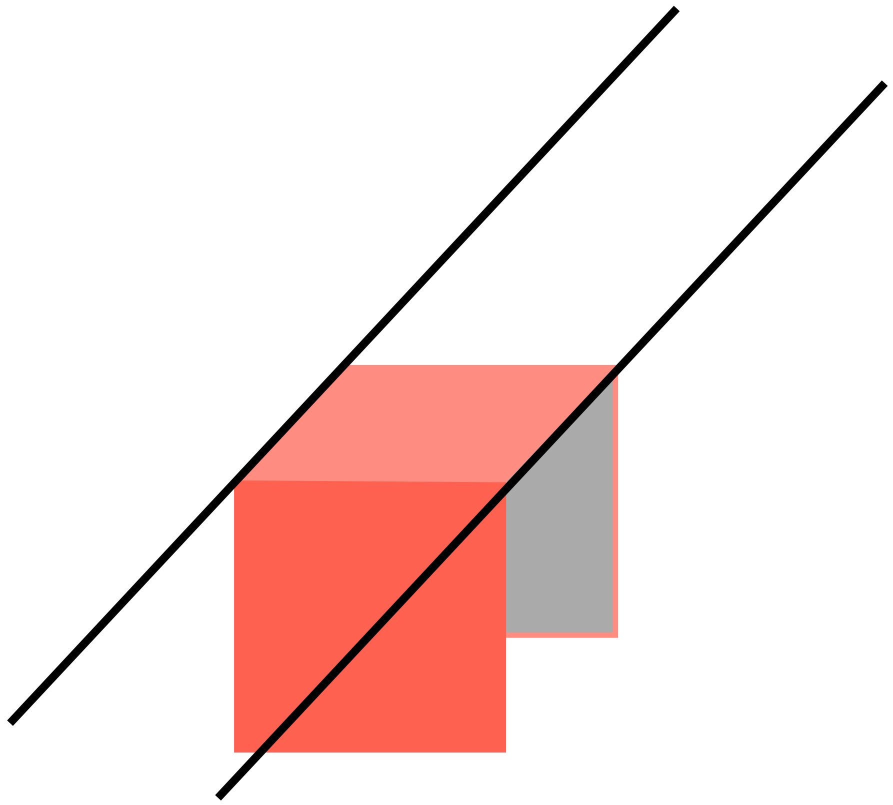


$$\mathbf{x}_1''' = \mathbf{K}_{2\times 4}\mathbf{S}_{4\times 4}\mathbf{R}_{4\times 4}\mathbf{T}_{4\times 4}\hat{\mathbf{X}}_1$$

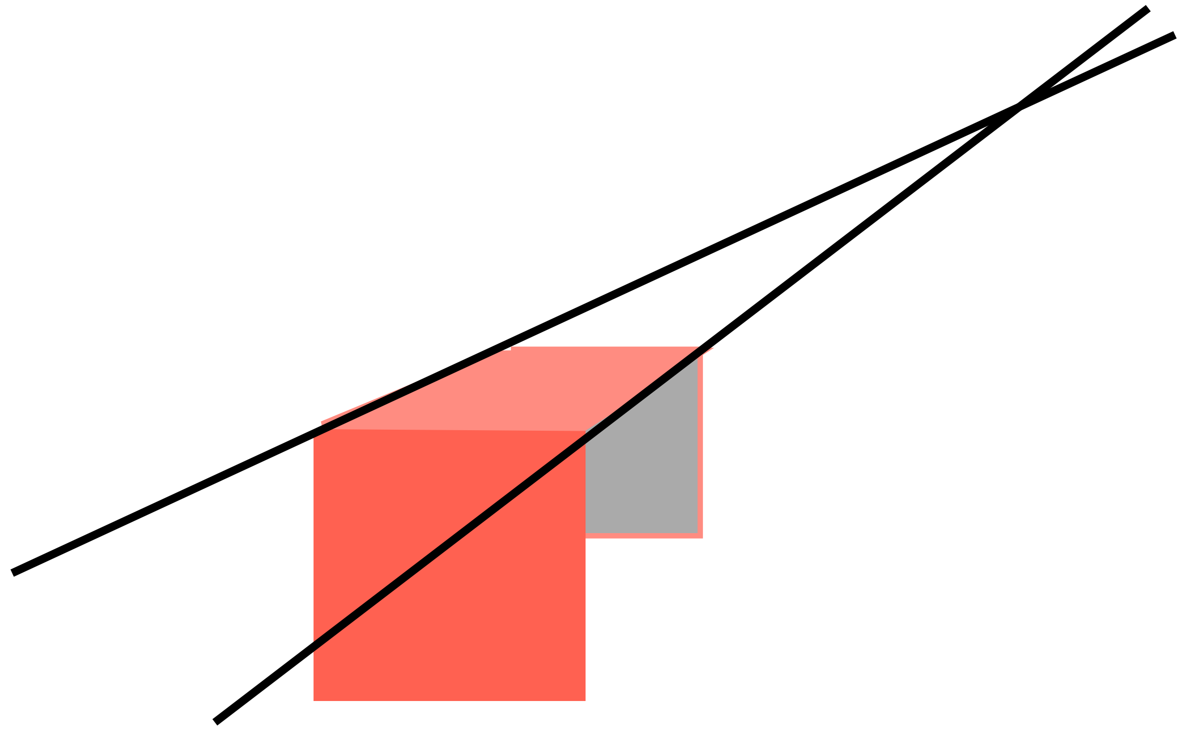
$$\mathbf{x} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{bmatrix} \mathbf{X}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

ORTHOGRAPHIC VS PERSPECTIVE

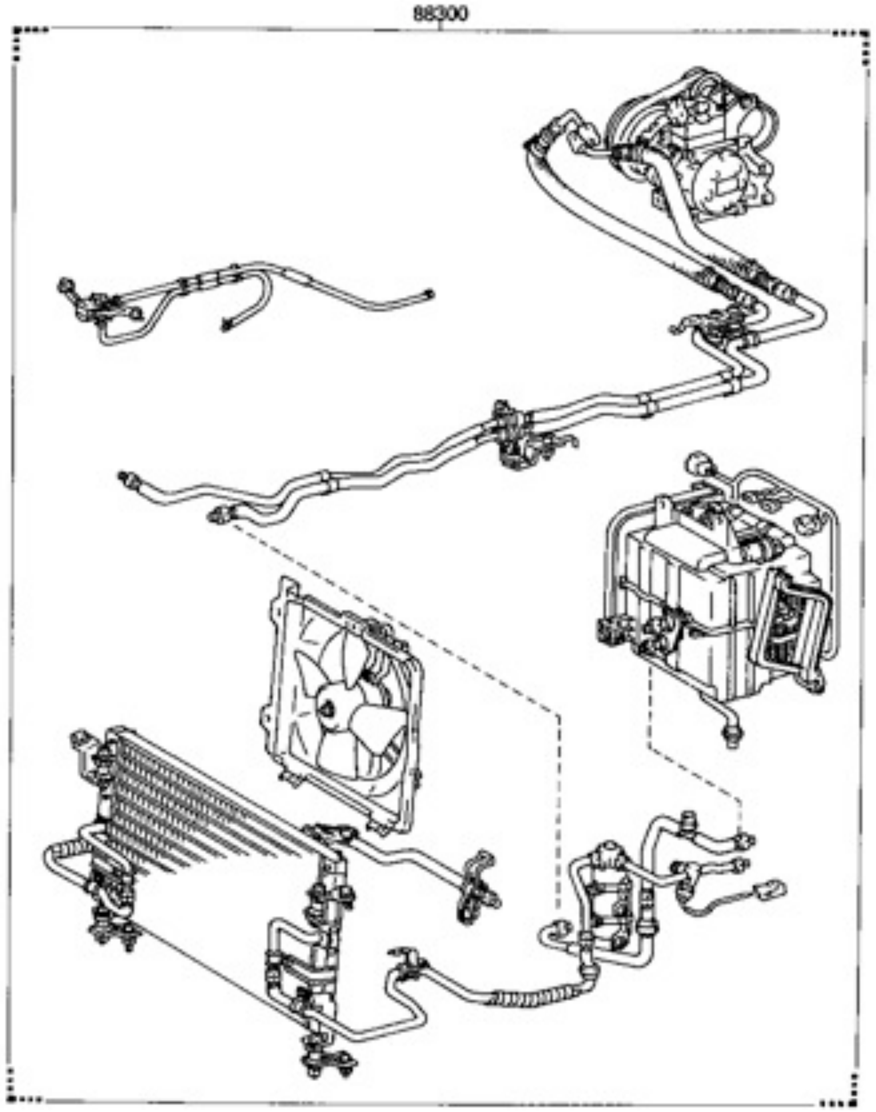


parallel projection



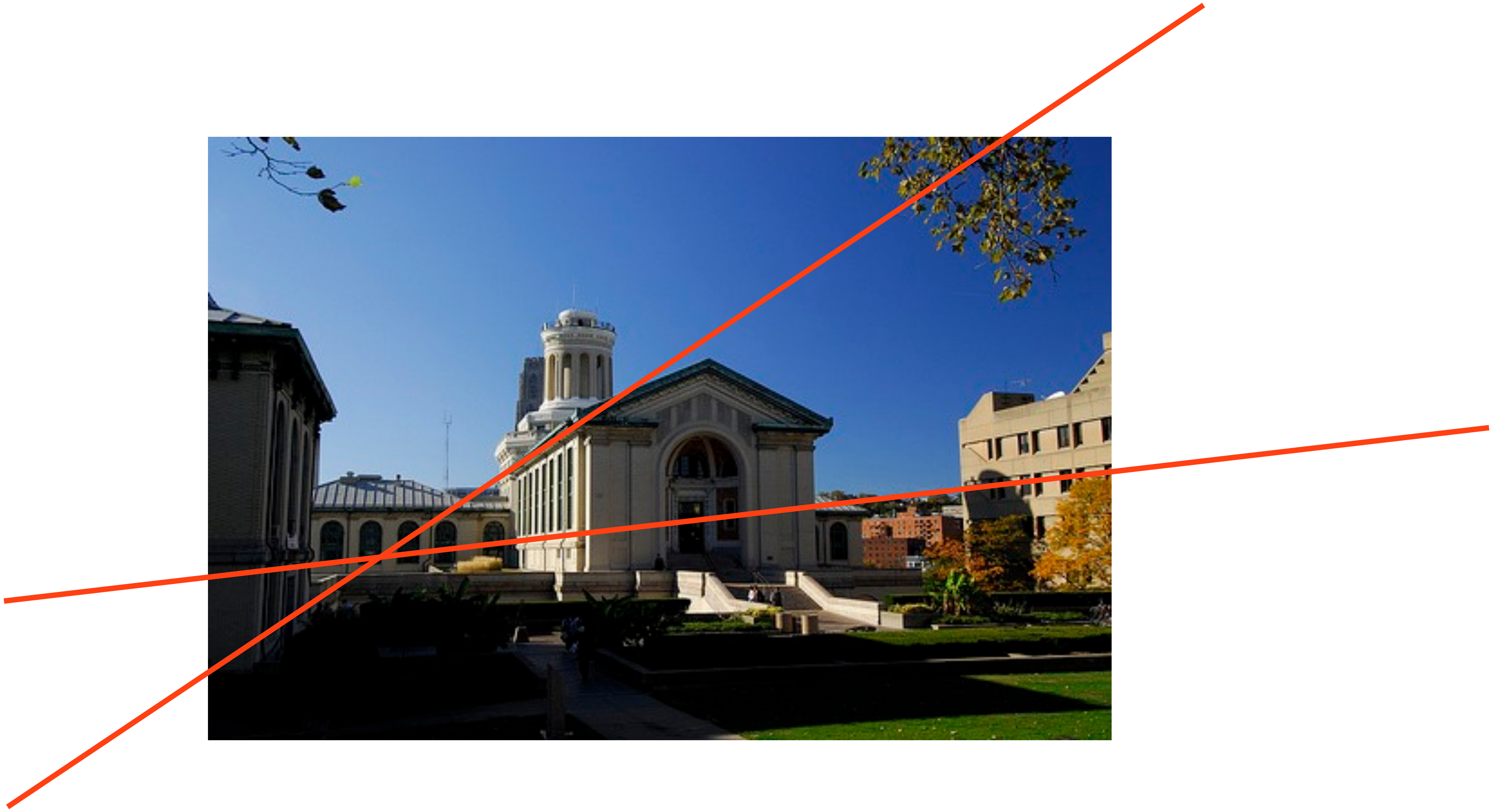
central projection

ORTHOGRAPHIC PROJECTION



REAL IMAGES

REAL IMAGES



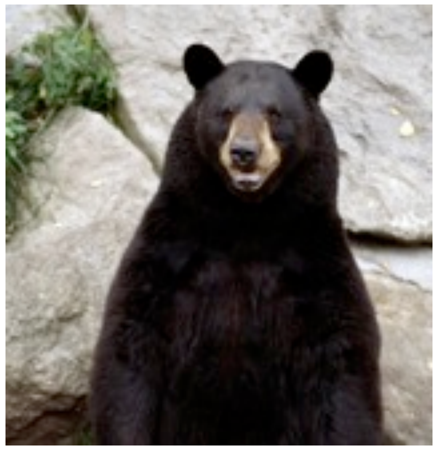
FORCED PERSPECTIVE

ANOTHER PERSPECTIVE...

HOW DO WE RECOVER 3D?

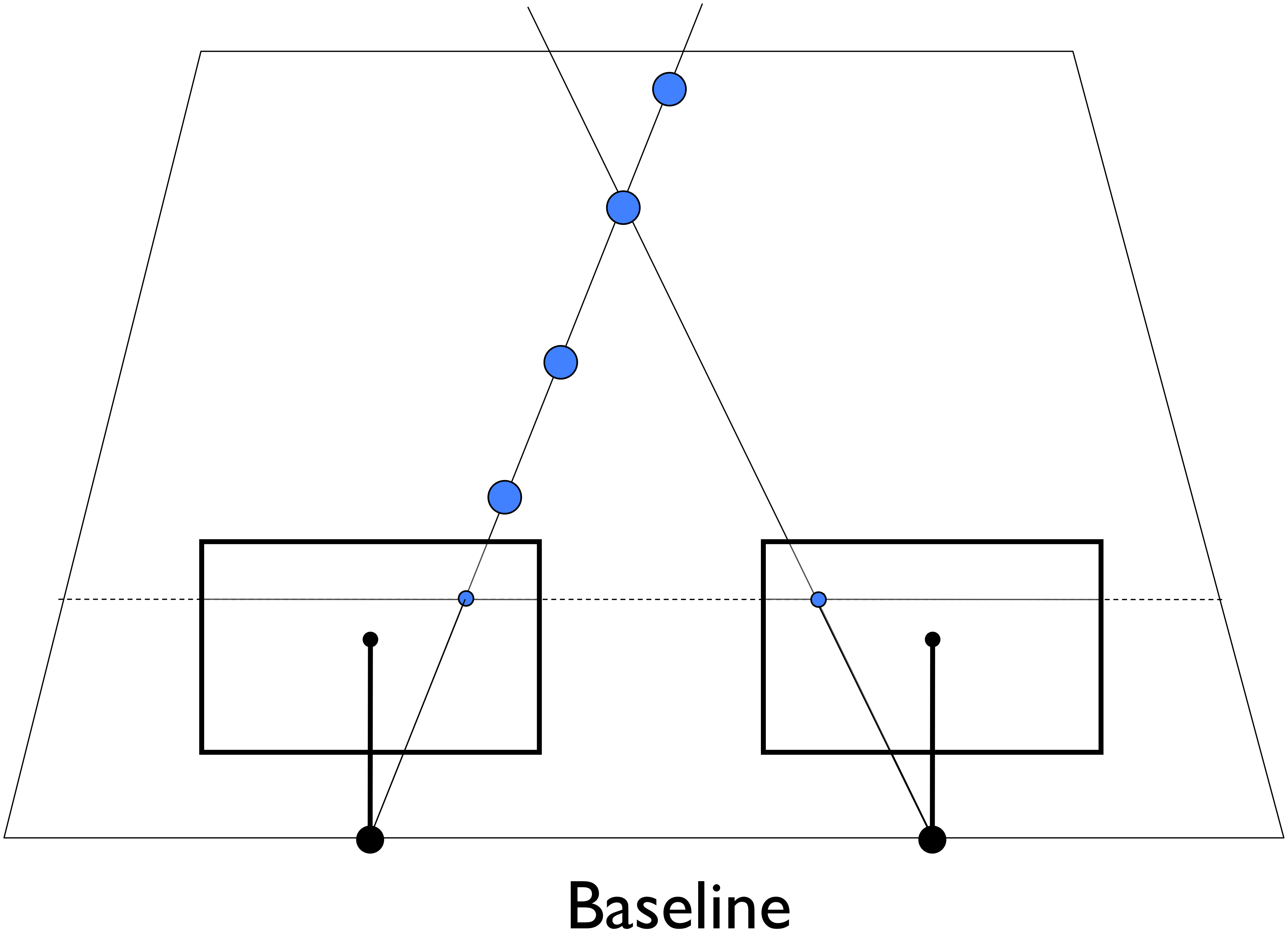








EPIPOLAR GEOMETRY

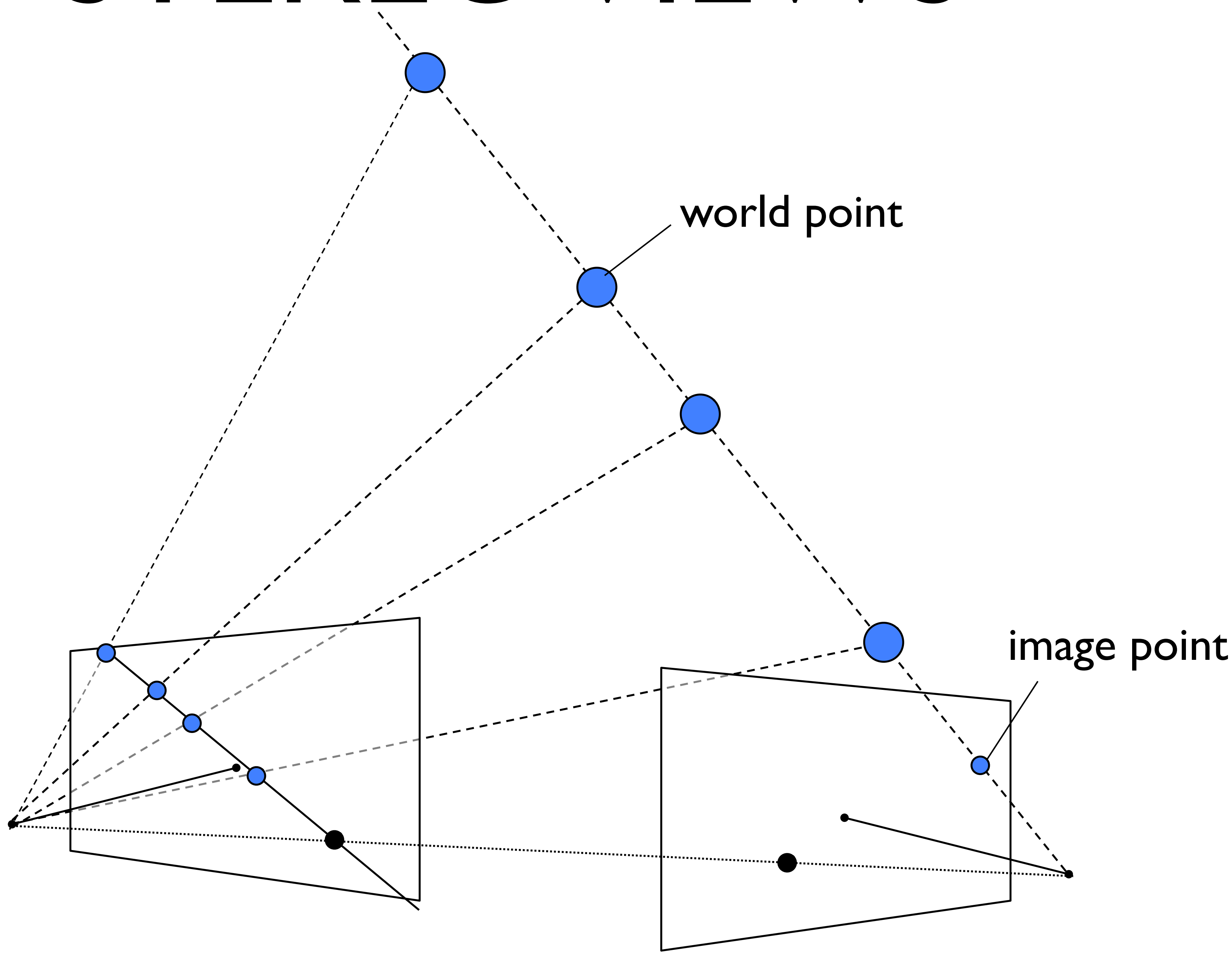


STEREOPSIS

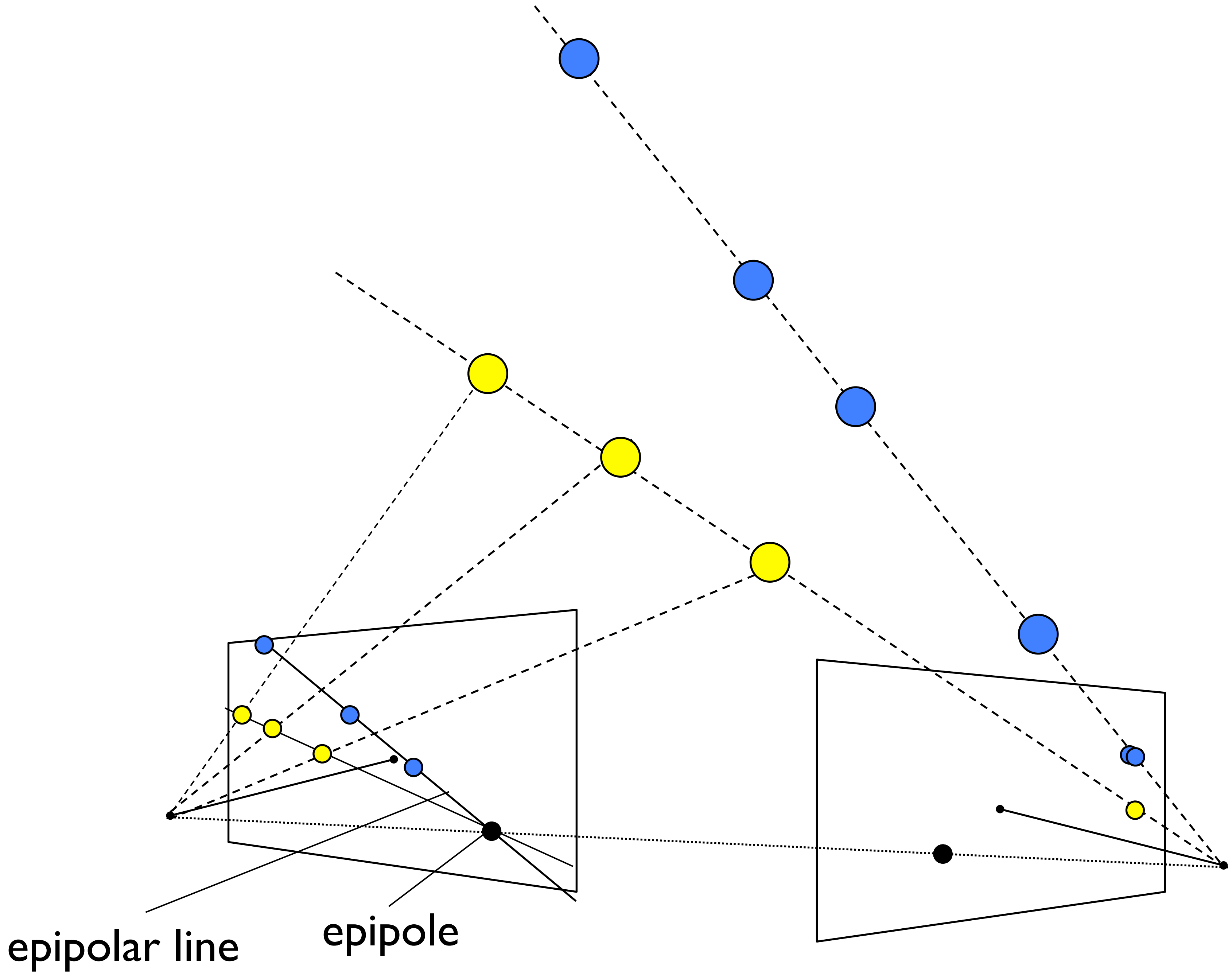
- **CORRESPONDENCE:** FINDING THE IMAGE OF A 3D POINT IN BOTH IMAGES
- **RECONSTRUCTION:** RECOVERING THE LOCATION OF THE 3D POINT

STEREO VIEWS

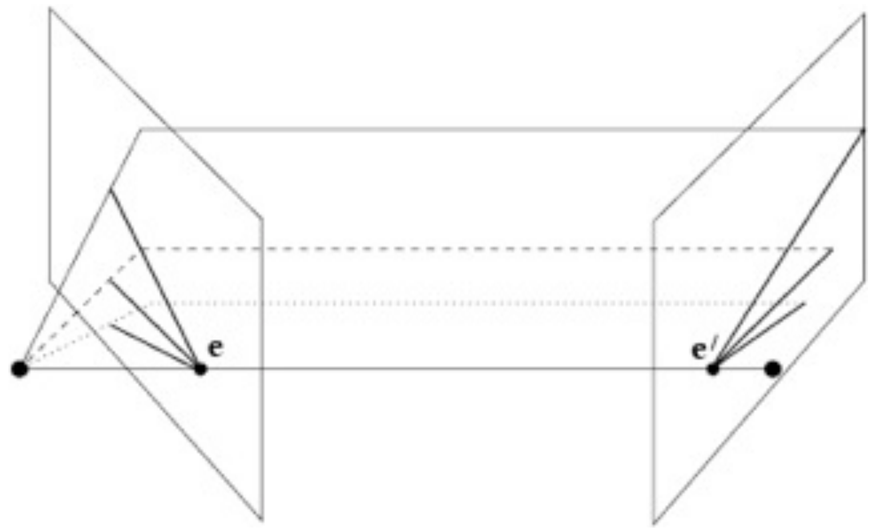
STEREO VIEWS

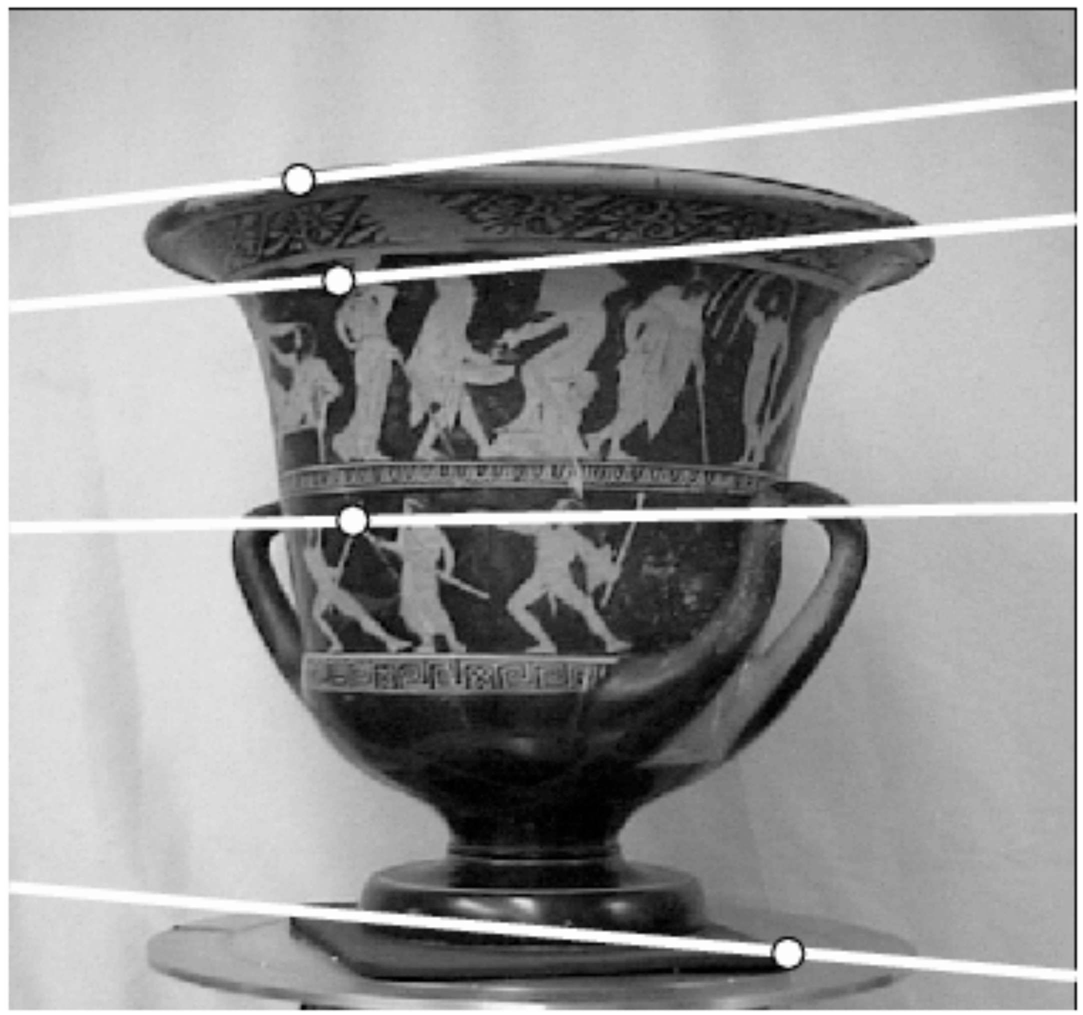


STEREO VIEWS

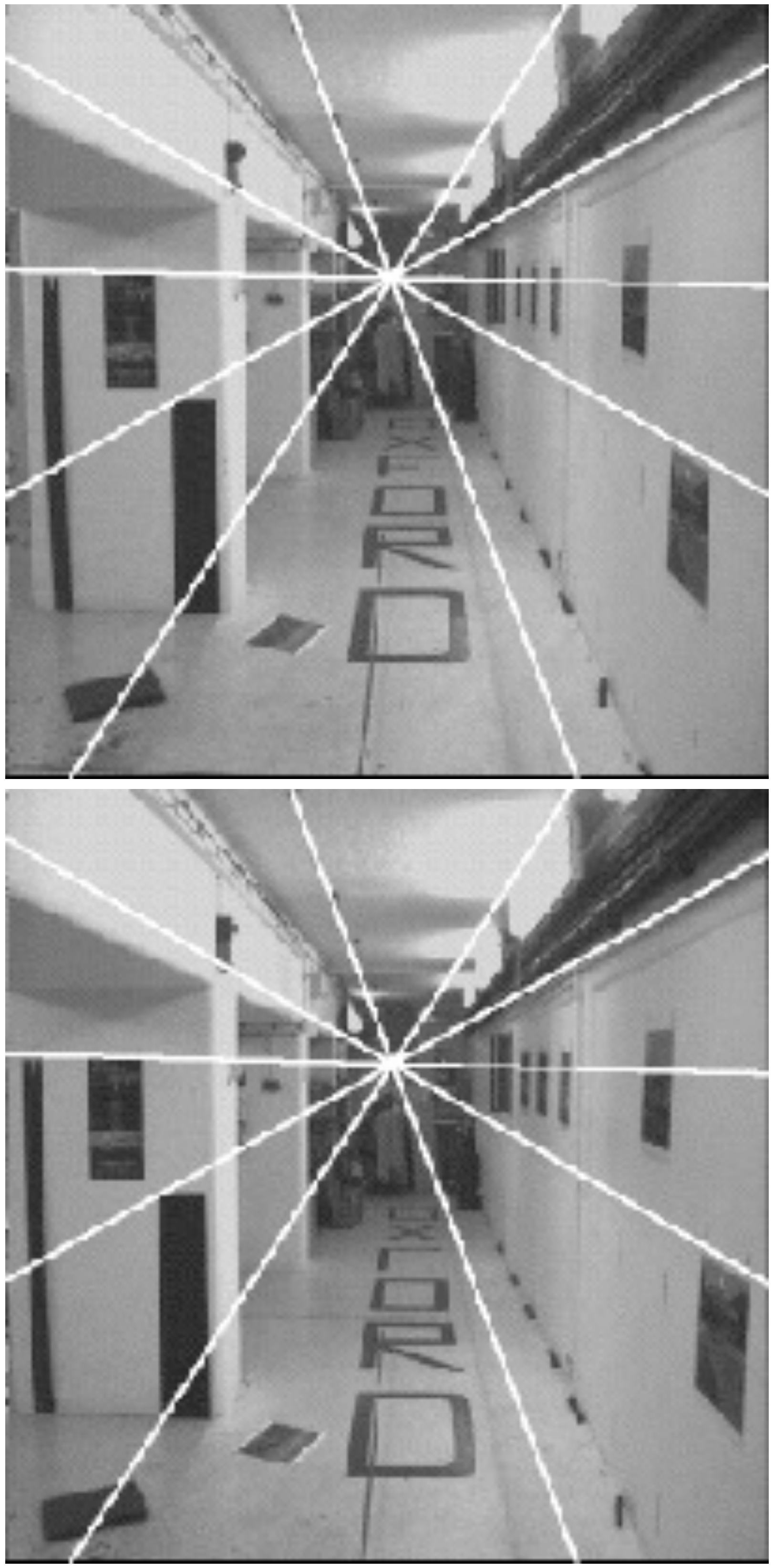


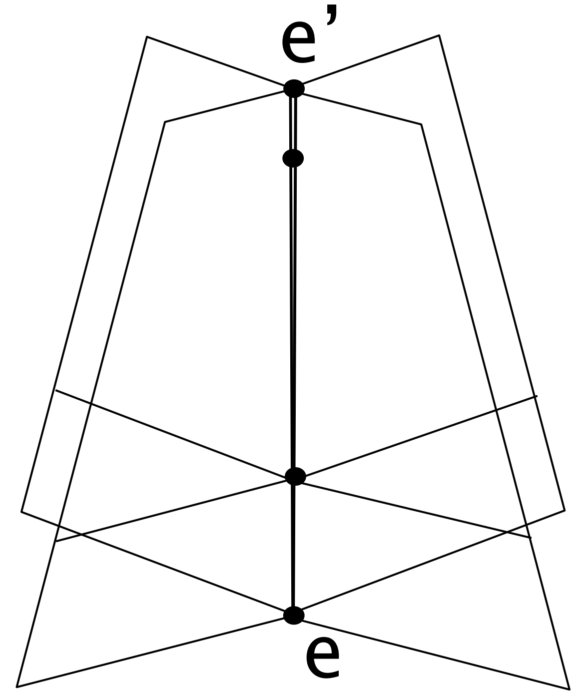
Example 1





Example 2



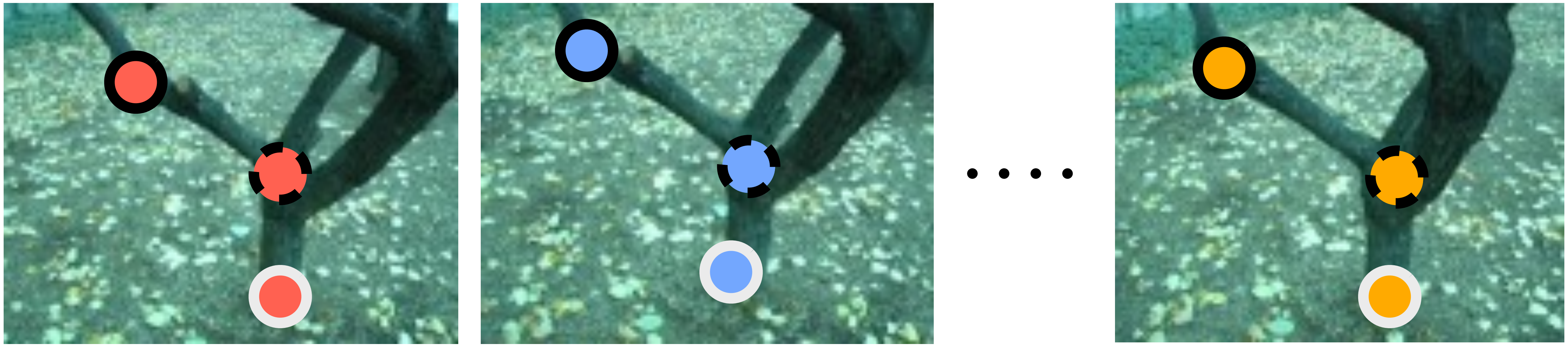


MULTIPLE VIEWS?

STRUCTURE FROM MOTION

MULTIPLE VIEWS

STRUCTURE FROM MOTION



1 2 ···· *F*

$$\mathbf{W} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1P} \\ y_{11} & y_{12} & \cdots & y_{1P} \\ x_{21} & x_{22} & \cdots & x_{2P} \\ y_{21} & y_{22} & \cdots & y_{2P} \\ \vdots & & & \vdots \\ x_{F1} & x_{F2} & \cdots & x_{FP} \\ y_{F1} & y_{F2} & \cdots & y_{FP} \end{bmatrix}$$

2xF (*F* frames)

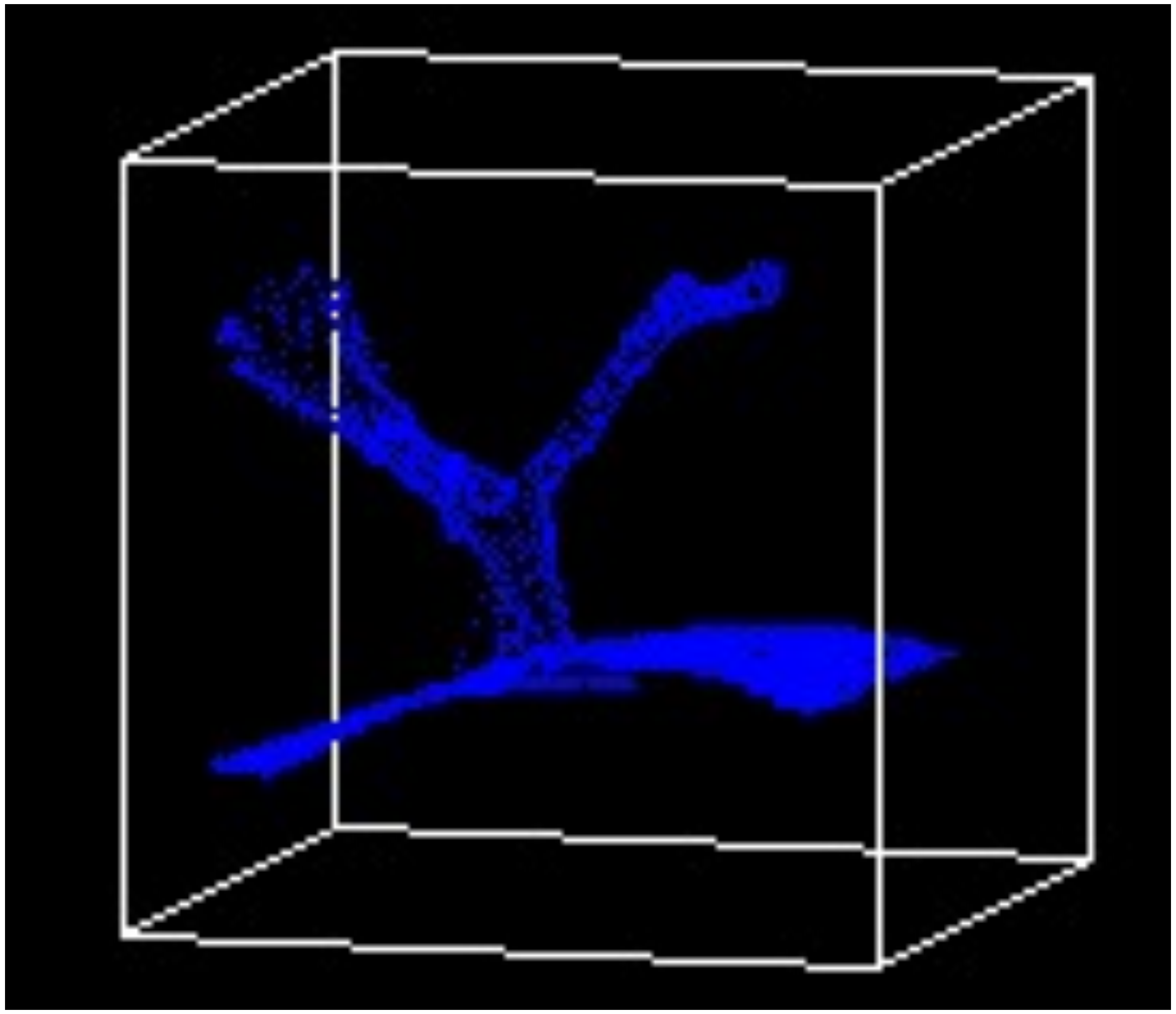
P (*P* points)

ALGORITHM

1. Input **W** matrix of tracks (*F* frames and *P* points)
2. Perform SVD of W: [**U**,**D**,**V**] = SVD(**W**)
3. Compute camera motion as **R’** = **UD**’, where **D** is a submatrix of **D**
4. Compute 3D structure as **S’** = **V’**^{T} where **V’**^{T} is a submatrix of **V**^{T}
5. <u>Compute metric upgrade using orthonormality constraints</u>

STRUCTURE FROM MOTION

STRUCTURE



$$\mathbf{S} = \begin{bmatrix} X_1 & X_2 & \dots & X_P \\ Y_1 & Y_2 & & Y_P \\ Z_1 & Z_2 & & Z_P \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 & \dots & \mathbf{S}_P \end{bmatrix}$$

$$\mathbf{S}_1 = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}$$

STRUCTURE FROM MOTION

CAMERA MOTION

$$\mathbf{R}_1 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_F \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}$$

$$\mathbf{x}_1 = \mathbf{R}_1 \mathbf{S}_1$$

STRUCTURE FROM MOTION

CAMERA MOTION

$$\begin{bmatrix} \mathbf{x}_{11} \\ \mathbf{x}_{21} \\ \vdots \\ \mathbf{x}_{F1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_F \end{bmatrix} \mathbf{S}_1$$

one point seen in many cameras

$$\begin{bmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \dots & \mathbf{x}_{1P} \end{bmatrix} = \mathbf{R}_1 \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 & \dots & \mathbf{S}_F \end{bmatrix}$$

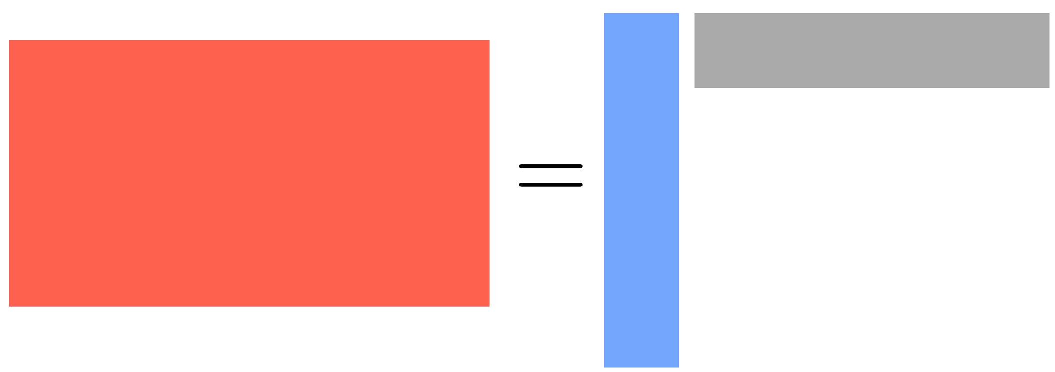
many points seen in one camera

STRUCTURE FROM MOTION

FACTORIZATION

$$\begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_F \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 & \dots & \mathbf{S}_F \end{bmatrix}$$

rank 4 $\longrightarrow$



RANK CONSTRAINT

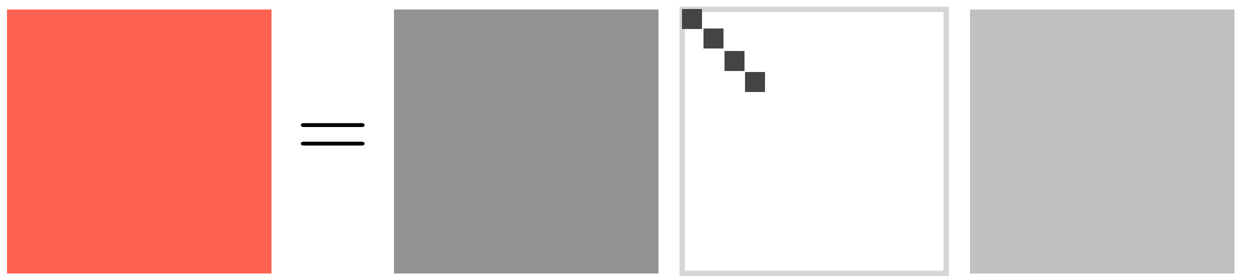
- RANK(**W**) = 4
  - IN THE ORIGINAL PAPER, A RANK 3 CONSTRAINT IS DESCRIBED
- HOW DO WE USE THIS PROPERTY TO ESTIMATE CAMERA MOTION AND STRUCTURE?

LINEAR ALGEBRA PRIMER

SINGULAR VALUE DECOMPOSITION

Any *m* x *n* matrix **A** can be written as a product of three matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

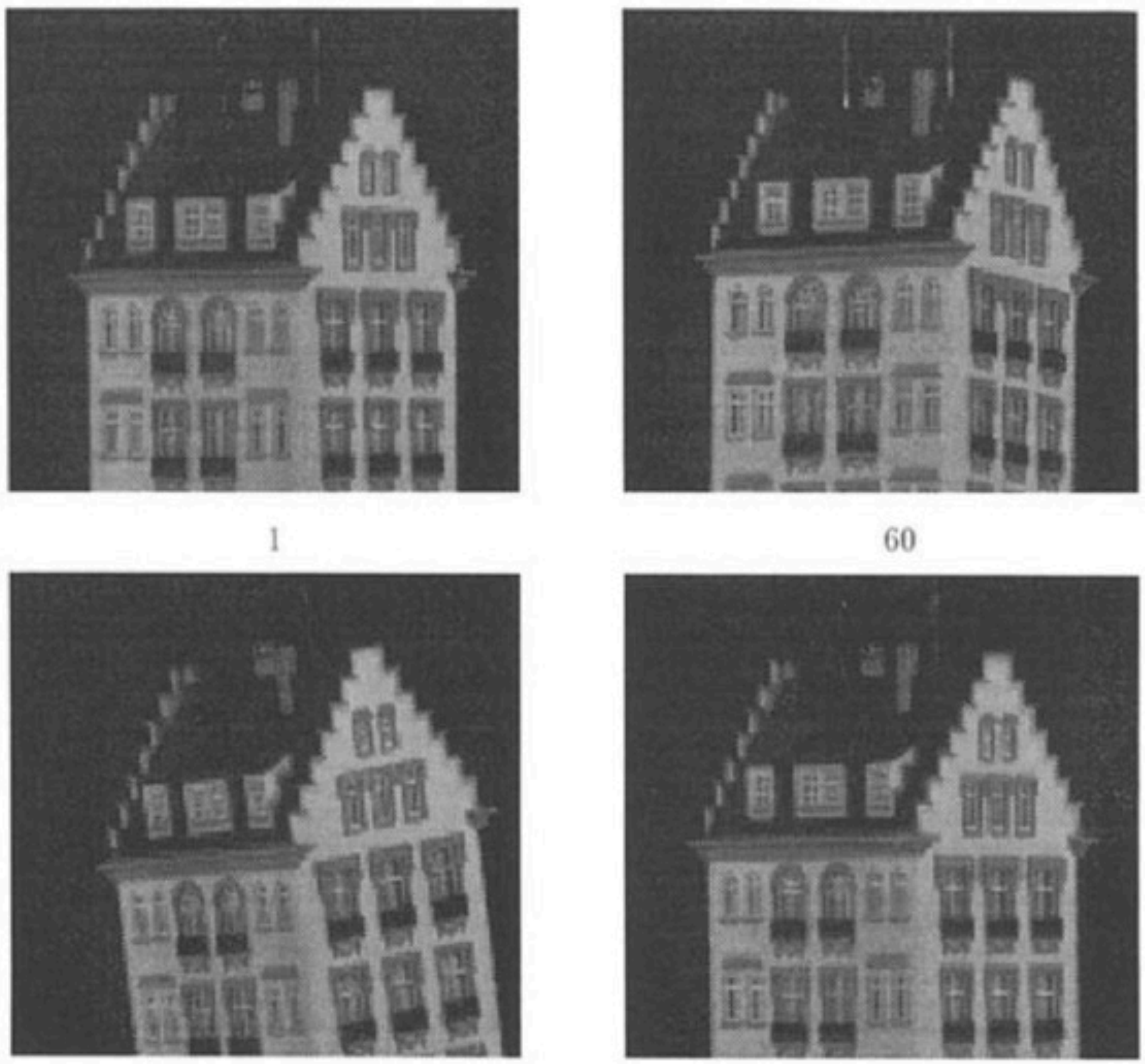


Rank 4 means the diagonal has only two non-zero elements

ALGORITHM

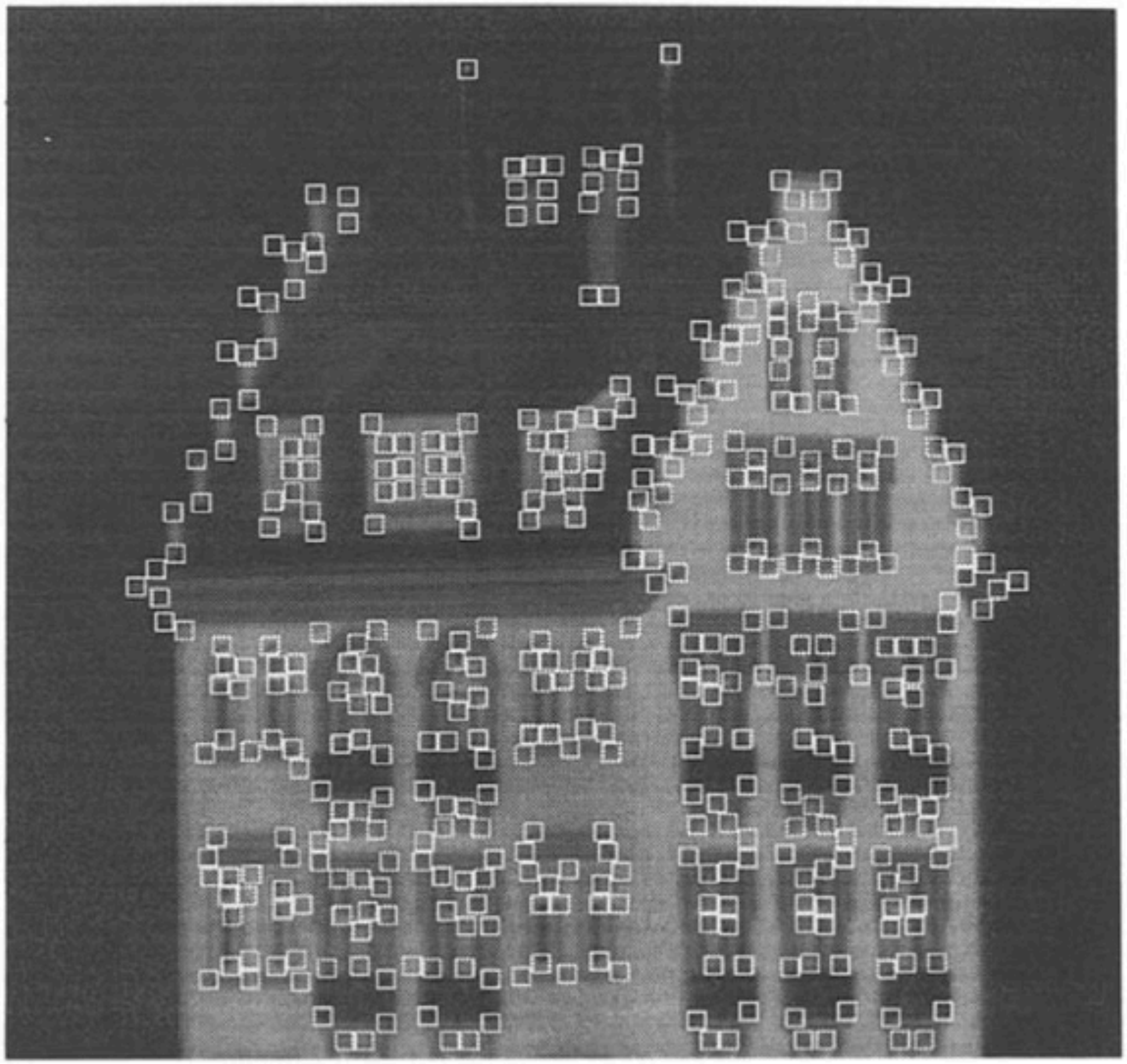
1. Input **W** matrix of tracks (*F* frames and *P* points)
2. Perform SVD of W: [**U**,**D**,**V**] = SVD(**W**)
3. Compute camera motion as **R'** = **UD**', where **D** is a submatrix of **D**
4. Compute 3D structure as **S'** = **V'**^{T} where **V'**^{T} is a submatrix of **V**^{T}
5. Compute metric upgrade using orthonormality constraints

RESULTS

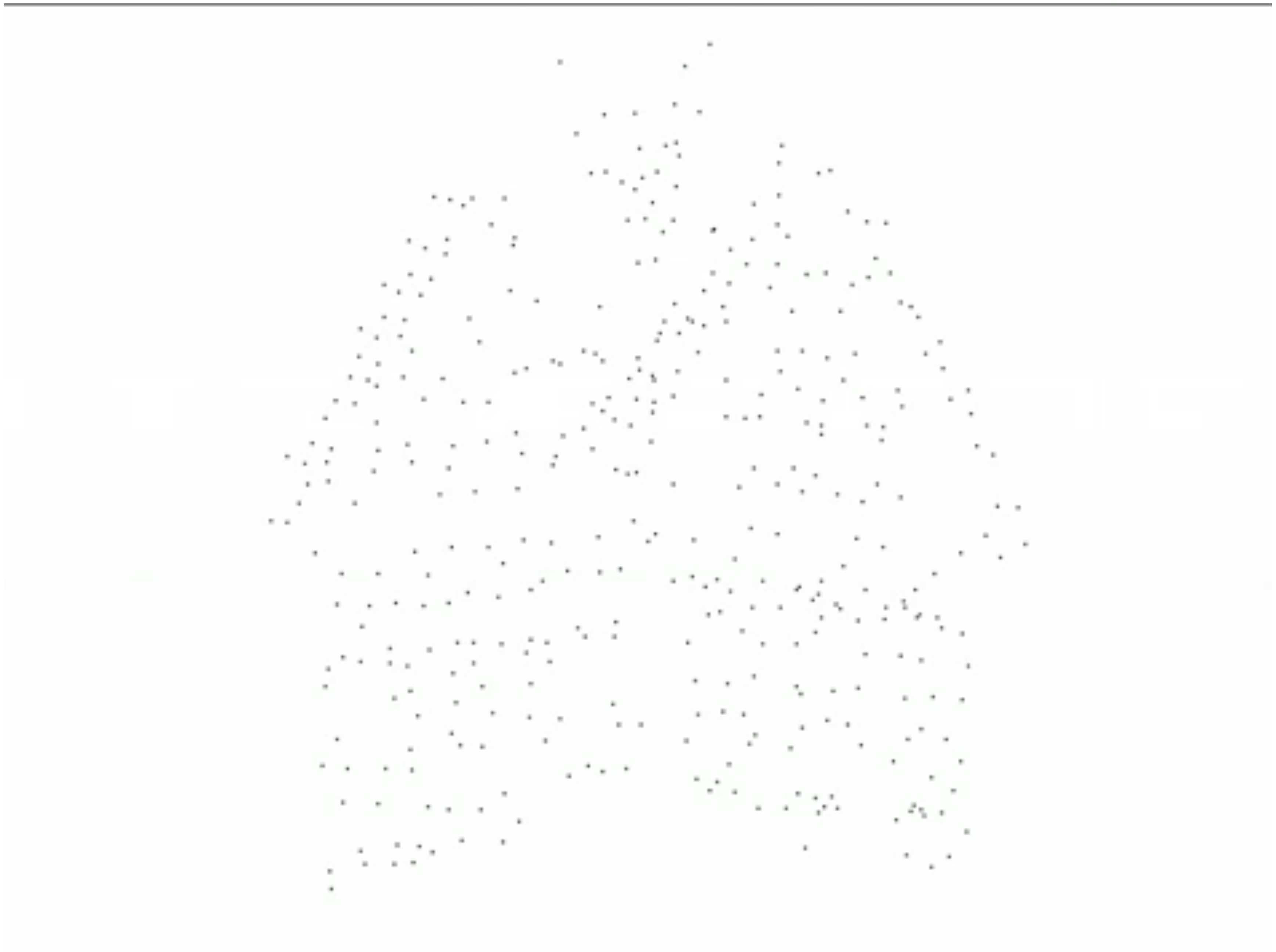


RESULTS

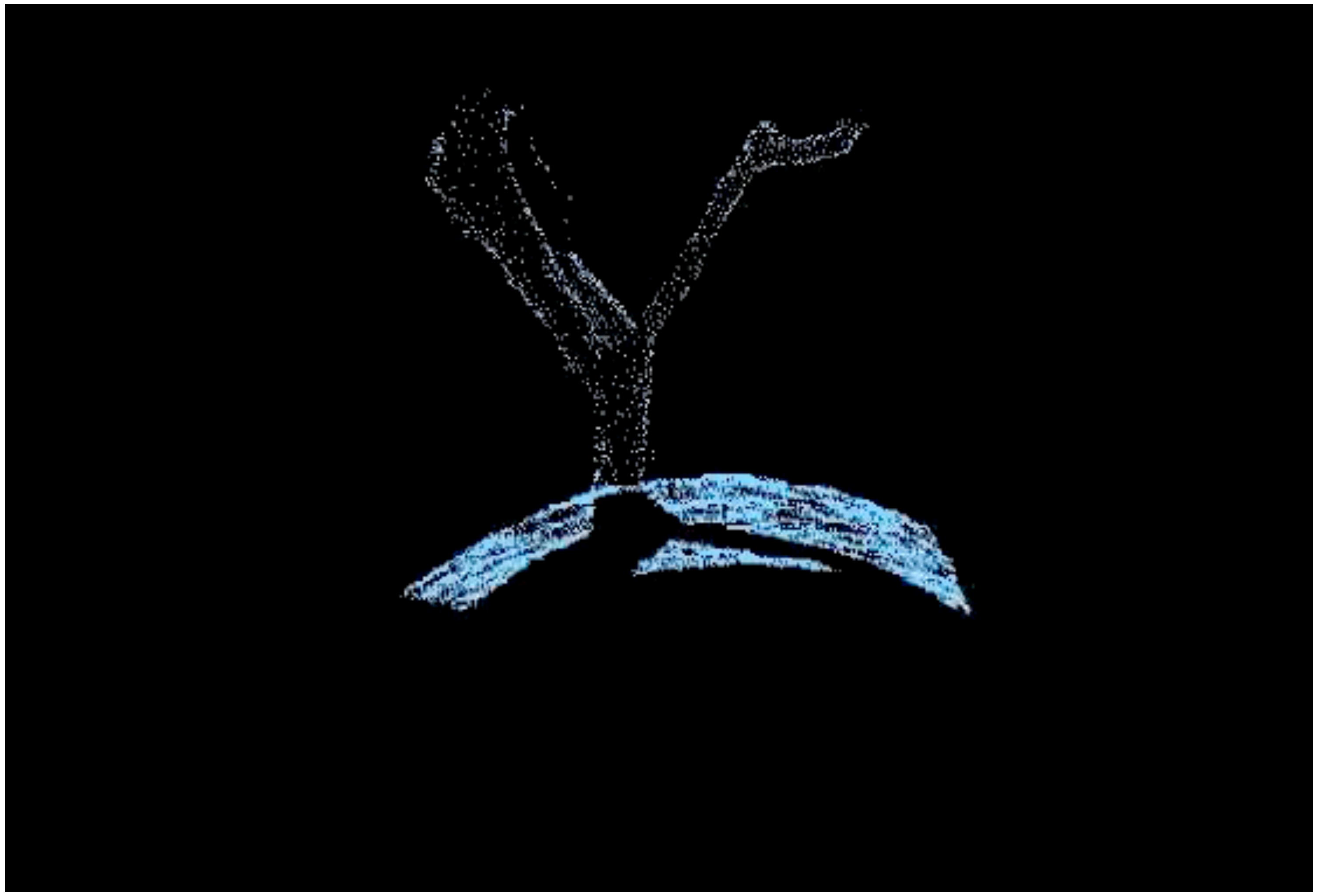
TRACKS



RESULTS



STRUCTURE FROM MOTION



Agarwal *et al.*, ICCV 2009