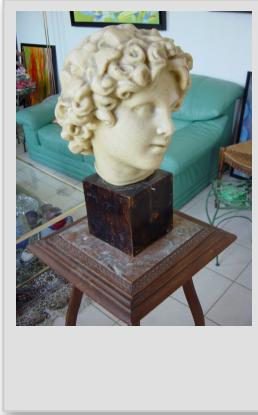
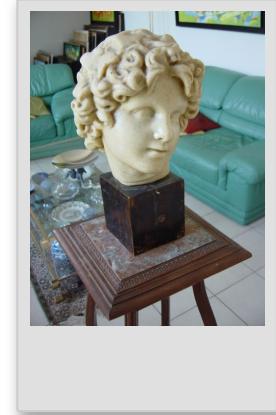




+



+



=



Multi-view 3D Reconstruction for Dummies

Jianxiong Xiao



SFMedu Program with Code

Download from:

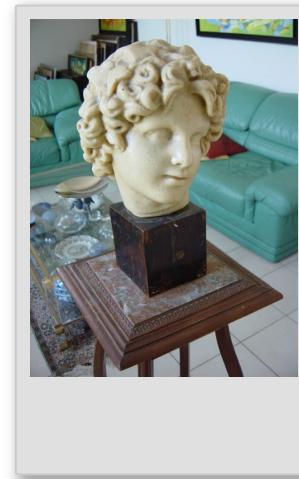
<http://mit.edu/jxiao/Public/software/SFMedu/>



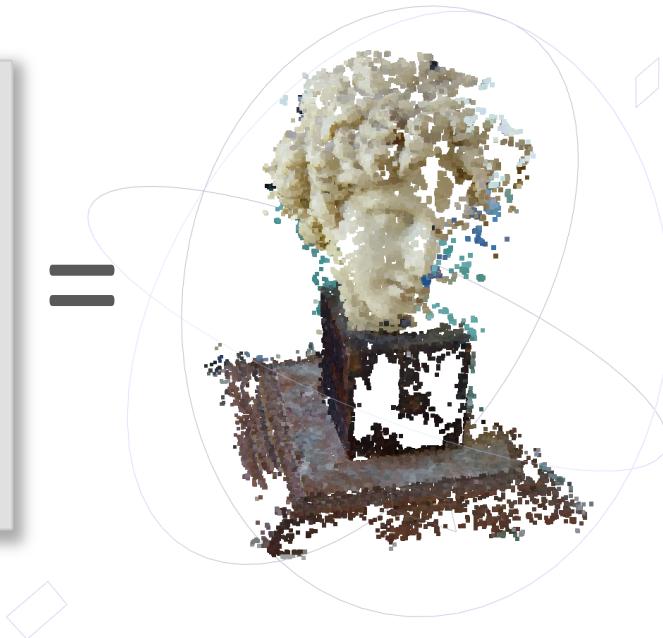
+



+



=



What

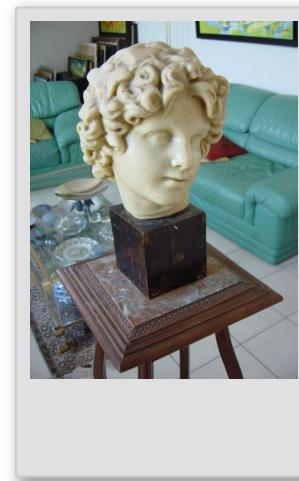
What will you learn in the following one hour?



+



+



=

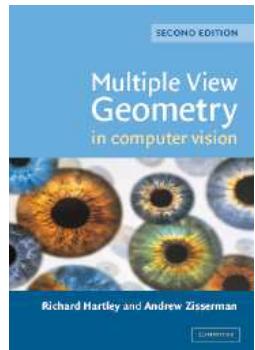


After this Lecture

You should be able to

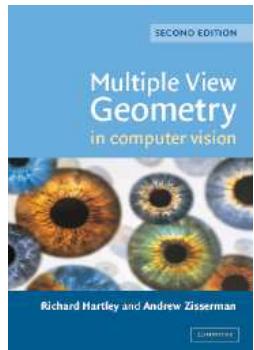
- Write your own structure from motion pipeline
- Implement a multiple view stereo system
- Know well about pinhole camera model
- Establish foundation to learn more multiple view geometry theories
- Know how to estimate the parameters for a model using linear system of equations
- Know how to solve non-linear least square problem in practice

How



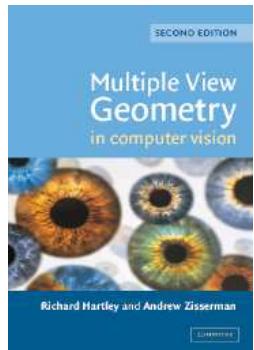
Reading: the MVG bible

How



Reading: the MVG bible
(need ~2 years)

How

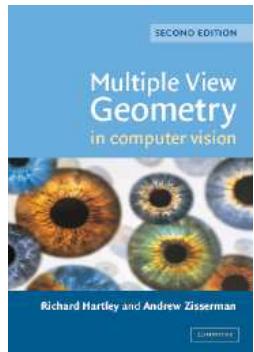


Reading: the MVG bible
(need ~2 years)



Coding

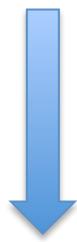
How



Reading: the MVG bible
(need ~2 years)



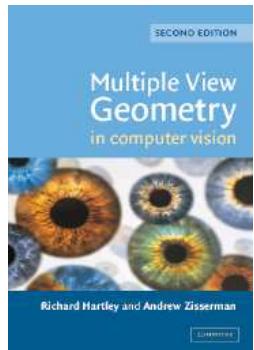
Coding



Crying: Not Working At All



How



Reading: the MVG bible
(need ~2 years)



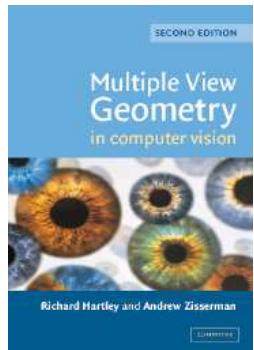
Coding



Crying: Not Working At All



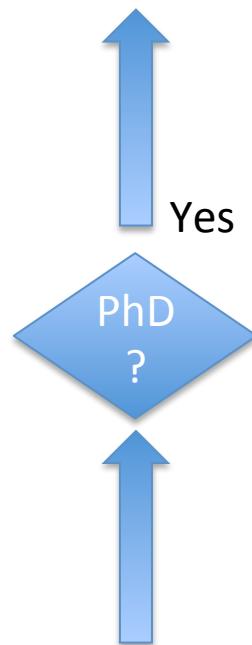
How



Reading: the MVG bible
(need ~2 years)



Coding

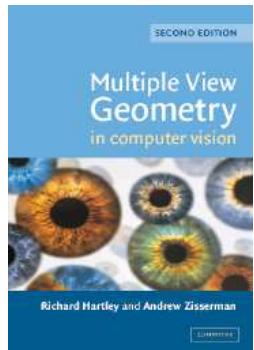


Yes



Crying: Not Working At All

How



Reading: the MVG bible
(need ~2 years)



5-6
years



Coding

PhD
?



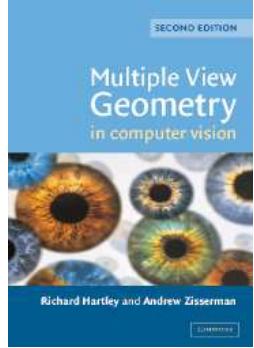
Yes



Crying: Not Working At All

How: a complete new way of learning

Standard Way



Reading: the MVG bible
(need ~2 years)



Coding



Crying: Not Working At All

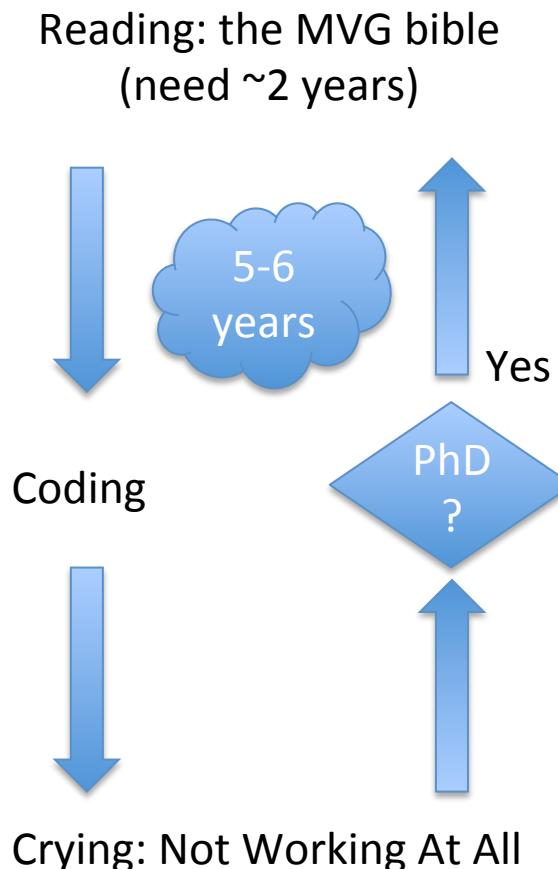
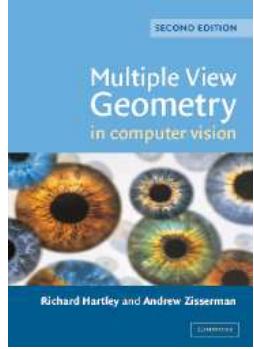
Our Way



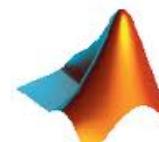
Our Way

How: a complete new way of learning

Standard Way



Our Way



Run a program



See pictures



Listen stories

$$\mathbf{X} = \mathbf{P}\mathbf{X}$$

Play with math



Dance or sing

The Basics: My Life Story

My Life Story



My Life Story



Mr. Xiao

My Life Story



Xiao

My Life Story



X

I am a 3D Point



X

I am a 3D Point



X

I am a 3D Point



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

I am small



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

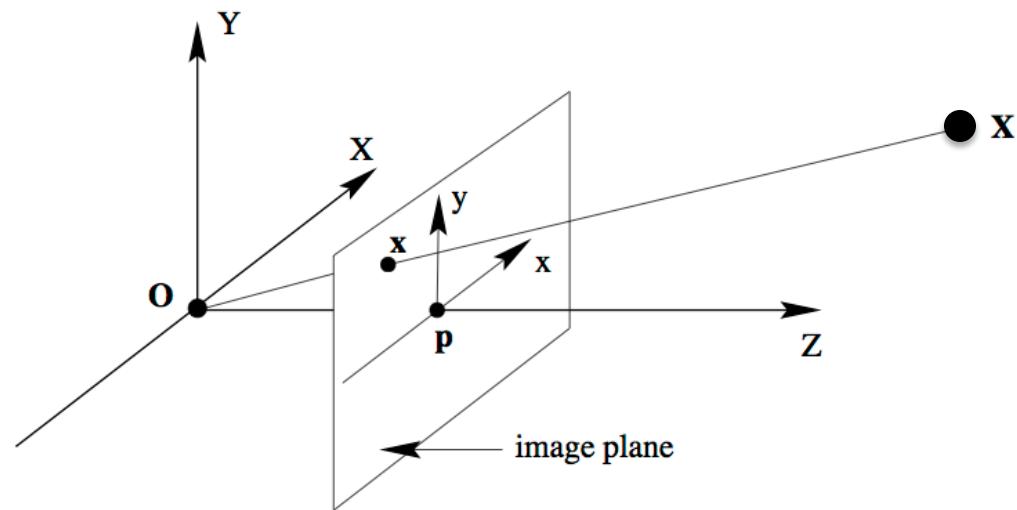
I am sexy

People loves to take picture of me.



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

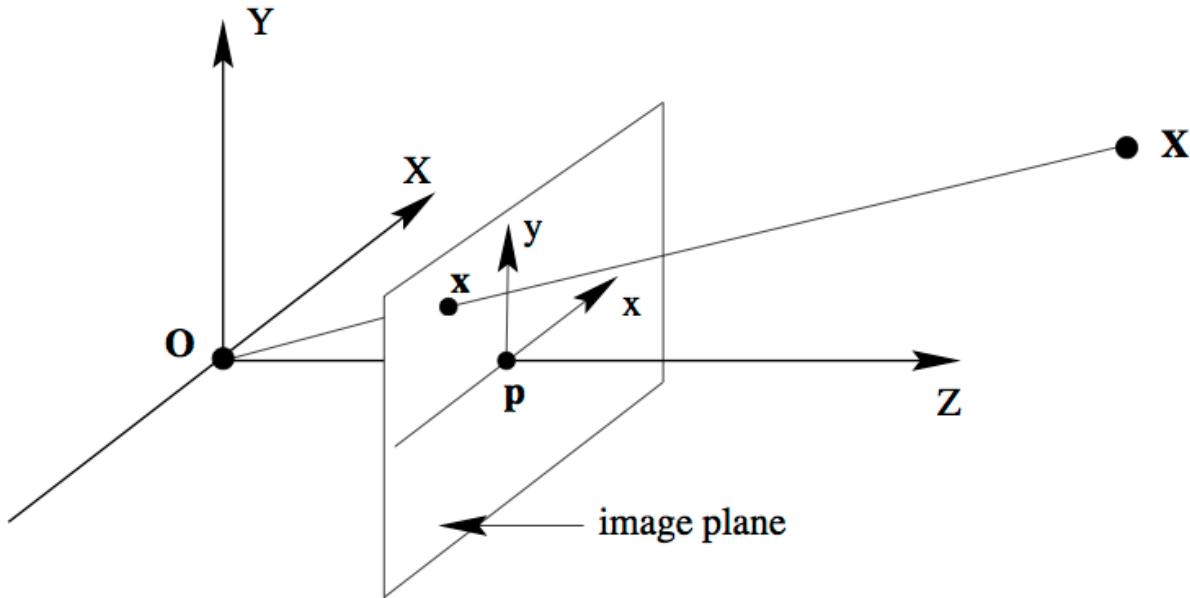
When they take a picture:



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

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

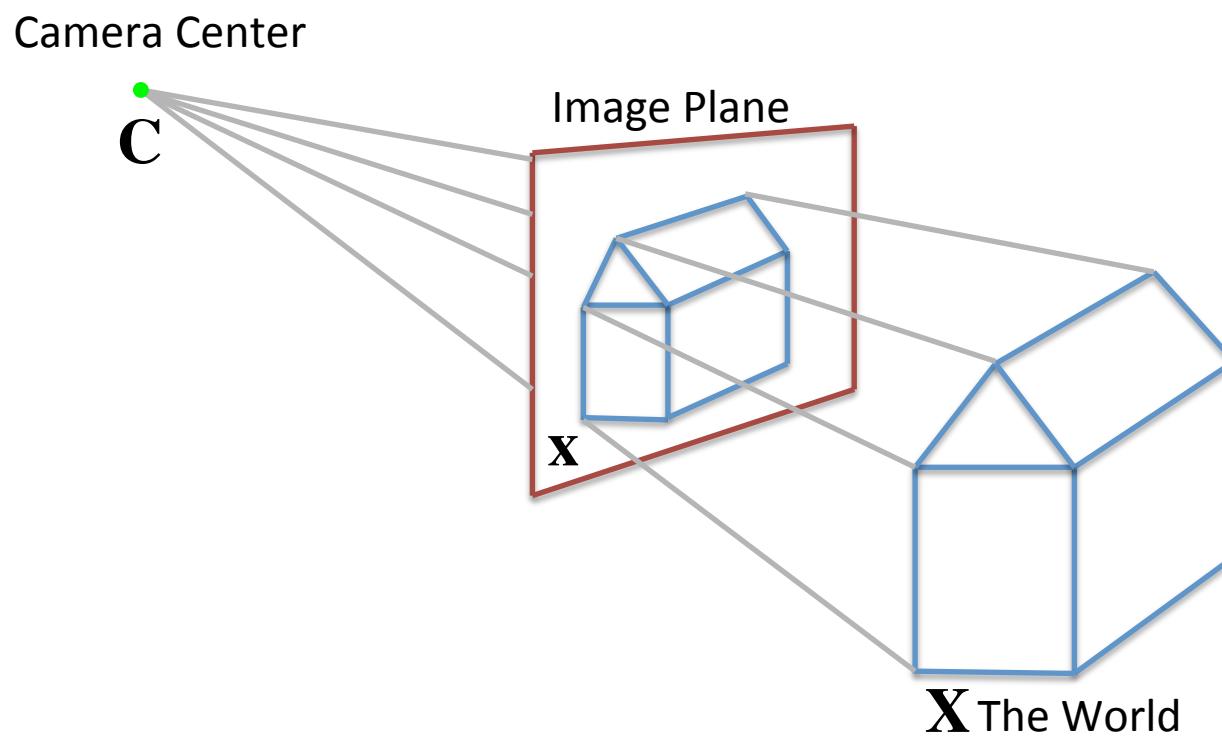
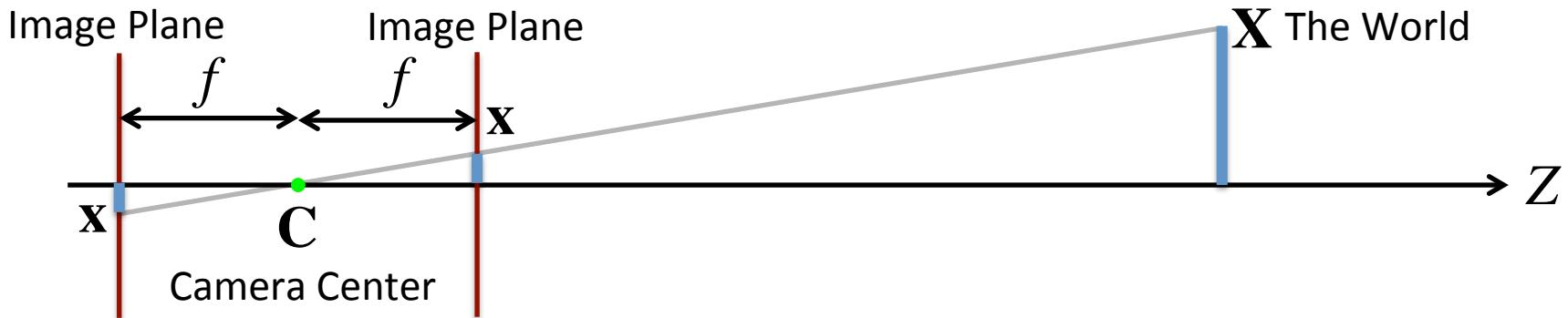
When they take a picture:



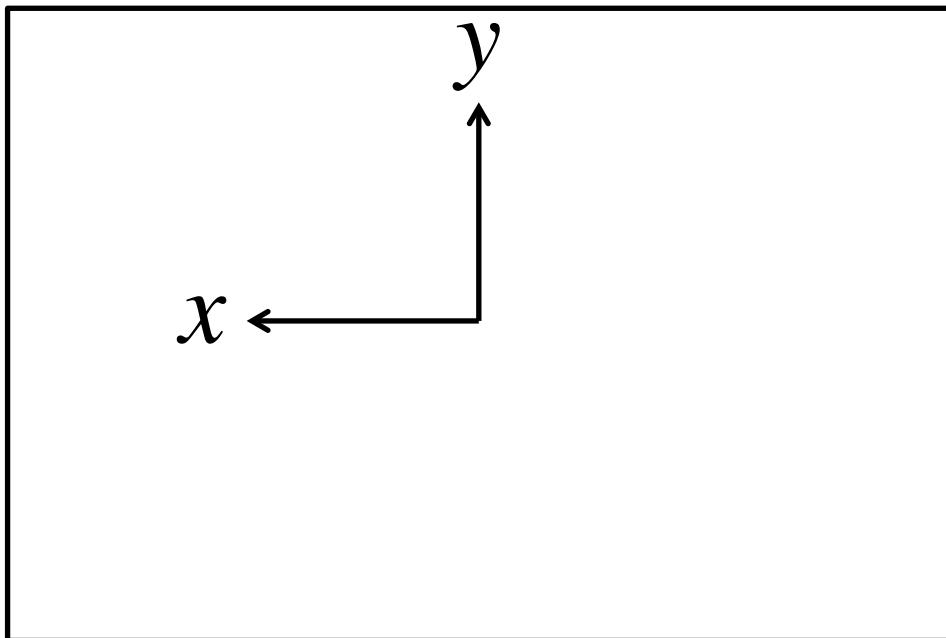
$$\lambda \begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \leftrightarrow \quad \begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Up to a scale

Why the image plane is in front?

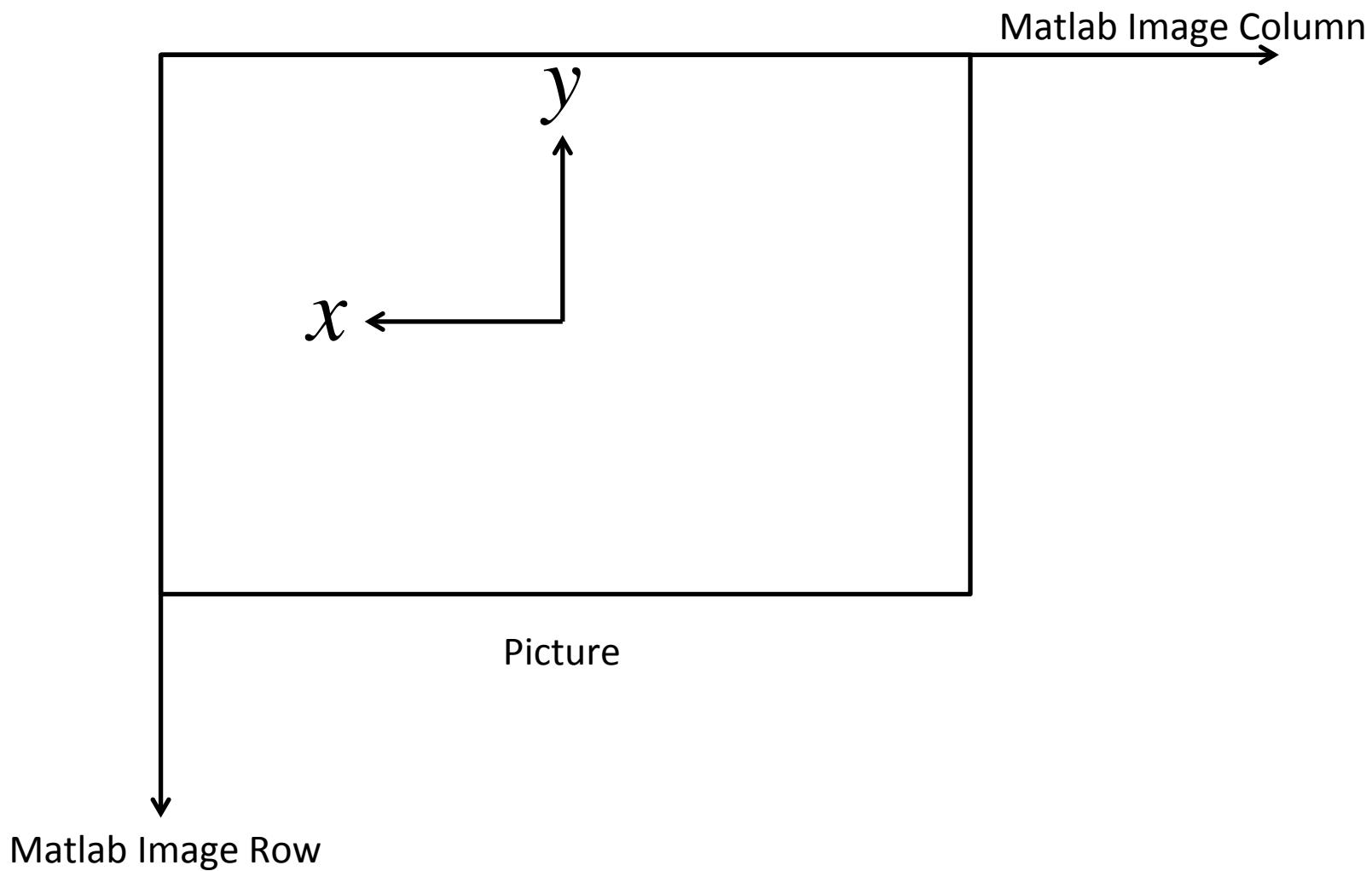


When they take a picture:

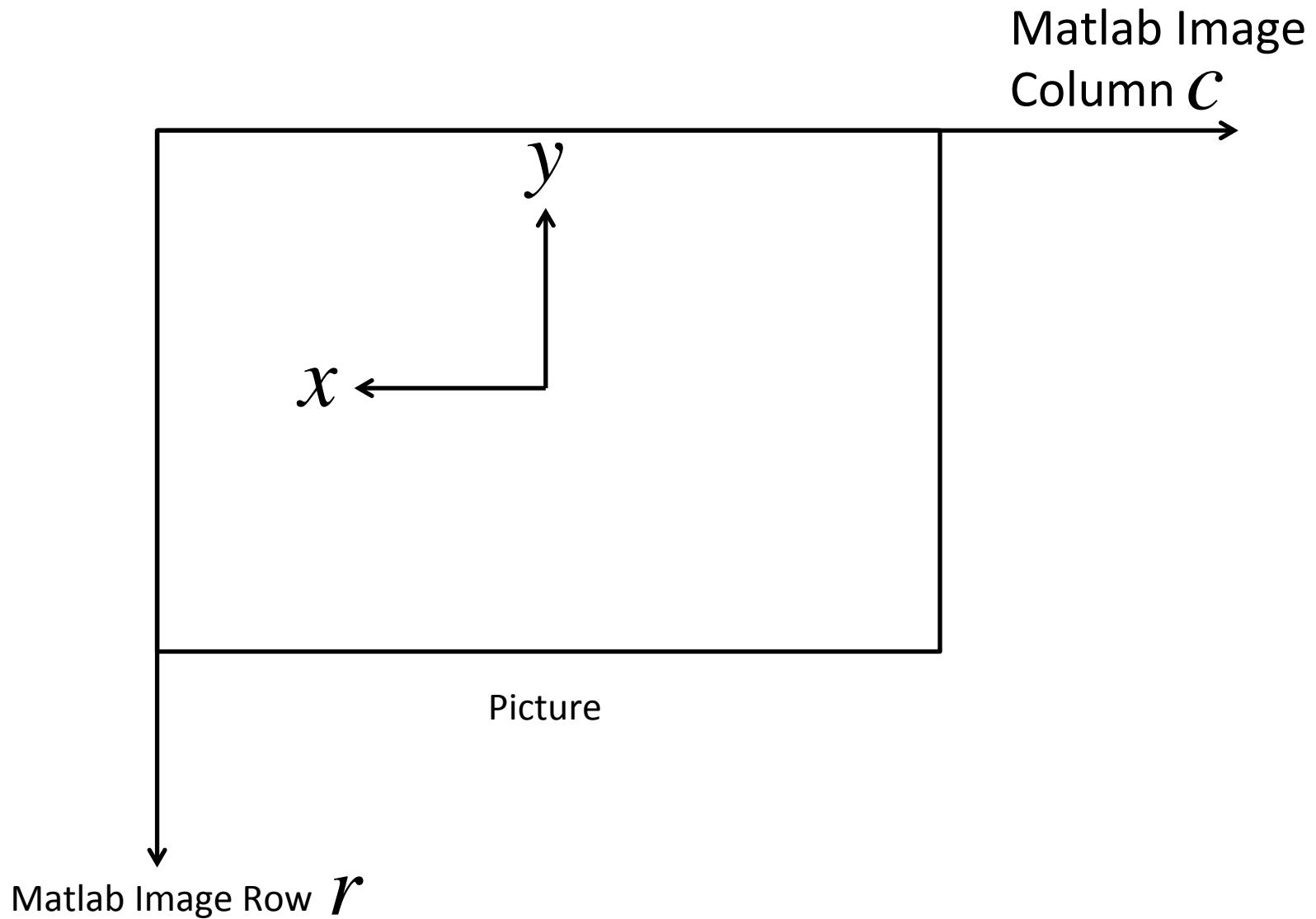


Picture

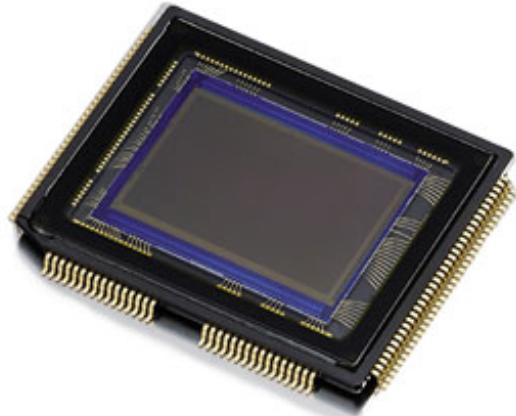
When they take a picture:



When they take a picture:



When they take a picture:



We let f to take care of this as well.
Unit of f is pixel/world unit.



World Unit: e.g. Meters

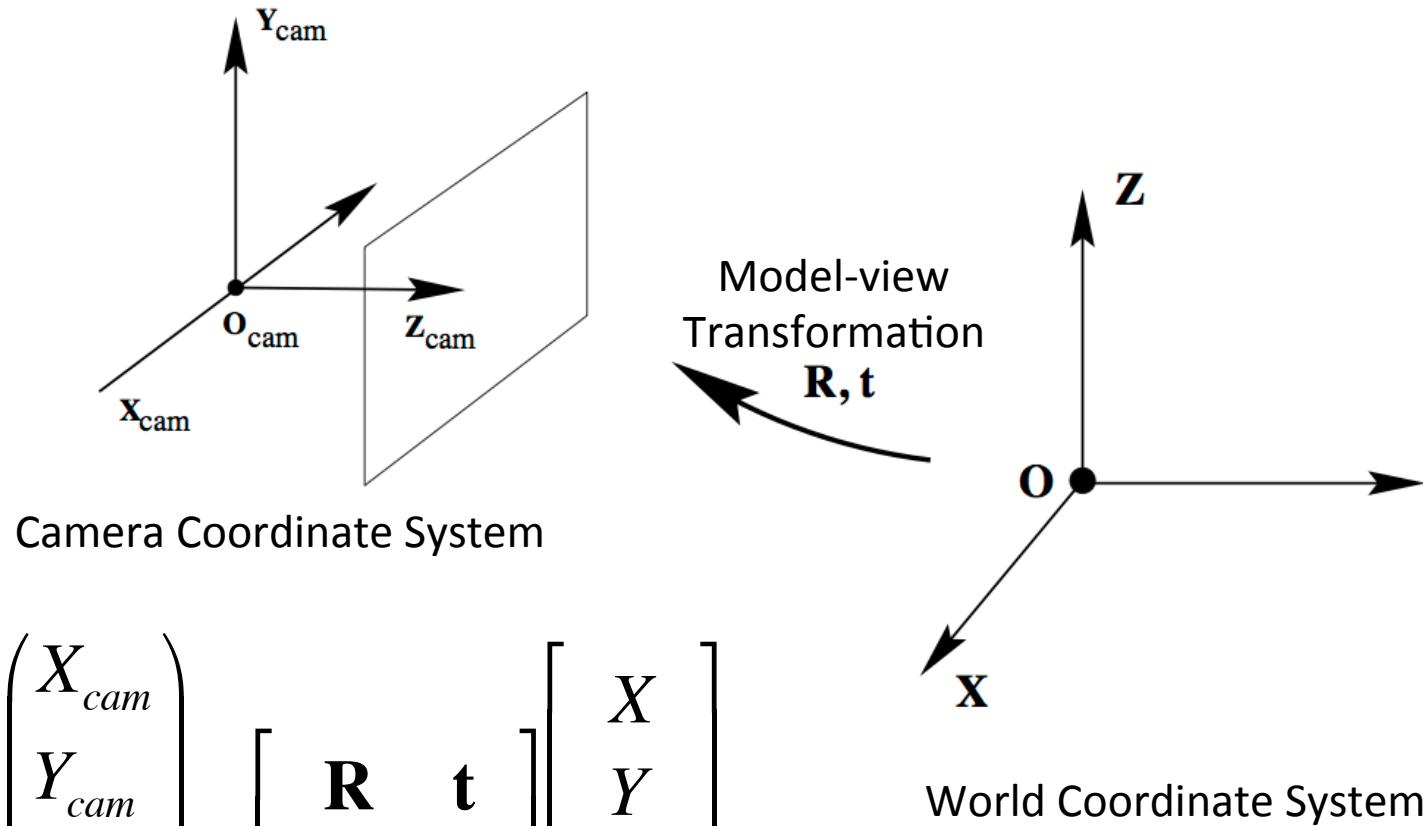
Image Unit: Pixels

$$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

I live in a real world



$$\begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

World Coordinate System

World Coor. → Camera Coor.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



$$\mathbf{x} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \mathbf{X}$$

Camera Parameter
Camera Projection Matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

Intrinsic

Extrinsic



$$\mathbf{x} = \mathbf{P} \mathbf{X}$$

I am sexy

- When people take a picture of me:

$$\mathbf{x} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \mathbf{X}$$

I am very sexy

- When people take two pictures with same camera setting:

$$\mathbf{x}_1 = \mathbf{K}[\mathbf{R}_1 | \mathbf{t}_1] \mathbf{X}$$

$$\mathbf{x}_2 = \mathbf{K}[\mathbf{R}_2 | \mathbf{t}_2] \mathbf{X}$$

I am very very sexy

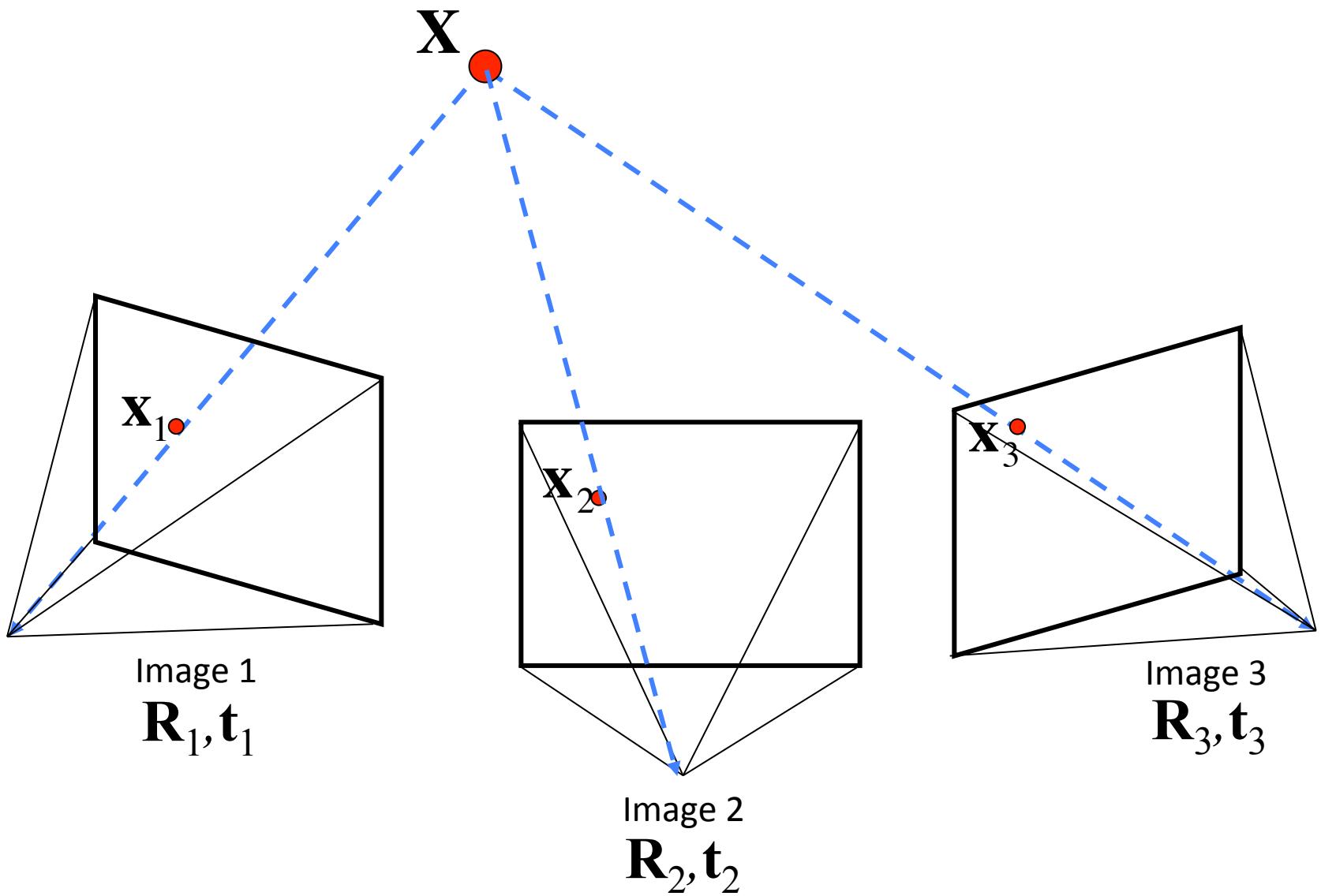
- When people take three pictures with same camera setting:

$$\mathbf{x}_1 = \mathbf{K}[\mathbf{R}_1 | \mathbf{t}_1] \mathbf{X}$$

$$\mathbf{x}_2 = \mathbf{K}[\mathbf{R}_2 | \mathbf{t}_2] \mathbf{X}$$

$$\mathbf{x}_3 = \mathbf{K}[\mathbf{R}_3 | \mathbf{t}_3] \mathbf{X}$$

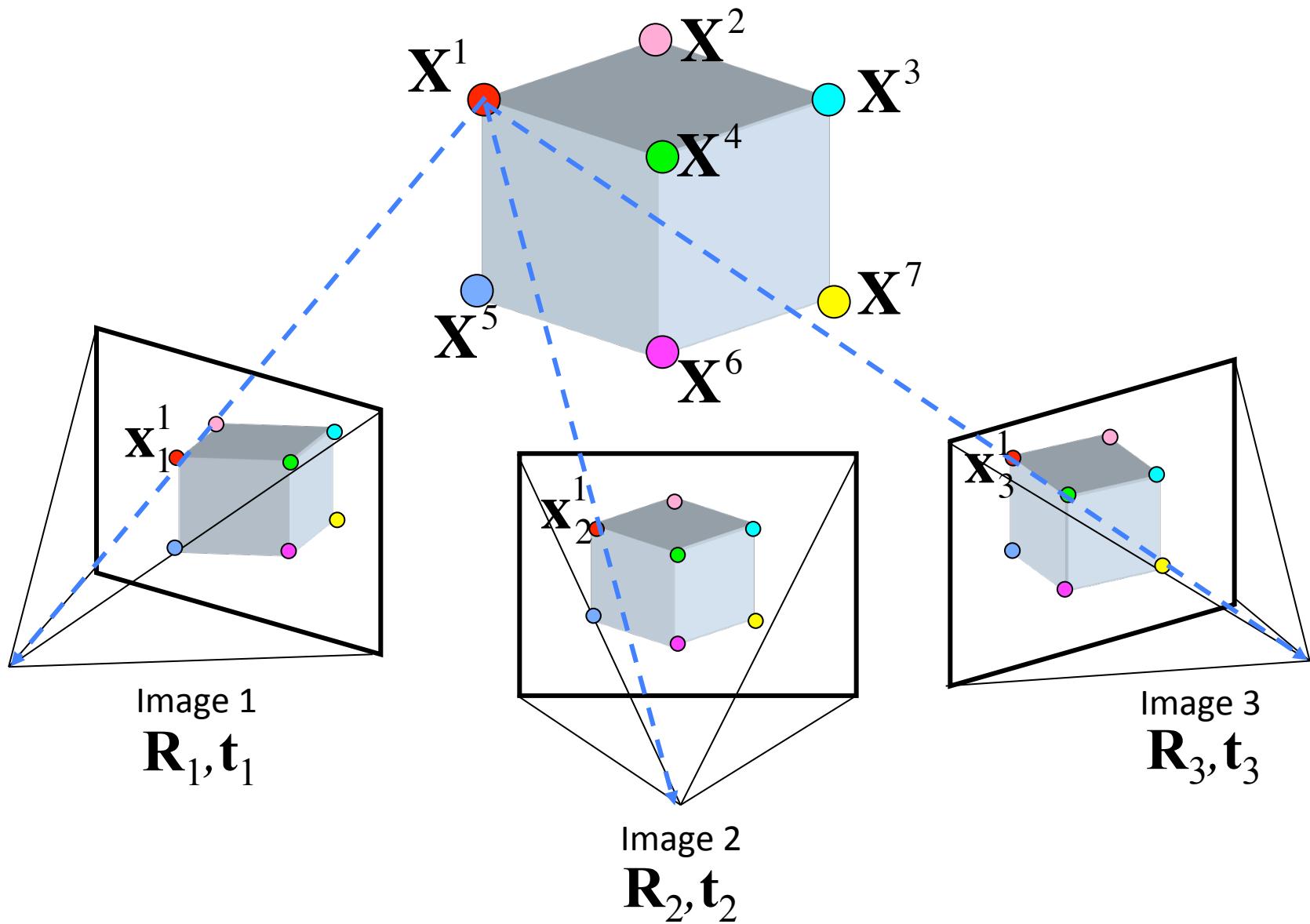
I am very very sexy



I join “America's Next Top Model”



I join “America's Next Top Model”

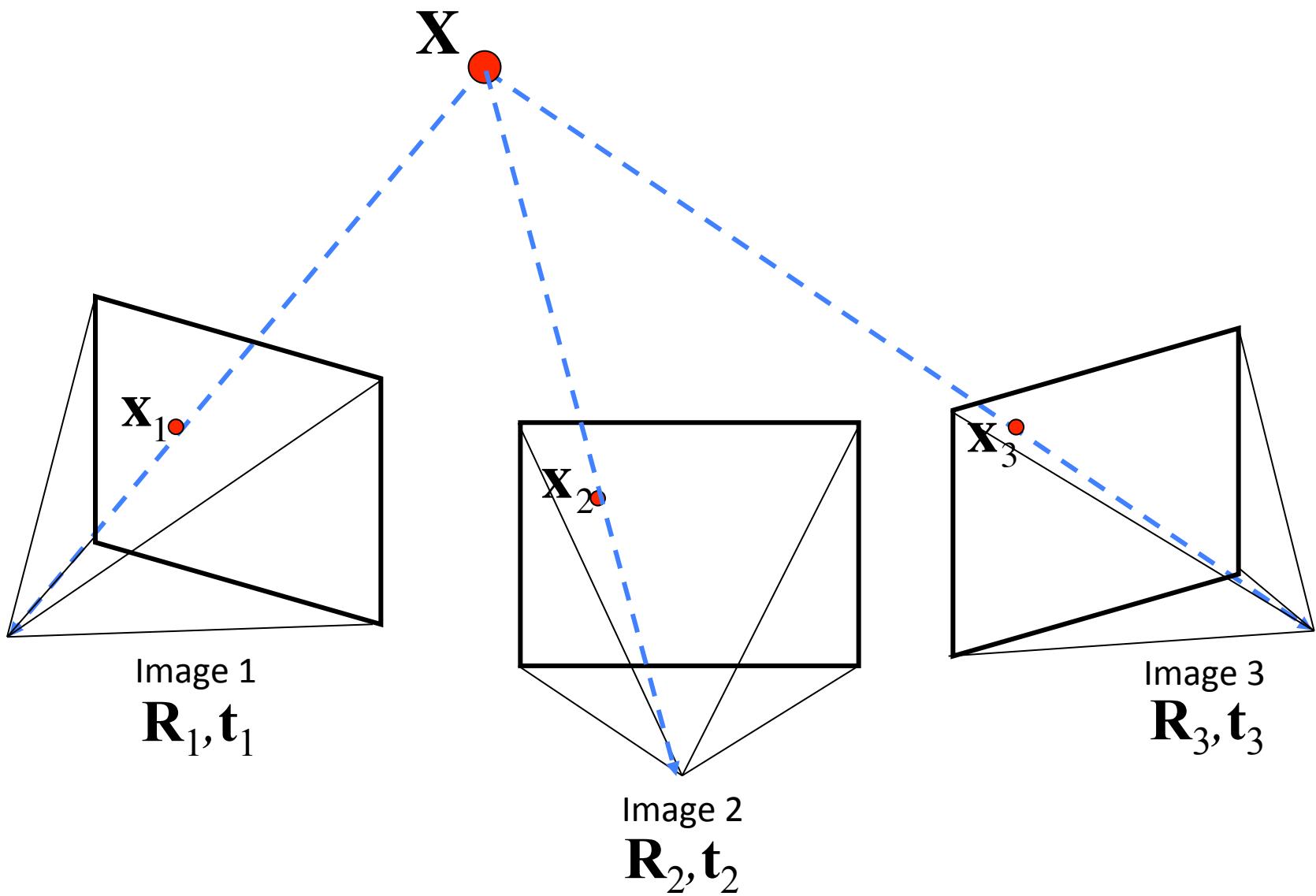


I join “America's Next Top Model”

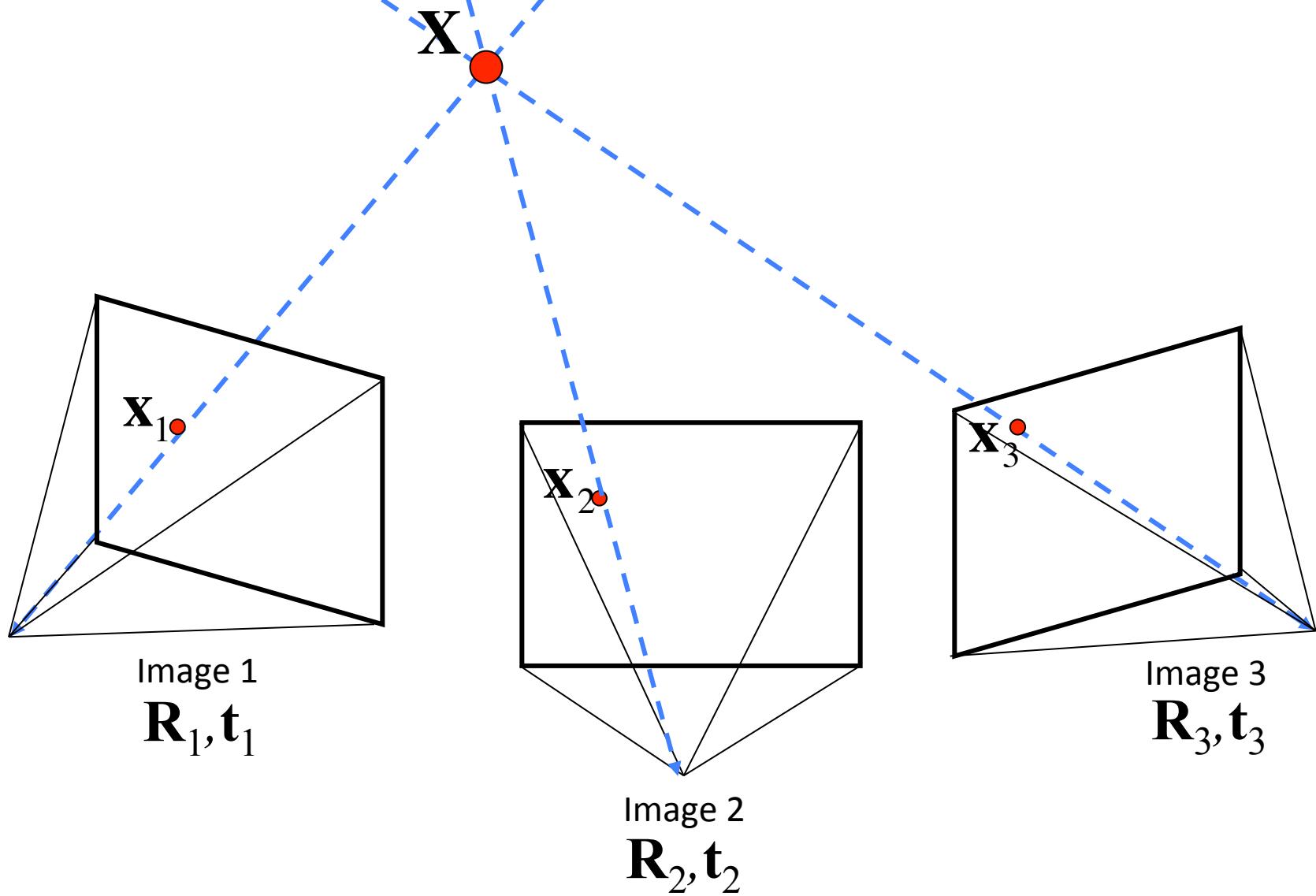
	Point 1	Point 2	Point 3
Image 1	$\mathbf{x}_1^1 = \mathbf{K}[\mathbf{R}_1 \mathbf{t}_1] \mathbf{X}^1$	$\mathbf{x}_1^2 = \mathbf{K}[\mathbf{R}_1 \mathbf{t}_1] \mathbf{X}^2$	
Image 2	$\mathbf{x}_2^1 = \mathbf{K}[\mathbf{R}_2 \mathbf{t}_2] \mathbf{X}^1$	$\mathbf{x}_2^2 = \mathbf{K}[\mathbf{R}_2 \mathbf{t}_2] \mathbf{X}^2$	$\mathbf{x}_2^3 = \mathbf{K}[\mathbf{R}_2 \mathbf{t}_2] \mathbf{X}^3$
Image 3	$\mathbf{x}_3^1 = \mathbf{K}[\mathbf{R}_3 \mathbf{t}_3] \mathbf{X}^1$		$\mathbf{x}_3^3 = \mathbf{K}[\mathbf{R}_3 \mathbf{t}_3] \mathbf{X}^3$

Same Camera Same Setting = Same \mathbf{K}

Triangulation



Triangulation



Other Intrinsic Camera Parameters

- Principle point offset
 - especially when images are cropped (Internet)
- Skew

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \xrightarrow{\text{blue arrow}} \quad \mathbf{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Radial distortion (due to optics of the lens)

$$r^2 = \|\mathbf{x}\|^2 = x^2 + y^2$$

$$\mathbf{x}' = (1 + k_1 r^2 + k_2 r^4) \mathbf{x}$$



Other Camera Models

- Fisheye
- Mirror
- Panorama
- Tilt-Shift Lens
- Biological Eyes



http://www.popgadget.net/2006/07/fisheye_camera.php



<http://www.0-360.com/>



<http://sun360.csail.mit.edu>

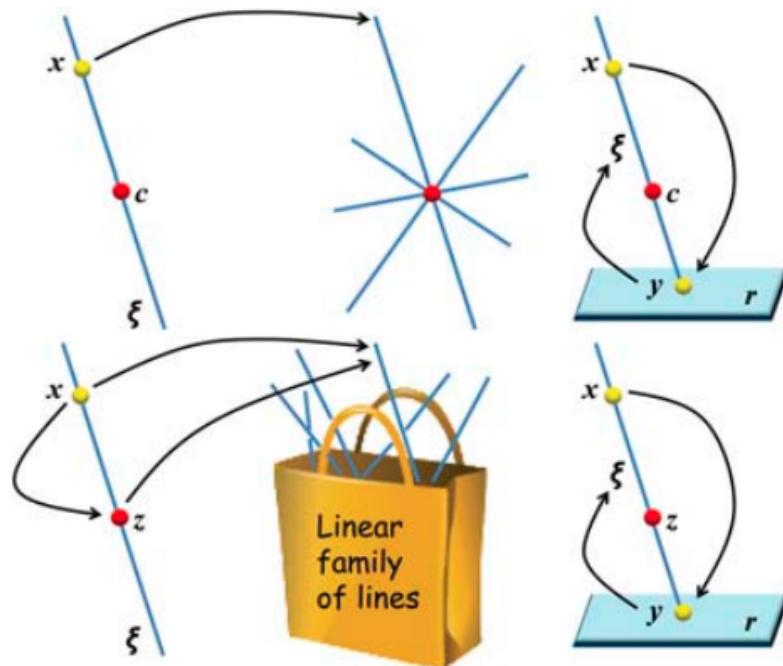


Canon TS-E 24mm f/3.5L II



What is a camera?

Definition: A camera is a two-parameter linear family of lines—that is, a degenerate regulus (rank-3 family), or a non-degenerate linear congruence (rank-4 family).



White-board Exercise

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

I brought a normal compact digital camera.
I took a picture at resolution 640x480 pixels.
What is the possible value of the focal length f ?

Focal Length
5.0 (W) - 40.0 (T) mm (35mm film equivalent: 28 (W) - 224 (T) mm)

Digital Zoom
4x

Focusing Range
Normal: 2.0 in. (5cm) - infinity (W), 3.3 ft. (1m) - infinity (T)
Auto: 0.4 in. (1cm) - infinity (W), 3.3 ft. (1m) - infinity (T)
Macro: 0.4 in. - 1.6 ft. (1-50cm) (W)

Autofocus System
TTL Autofocus

full-frame 35mm = width 36 mm height 24 mm
http://en.wikipedia.org/wiki/Angle_of_view

Canon

Guest | Sign in
0 items in shopping cart
Enter product name or item # Search

FREE GROUND SHIPPING PROMOTION. CLICK FOR DETAILS

SHOP ACCESSORIES

SIGN UP FOR THE CANON HOLIDAY HUSH

PowerShot A4000 IS Blue Digital Camera

Average Rating: ★★★★☆ 4.5 Read all Reviews (1) Write a Review

You Pay: \$179.99 Quantity: 1 In Stock
Price Drop: \$20.00

ADD TO CART

Click to Enlarge

View other colors

OVERVIEW SPECIFICATIONS CUSTOMER REVIEWS WHAT'S IN THE BOX

Type: Compact digital camera with flash, 8x Optical, 4x Digital, and 32x Combined Zoom with Optical Image Stabilizer

Type: 16.0 Megapixel 1/2.3-inch CCD

Total Pixels: 16.6 Megapixels

Effective Pixels: 16.0 Megapixels

Focal Length: 5.0 (W) - 40.0 (T) mm (35mm film equivalent: 28 (W) - 224 (T) mm)

Digital Zoom: 4x

Focusing Range: Normal: 2.0 in. (5cm) - infinity (W), 3.3 ft. (1m) - infinity (T)
Auto: 0.4 in. (1cm) - infinity (W), 3.3 ft. (1m) - infinity (T)
Macro: 0.4 in. - 1.6 ft. (1-50cm) (W)

<http://goo.gl/ENaw4>

Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+

Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling

Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

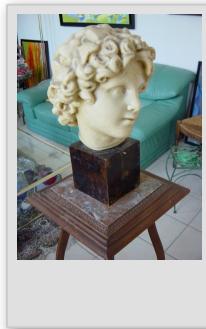
Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



+



+



=



Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



+



+



=



Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



+



+



+



Steps

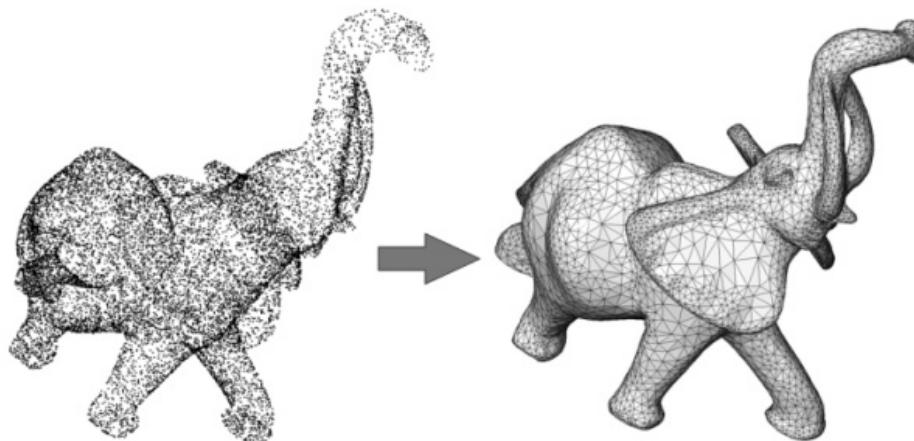
Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

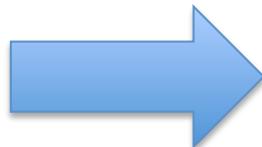
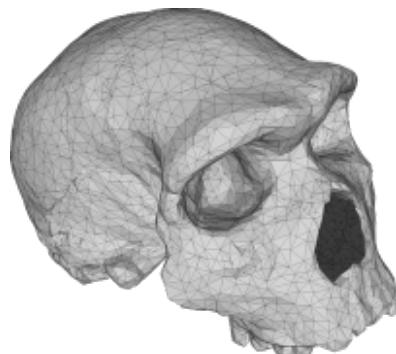
Points → Meshes: Model Fitting

Meshes → Models: Texture Mapping

+

=

Images → Models: Image-based Modeling



Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+

Meshes → Models: Texture Mapping

= **Images → Models: Image-based Modeling**

Image-based Modeling



Input Images

Image-based Modeling

Pittsburgh -- section 1



Steps

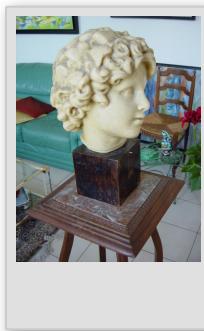
Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



+



+



=



Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

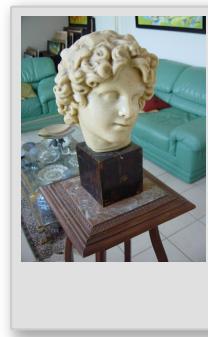
= Images → Models: Image-based Modeling



+



+



=



Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

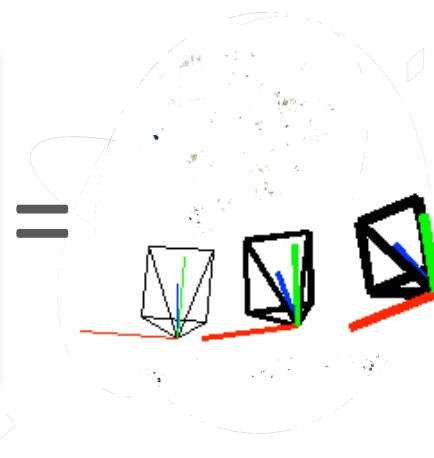
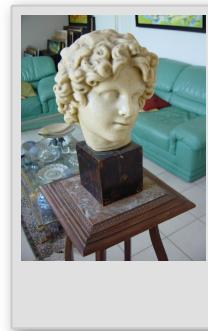
= Images → Models: Image-based Modeling



+



+

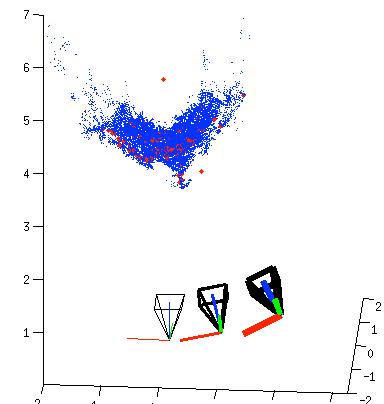
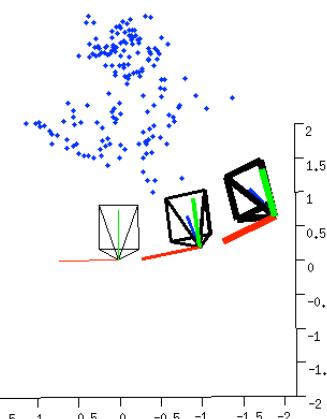
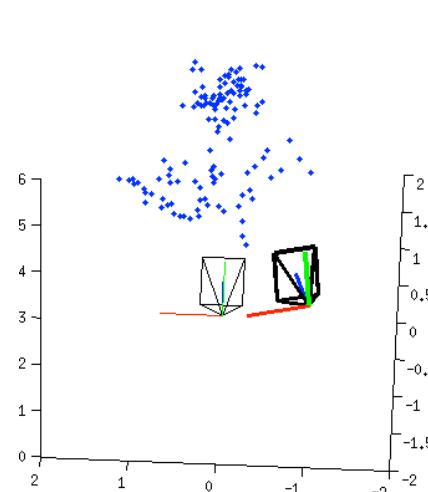
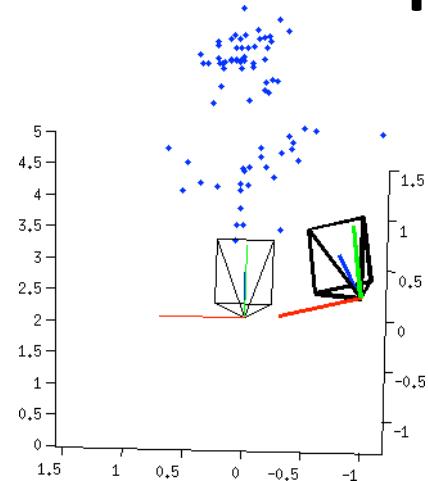


Structure From Motion

- Structure = 3D Point Cloud of the Scene
- Motion = Camera Location and Orientation
- SFM = Get the Point Cloud from Moving Cameras
- Structure and Motion: Joint Problems to Solve



Pipeline



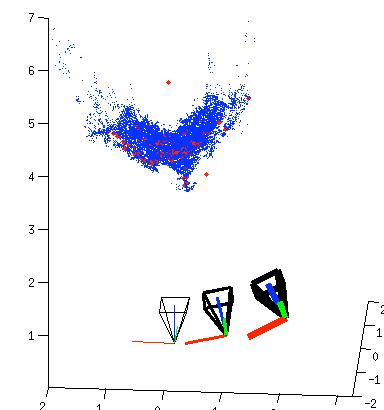
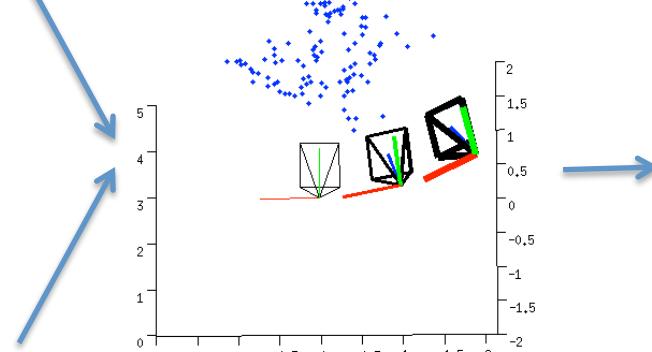
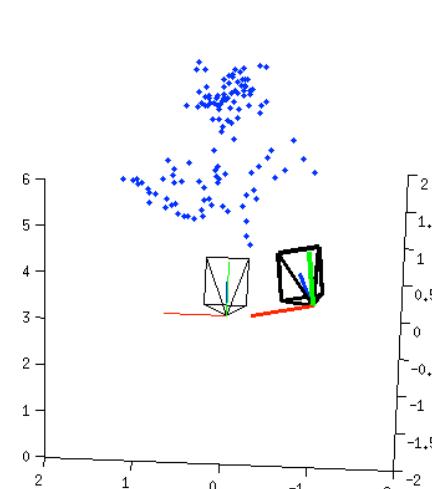
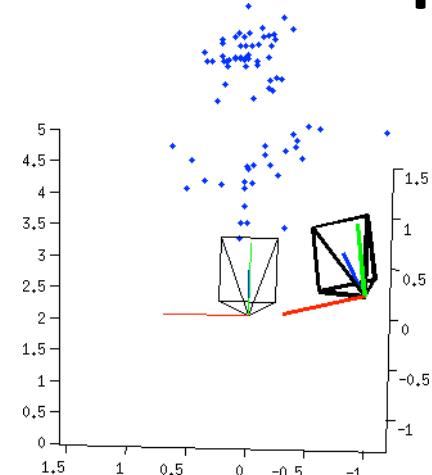
Structure from Motion (SfM)

Multi-view Stereo (MVS)

Pipeline

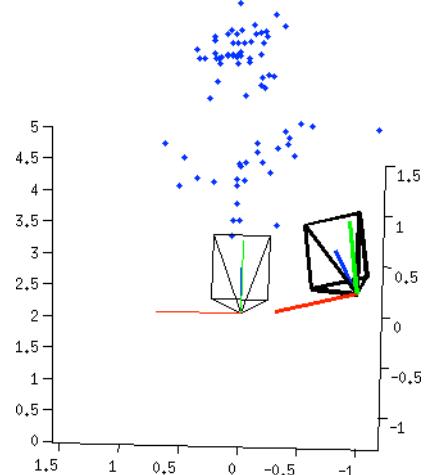


Structure from Motion (SFM)

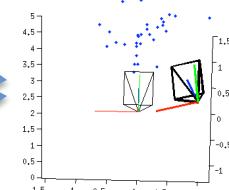
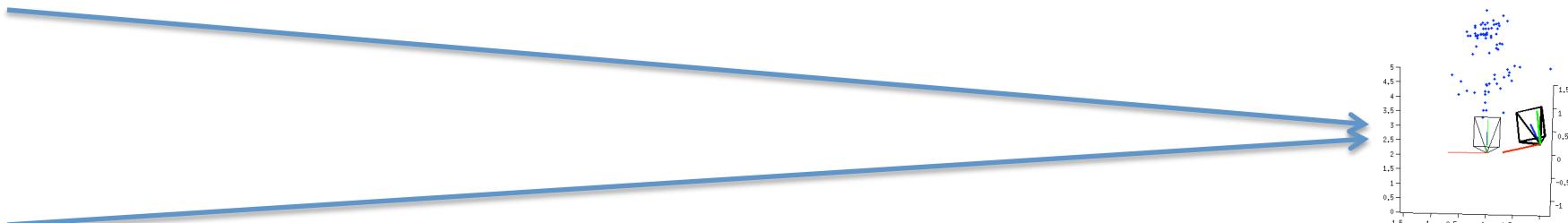


Multi-view Stereo (MVS)

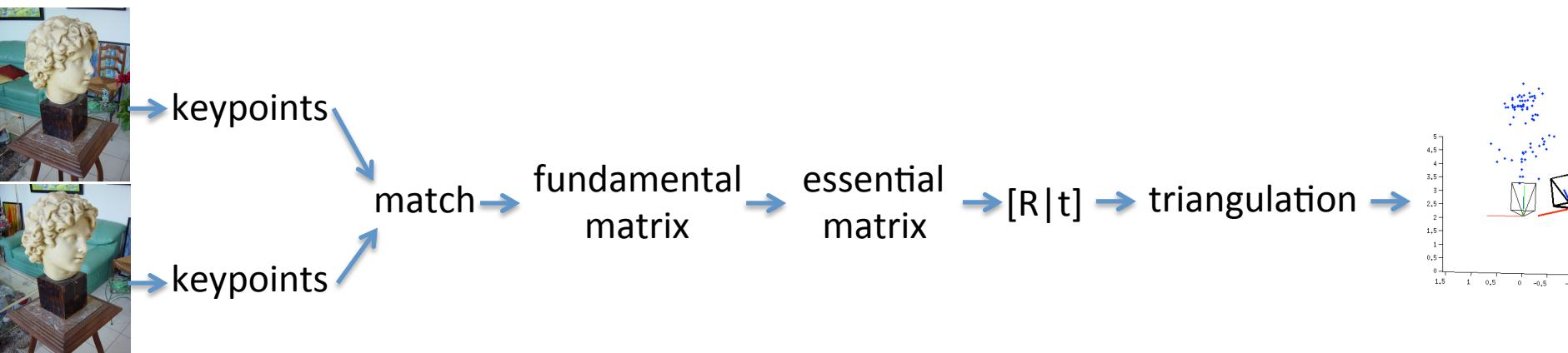
Two-view Reconstruction



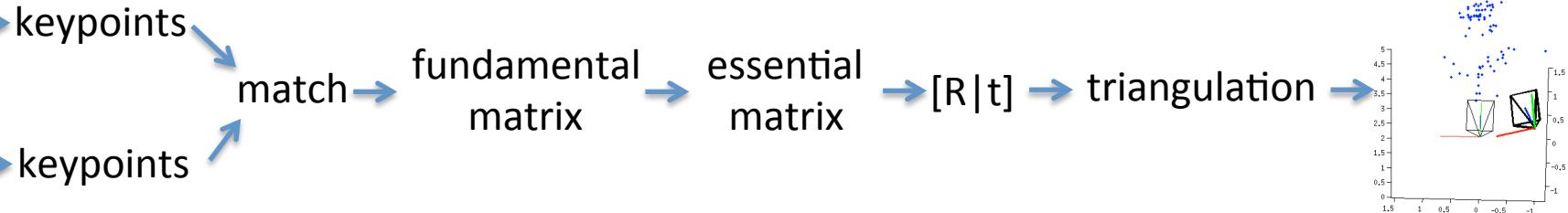
Two-view Reconstruction



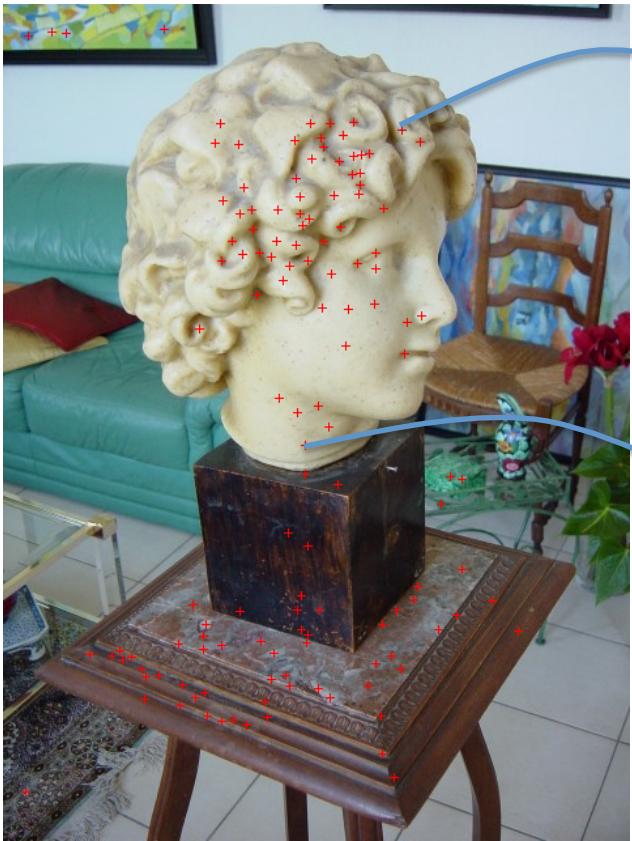
Two-view Reconstruction



Keypoints Detection



Descriptor for each point



SIFT
descriptor

SIFT
descriptor



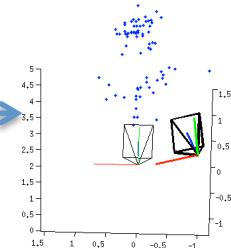
keypoints

match

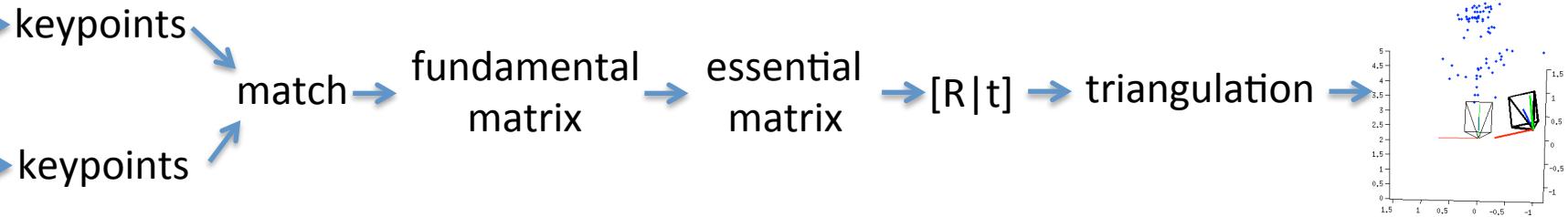
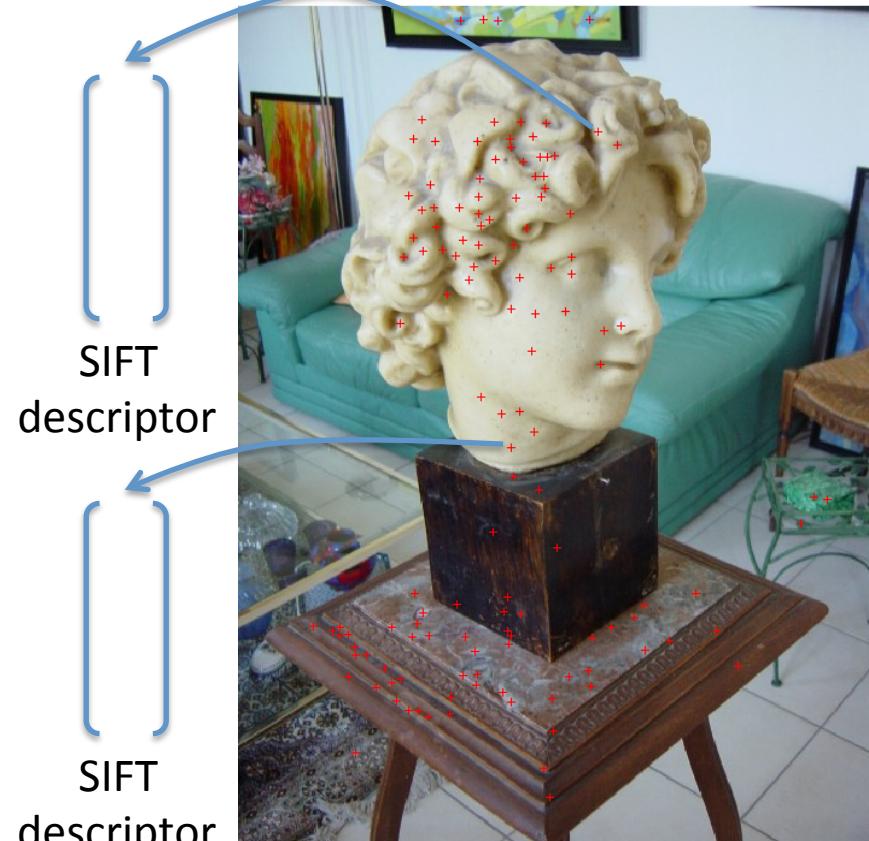
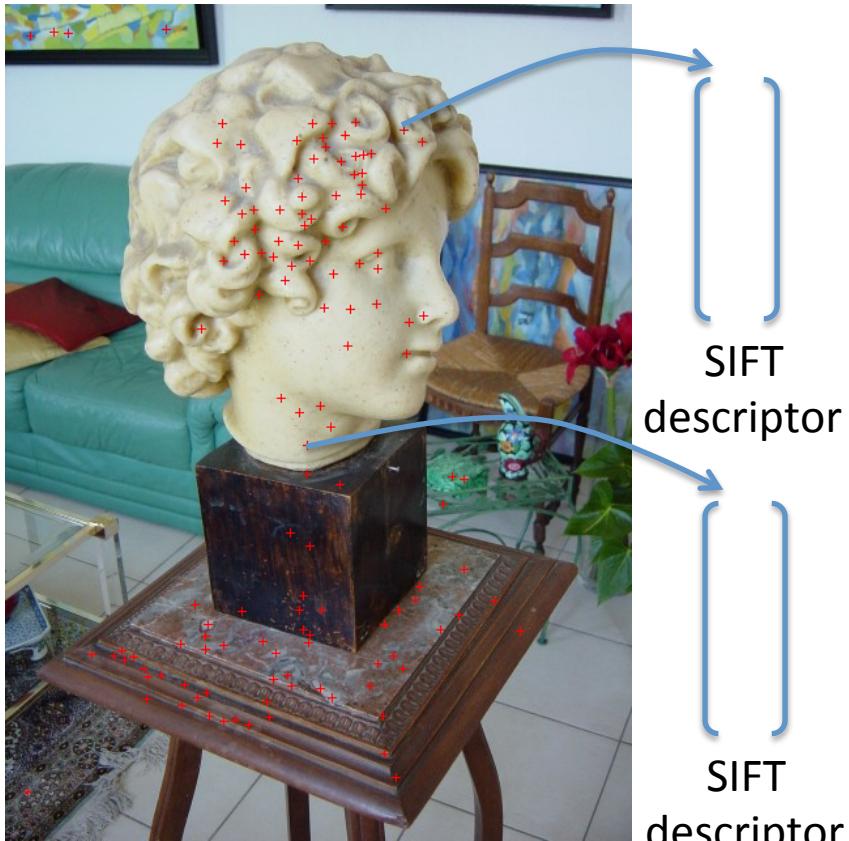
fundamental
matrix

essential
matrix

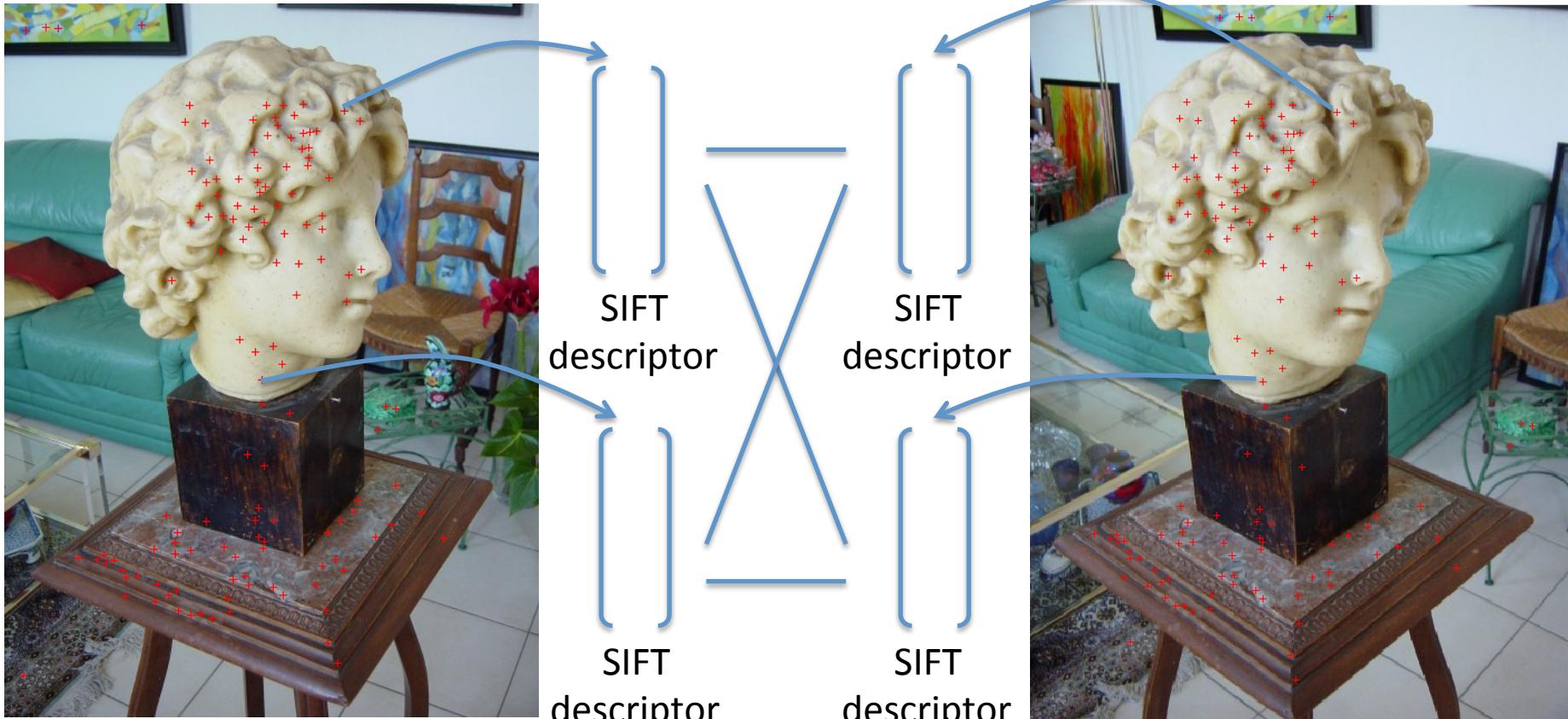
$[R|t]$ \rightarrow triangulation



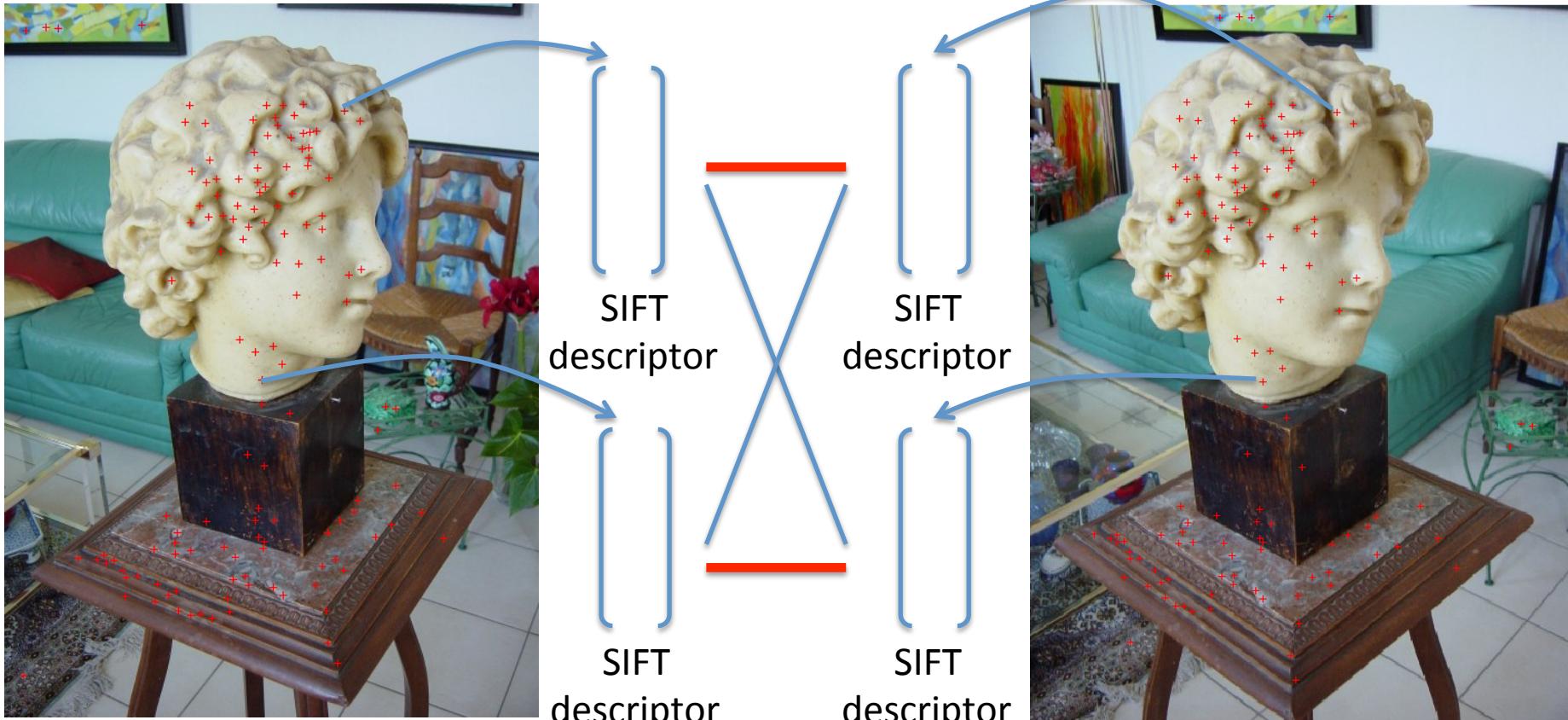
Same for the other images



Point Match for correspondences



Point Match for correspondences



keypoints

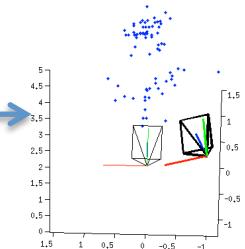
match

fundamental
matrix

essential
matrix

$[R|t]$

triangulation



Fundamental Matrix

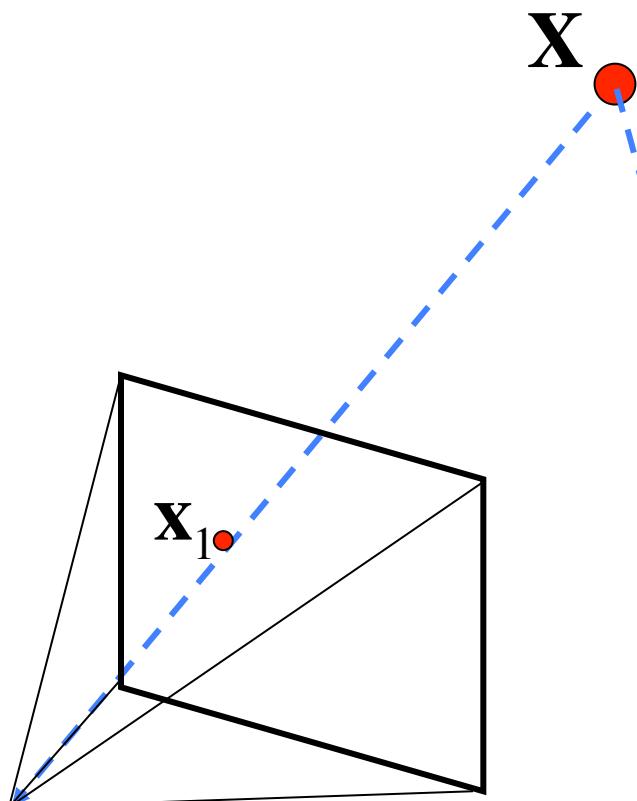


Image 1

$$\mathbf{R}_1, \mathbf{t}_1$$

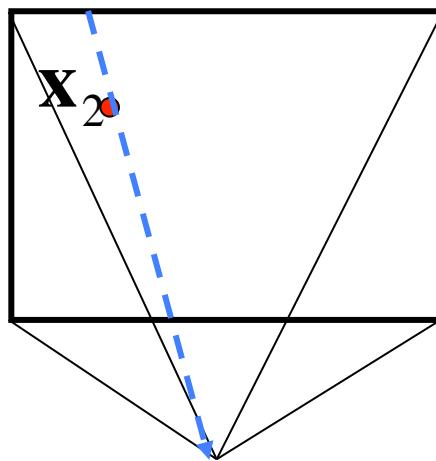


Image 2

$$\mathbf{R}_2, \mathbf{t}_2$$

$$\mathbf{x}_1 \Leftrightarrow \mathbf{x}_2$$

$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$

Gangnam Style



Gangnam Style



Fundamental Matrix Song



Fundamental Matrix Song

After this song, there will be a quiz.
Listen carefully!



Exercise

What is the difference between Fundamental Matrix and Homography?
(Both of them are to explain 2D point to 2D point correspondences.)



Estimating Fundamental Matrix

- Given a correspondence

$$\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$$

- The basic incidence relation is

$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$



$$\begin{bmatrix} x_1x_2, x_1y_2, x_1, y_1x_2, y_1y_2, y_1, x_2, y_2, 1 \end{bmatrix} = 0$$

Need 8 points

$$\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Estimating Fundamental Matrix

$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$ for 8 point correspondences:

$\mathbf{x}_1^1 \leftrightarrow \mathbf{x}_2^1, \mathbf{x}_1^2 \leftrightarrow \mathbf{x}_2^2, \mathbf{x}_1^3 \leftrightarrow \mathbf{x}_2^3, \mathbf{x}_1^4 \leftrightarrow \mathbf{x}_2^4, \mathbf{x}_1^5 \leftrightarrow \mathbf{x}_2^5, \mathbf{x}_1^6 \leftrightarrow \mathbf{x}_2^6, \mathbf{x}_1^7 \leftrightarrow \mathbf{x}_2^7, \mathbf{x}_1^8 \leftrightarrow \mathbf{x}_2^8$

$$\left[\begin{array}{cccccccc|c} x_1^1 x_2^1 & x_1^1 y_2^1 & x_1^1 & y_1^1 x_2^1 & y_1^1 y_2^1 & y_1^1 & x_2^1 & y_2^1 & 1 & f_{11} \\ x_1^2 x_2^2 & x_1^2 y_2^2 & x_1^2 & y_1^2 x_2^2 & y_1^2 y_2^2 & y_1^2 & x_2^2 & y_2^2 & 1 & f_{12} \\ x_1^3 x_2^3 & x_1^3 y_2^3 & x_1^3 & y_1^3 x_2^3 & y_1^3 y_2^3 & y_1^3 & x_2^3 & y_2^3 & 1 & f_{13} \\ x_1^4 x_2^4 & x_1^4 y_2^4 & x_1^4 & y_1^4 x_2^4 & y_1^4 y_2^4 & y_1^4 & x_2^4 & y_2^4 & 1 & f_{21} \\ x_1^5 x_2^5 & x_1^5 y_2^5 & x_1^5 & y_1^5 x_2^5 & y_1^5 y_2^5 & y_1^5 & x_2^5 & y_2^5 & 1 & f_{22} \\ x_1^6 x_2^6 & x_1^6 y_2^6 & x_1^6 & y_1^6 x_2^6 & y_1^6 y_2^6 & y_1^6 & x_2^6 & y_2^6 & 1 & f_{23} \\ x_1^7 x_2^7 & x_1^7 y_2^7 & x_1^7 & y_1^7 x_2^7 & y_1^7 y_2^7 & y_1^7 & x_2^7 & y_2^7 & 1 & f_{31} \\ x_1^8 x_2^8 & x_1^8 y_2^8 & x_1^8 & y_1^8 x_2^8 & y_1^8 y_2^8 & y_1^8 & x_2^8 & y_2^8 & 1 & f_{32} \\ \end{array} \right] = 0 \quad \text{A} \mathbf{f} = \mathbf{0}$$

Direct Linear Transformation (DLT)

Algebraic Error vs. Geometric Error

- Algebraic Error

$$\min \|\mathbf{A}\mathbf{f}\|$$

- Geometric Error (better) Unit: pixel

$$\min \sum_j \left\| \mathbf{x}_1^j - \mathbf{F} \mathbf{x}_2^j \right\|^2$$

Solved by (non-linear) least square solver (e.g. Ceres)

RANSAC to Estimate Fundamental Matrix

- For many times
 - Pick 8 points
 - Compute a solution for \mathbf{F} using these 8 points
 - Count number of inliers that with $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2$ close to 0
- Pick the one with the largest number of inliers

Minimal problems in Computer Vision

Minimal problems in Computer Vision

cmp.felk.cvut.cz/minimal/

- » Home
- » P3P problem
- » P4P + unknown focal length
- » P4P + focal + radial distortion
- » uP2P + known vertical direction
- » uP3P + focal + radial distortion
- » P2P1L problem
- » P1P2L problem
- » 5-pt relative pose problem
- » 6-pt focal length problem
- » 6-pt calibrated-uncalibrated
- » 4-pt 3-views relative pose problem
- » 2-pt panorama stitching (1 focal)
- » 3-pt panorama stitching (2 focals)
- » 3-pt panorama stitching (1 focal+radial distortion)
- » 6-pt generalized camera problem
- » 6-pt calibrated radial distortion
- » 8-pt uncalibrated radial distortion
- » 9-pt different distortion problem
- » P2Pi registration problem
- » 3-view triangulation
- » 9-pt catadioptric
- » Hand-eye
- » Automatic generator
- » Grobener basis solver

Overview

Minimal problems in computer vision arise when computing geometrical models from image data. They often lead to solving systems of algebraic equations.

This page provides links to publications, software, data, and evaluation of minimal problems.

NEW:

ACCV 2010

Bujnak M., Kukelova Z., Pajdla T., New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion, ACCV 2010, Queenstown, NZ, November 8-12, 2010. [[pdf](#)]

CODE:

[4-point absolute pose problem with unknown focal length and radial distortion \(P4Pfr\)](#)

Kukelova Z., Bujnak M., Pajdla T., Closed-form solutions to the minimal absolute pose problems with known vertical direction, ACCV 2010, Queenstown, NZ, November 8-12, 2010. [[pdf](#)]

CODE:

[2-point absolute pose problem with known vertical direction \(up2p\)](#)

[3-point absolute pose problem with known vertical direction and unknown focal length and radial distortion \(up3pfr\)](#)

SOURCE CODES TO SEVERAL MINIMAL PROBLEMS

[4-point absolute pose problem with unknown focal length \(P4Pf\) \(new fast Matlab version\)](#)

[8-point "uncalibrated" relative pose problem with radial distortion](#)

[4-point absolute pose problem with unknown focal length and radial distortion \(P4Pfr\)](#)

[2-point absolute pose problem with known vertical direction \(up2p\)](#)

[3-point absolute pose problem with known vertical direction and unknown focal length and radial distortion \(up3pfr\)](#)

Minimal problems:

[3-point absolute pose problem \(P3P\)](#)

[4-point absolute pose problem with unknown focal length \(P4Pf\) NEW FAST MATLAB SOURCE CODE](#)

[4-point absolute pose problem with unknown focal length and radial distortion \(P4Pfr\) NEW](#)

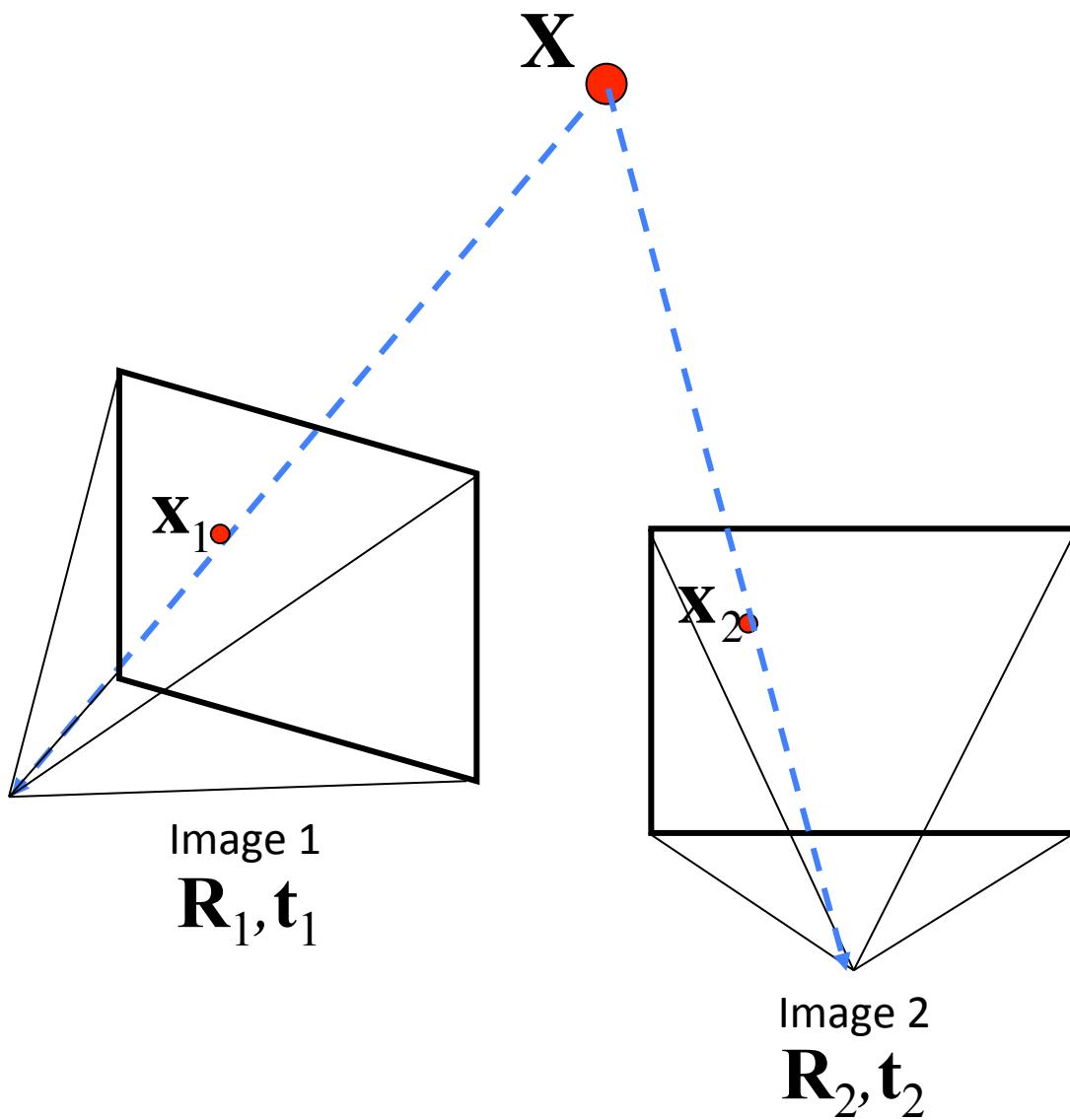
[2-point absolute pose problem with known vertical direction \(up2p\) NEW](#)

[3-point absolute pose problem with known vertical direction and unknown focal length and radial distortion \(up3pfr\) NEW](#)

[2-point + 1-line absolute pose problem NEW](#)

<http://cmp.felk.cvut.cz/minimal/>

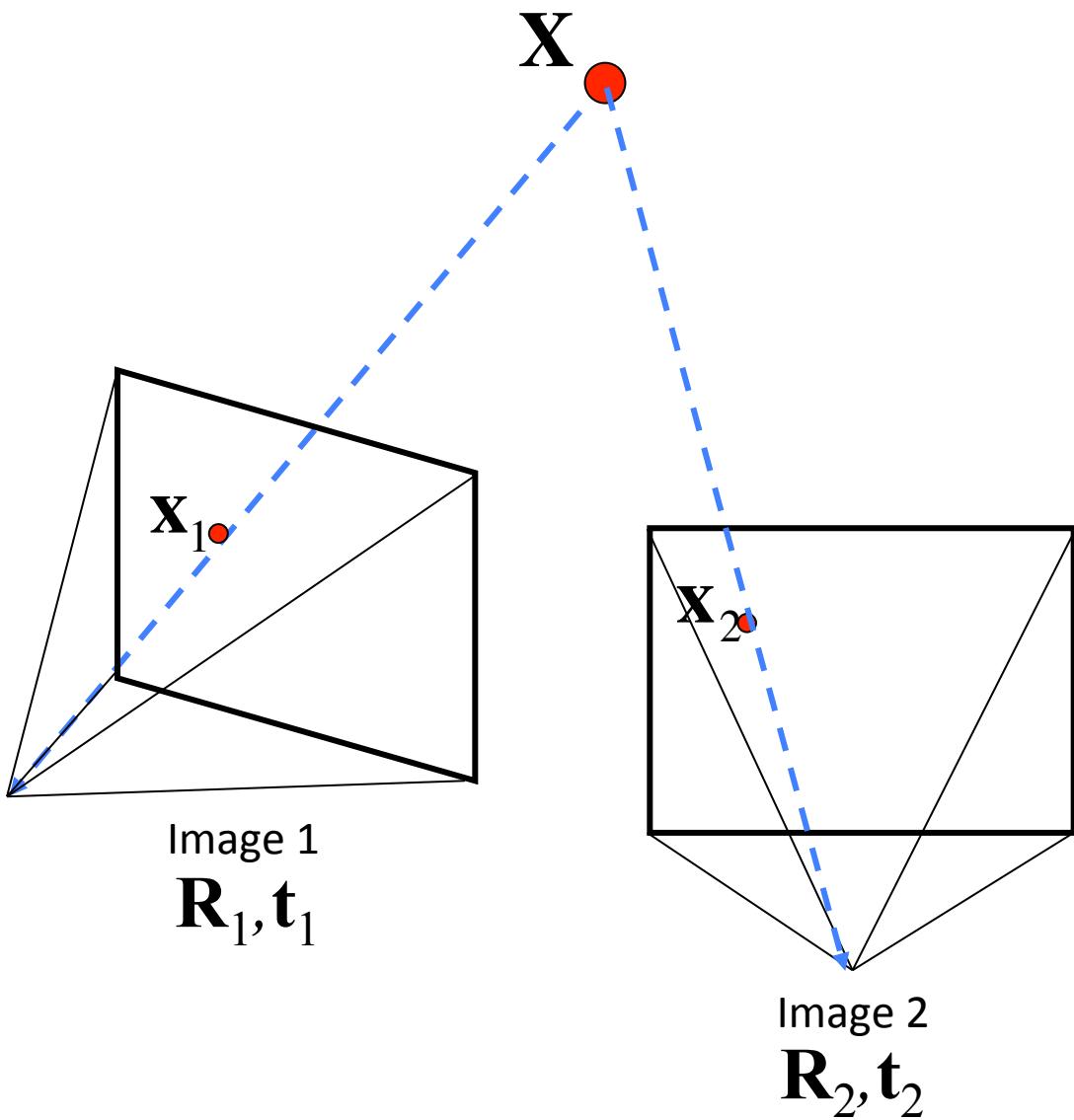
Fundamental Matrix → Essential Matrix



$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$

$$\mathbf{E} = \mathbf{K}_1^T \mathbf{F} \mathbf{K}_2$$

Essential Matrix $\rightarrow [\mathbf{R}|\mathbf{t}]$



$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$

$$\mathbf{E} = \mathbf{K}_1^T \mathbf{F} \mathbf{K}_2$$

Essential Matrix $\rightarrow [\mathbf{R}|\mathbf{t}]$

Result 9.19. For a given essential matrix

$$\mathbf{E} = \mathbf{U} \text{diag}(1,1,0) \mathbf{V}^T,$$

and the first camera matrix $\mathbf{P}_1 = [\mathbf{I}|0]$, there are four possible choices for the second camera matrix \mathbf{P}_2 :

$$\mathbf{P}_2 = [\mathbf{U} \mathbf{W} \mathbf{V}^T | +\mathbf{u}_3]$$

$$\mathbf{P}_2 = [\mathbf{U} \mathbf{W} \mathbf{V}^T | -\mathbf{u}_3]$$

$$\mathbf{P}_2 = [\mathbf{U} \mathbf{W}^T \mathbf{V}^T | +\mathbf{u}_3]$$

$$\mathbf{P}_2 = [\mathbf{U} \mathbf{W}^T \mathbf{V}^T | -\mathbf{u}_3]$$

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Four Possible Solutions

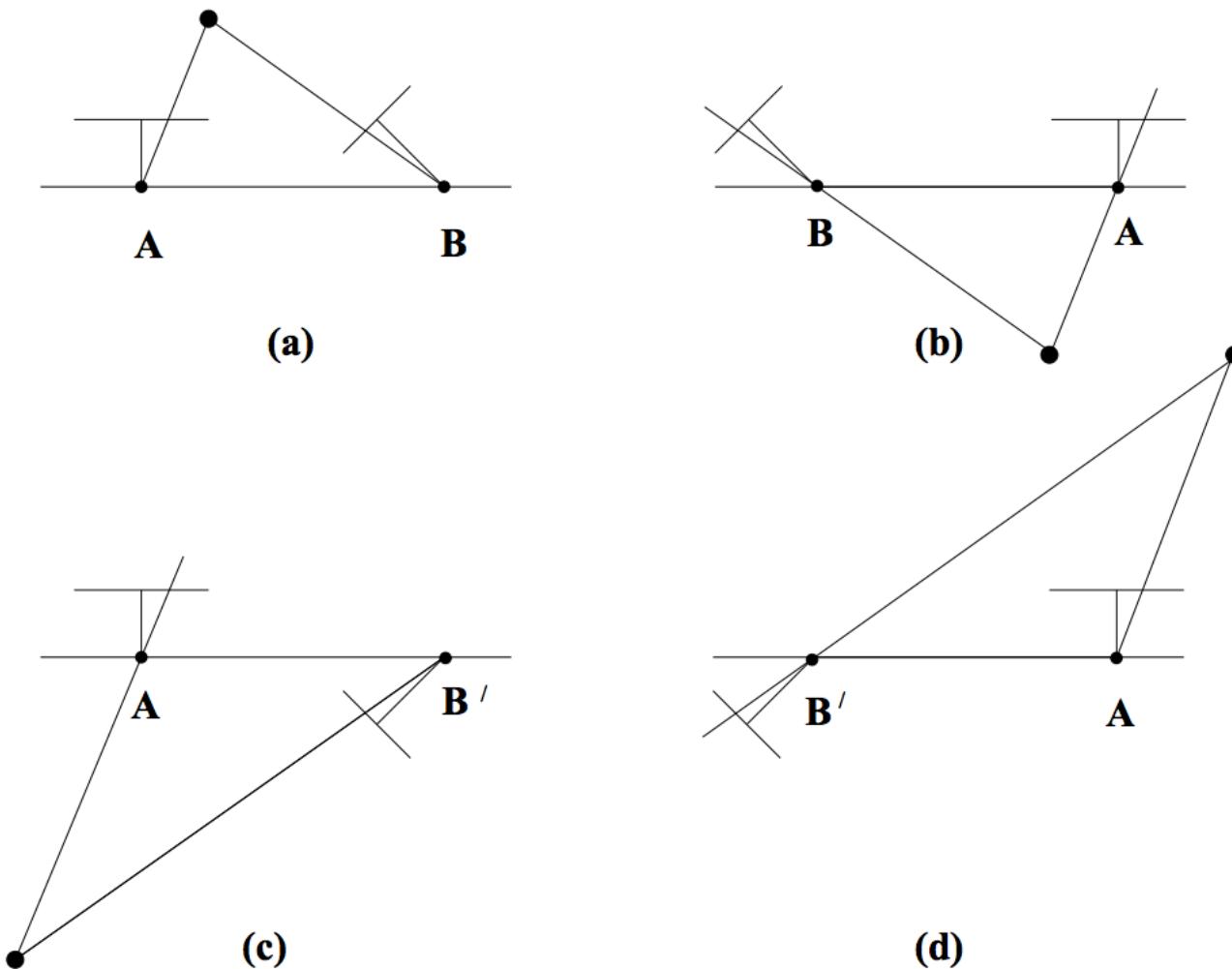
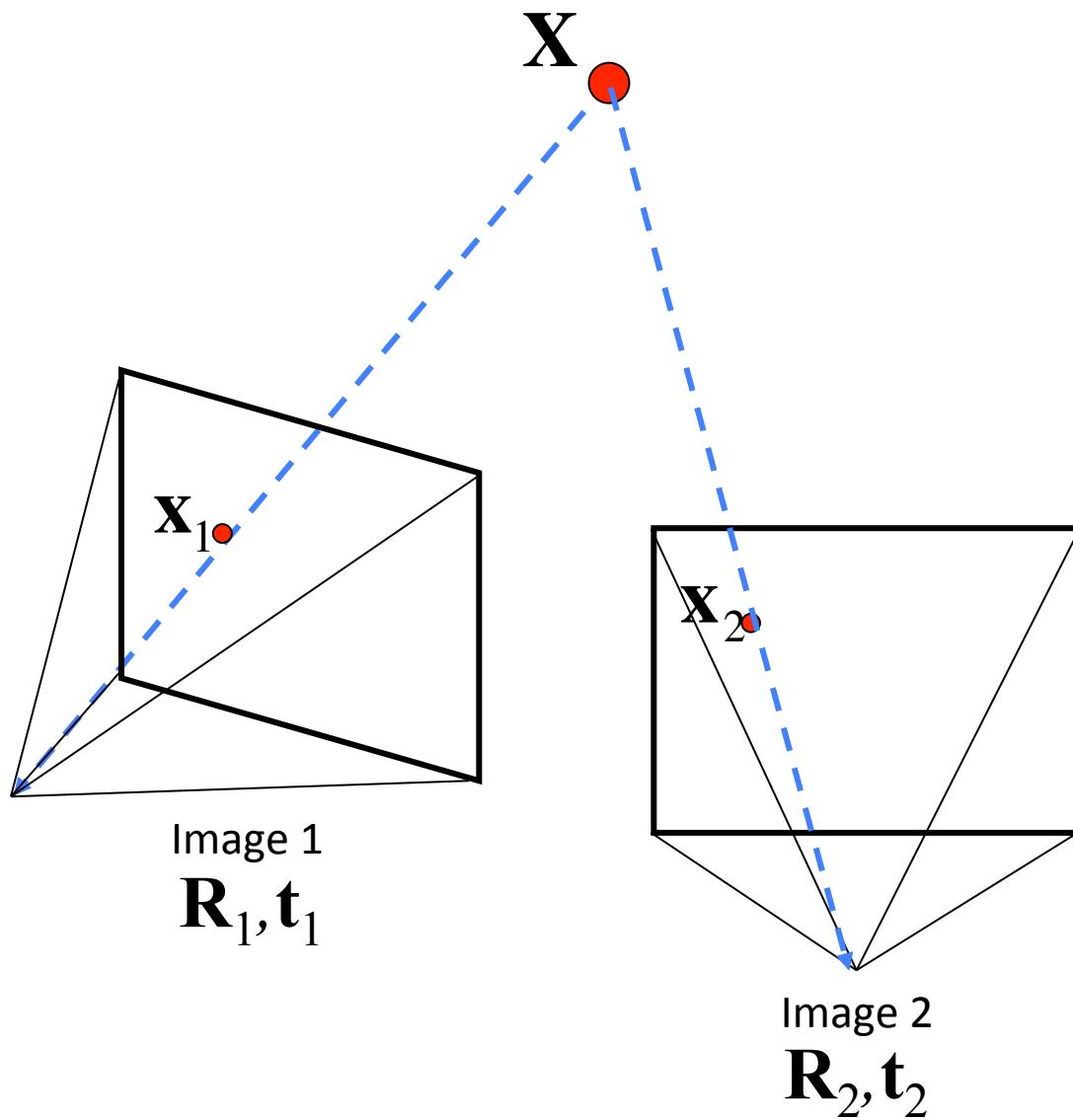


Fig. 9.12. The four possible solutions for calibrated reconstruction from E. Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.

Triangulation



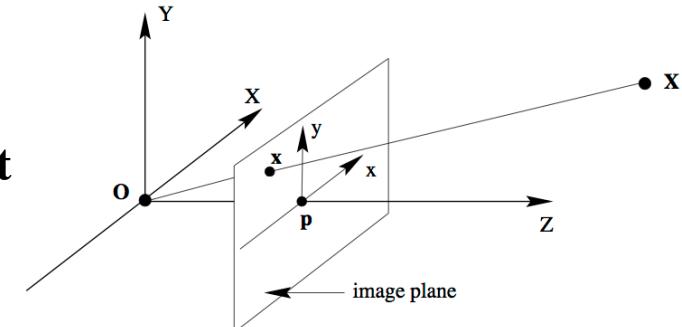
In front of the camera?

- Camera Extrinsic $[\mathbf{R}|\mathbf{t}]$

$$\begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \end{bmatrix} + \mathbf{t} \quad \leftrightarrow \quad \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \end{bmatrix} = \mathbf{R}^{-1} \left(\begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} - \mathbf{t} \right) = \mathbf{R}^T \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} - \mathbf{R}^T \mathbf{t}$$

- Camera Center

$$\begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \rightarrow \quad \mathbf{C} = \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \end{bmatrix} = \mathbf{R}^T \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \mathbf{R}^T \mathbf{t} = -\mathbf{R}^T \mathbf{t}$$



- View Direction

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \rightarrow \quad \left(\mathbf{R}^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \mathbf{R}^T \mathbf{t} \right) - (\mathbf{C}) = \left(\mathbf{R}(3,:)^T - \mathbf{R}^T \mathbf{t} \right) - \left(-\mathbf{R}^T \mathbf{t} \right) = \mathbf{R}(3,:)^T$$

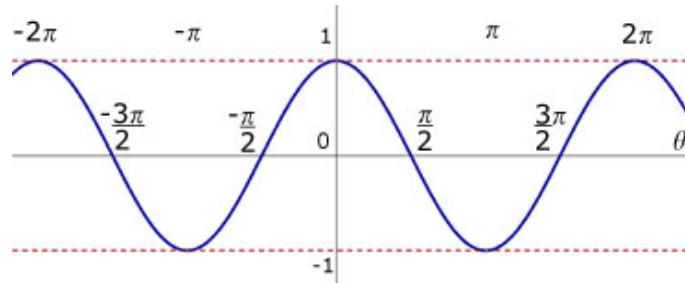
Camera Coordinate System

World Coordinate System

In front of the camera?

- A point \mathbf{X}
- Direction from camera center to point $\mathbf{X} - \mathbf{C}$
- Angle Between Two Vectors

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos \theta$$



- Angle Between $\mathbf{X} - \mathbf{C}$ and View Direction
- Just need to test

$$(\mathbf{X} - \mathbf{C}) \cdot \mathbf{R}(3,:)^T > 0 ?$$

Pick the Solution

With maximal number of points in front of both cameras.

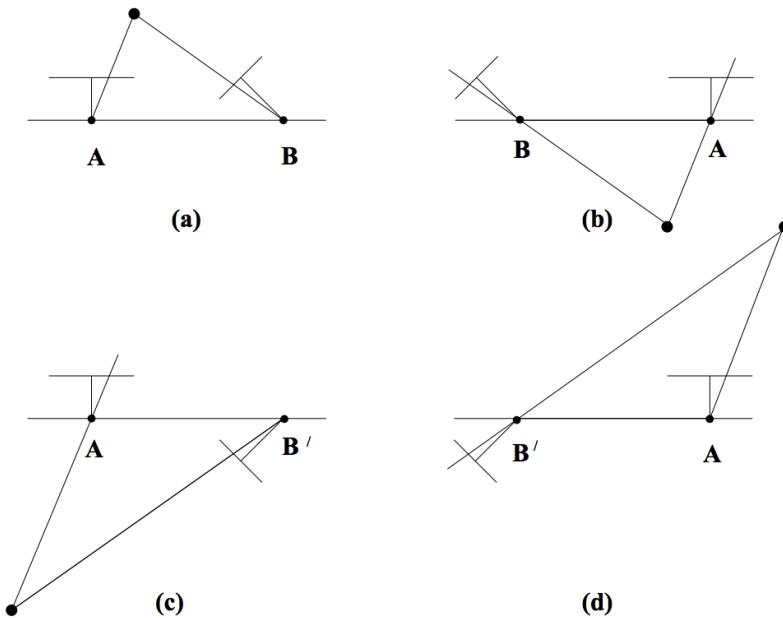
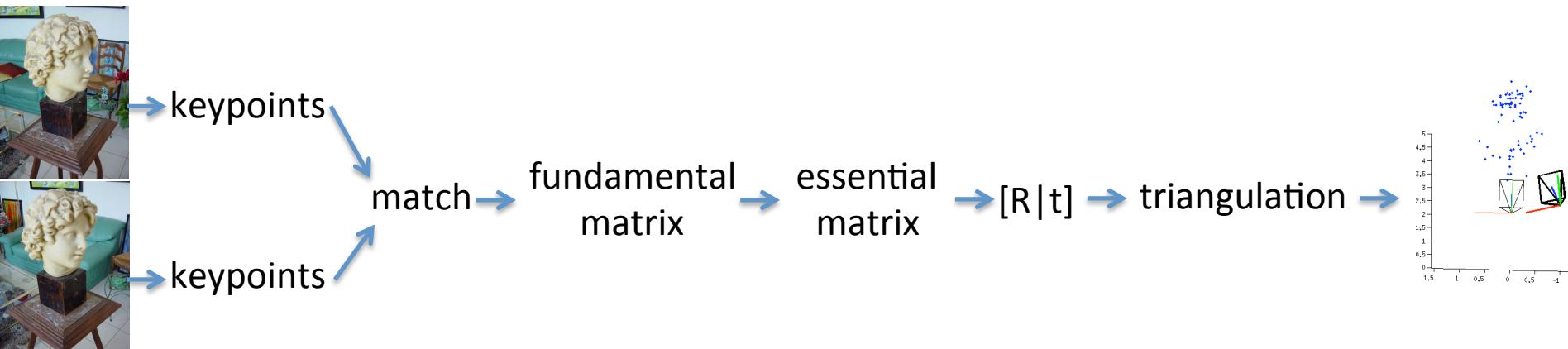
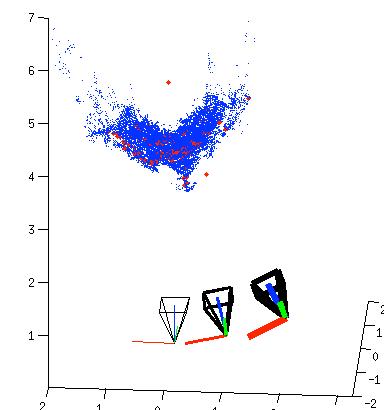
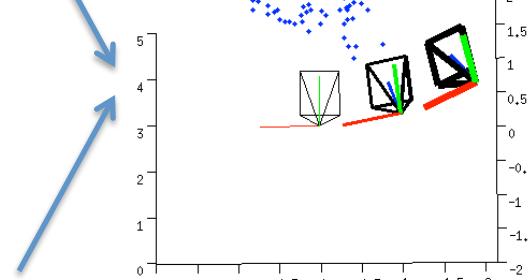
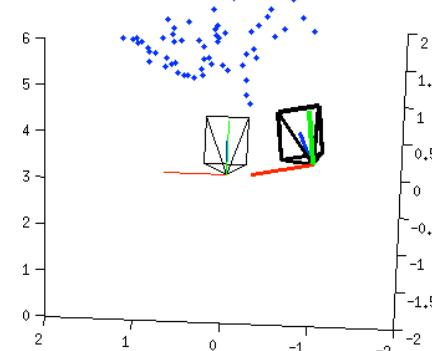
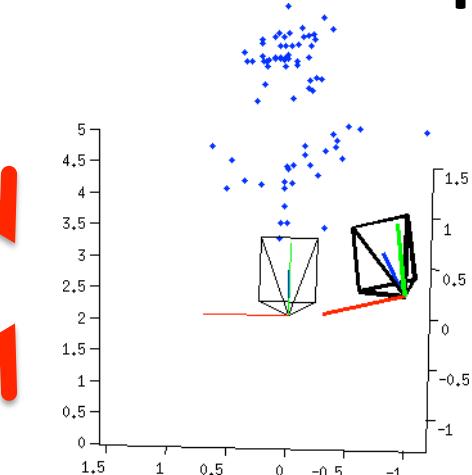


Fig. 9.12. The four possible solutions for calibrated reconstruction from E. Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.

Two-view Reconstruction



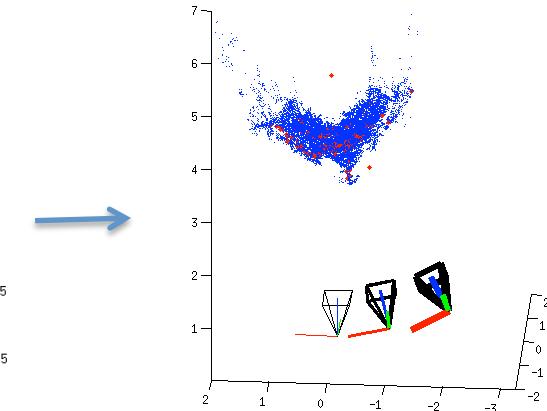
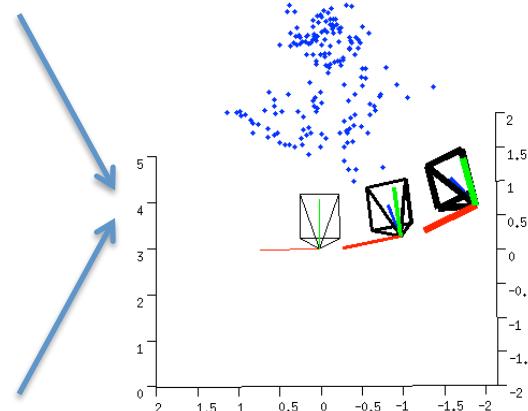
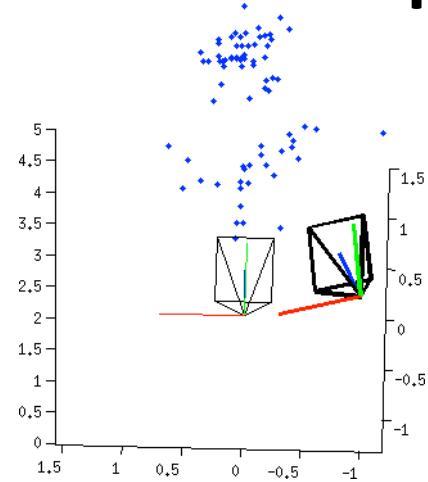
Pipeline



Structure from Motion (SfM)

Multi-view Stereo (MVS)

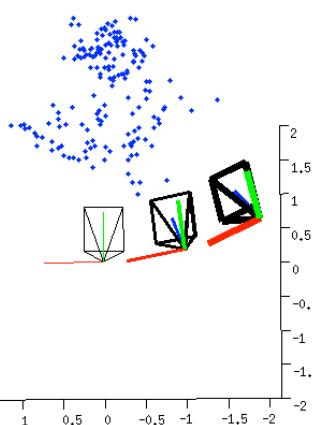
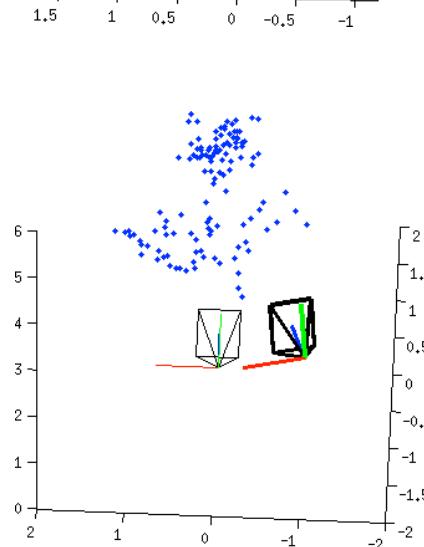
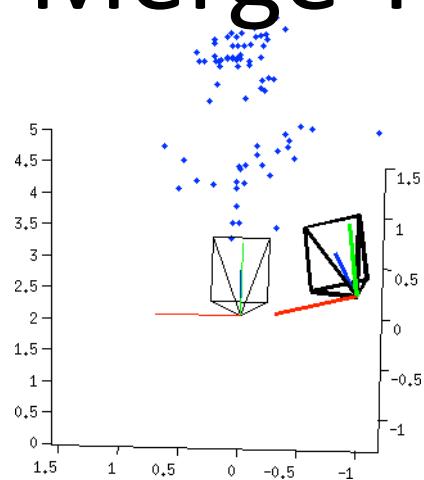
Pipeline



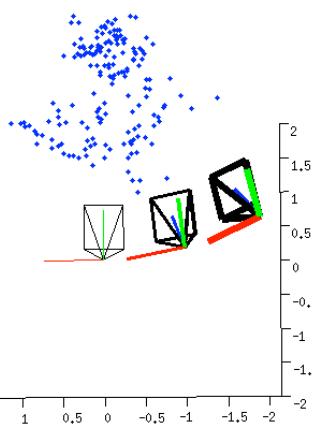
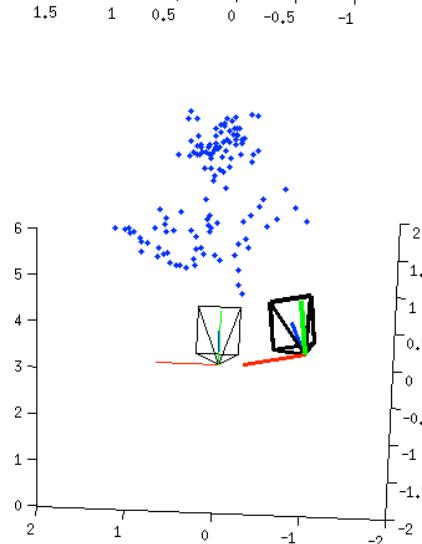
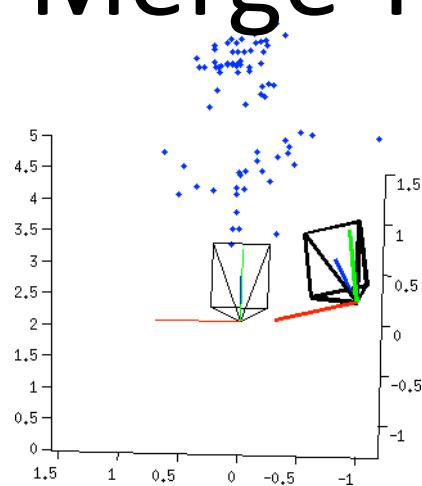
↓
Taught

↓
Next

Merge Two Point Cloud



Merge Two Point Cloud

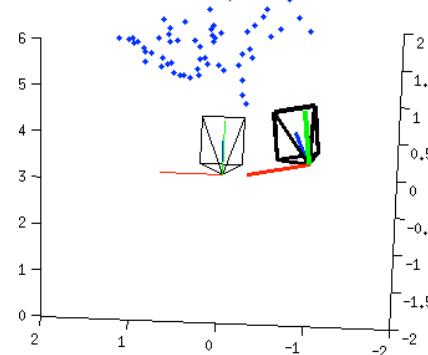
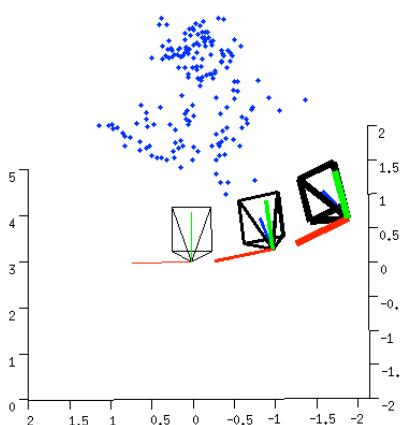
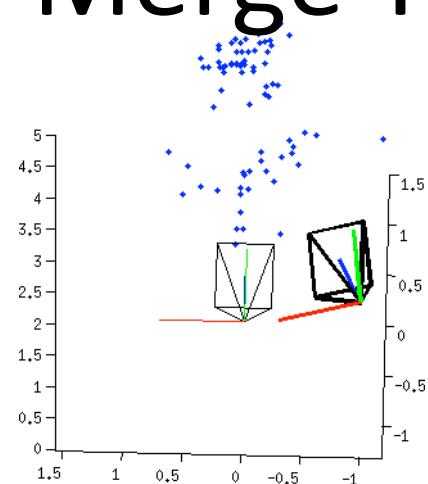


There can be only one $[\mathbf{R}_2 | \mathbf{t}_2]$

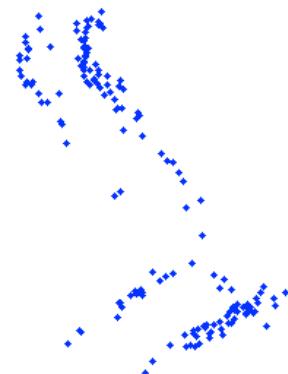
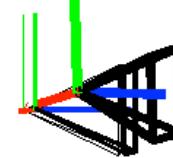
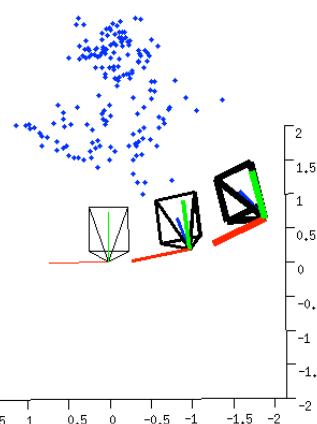
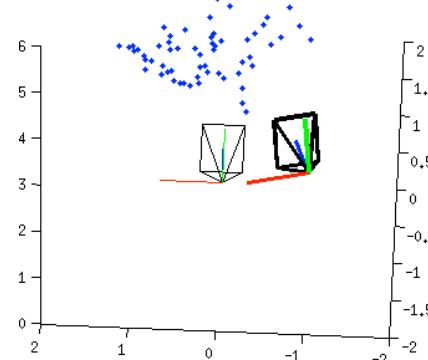
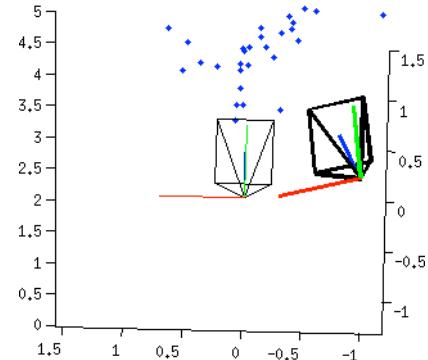
Merge Two Point Cloud

- From the 1st and 2nd images, we have $[\mathbf{R}_1 | \mathbf{t}_1]$ and $[\mathbf{R}_2 | \mathbf{t}_2]$
- From the 2nd and 3rd images, we have $[\mathbf{R}_2 | \mathbf{t}_2]$ and $[\mathbf{R}_3 | \mathbf{t}_3]$
- **Exercise:** How to transform the coordinate system of the second point cloud to align with the first point cloud so that there is only one $[\mathbf{R}_2 | \mathbf{t}_2]$?

Merge Two Point Cloud

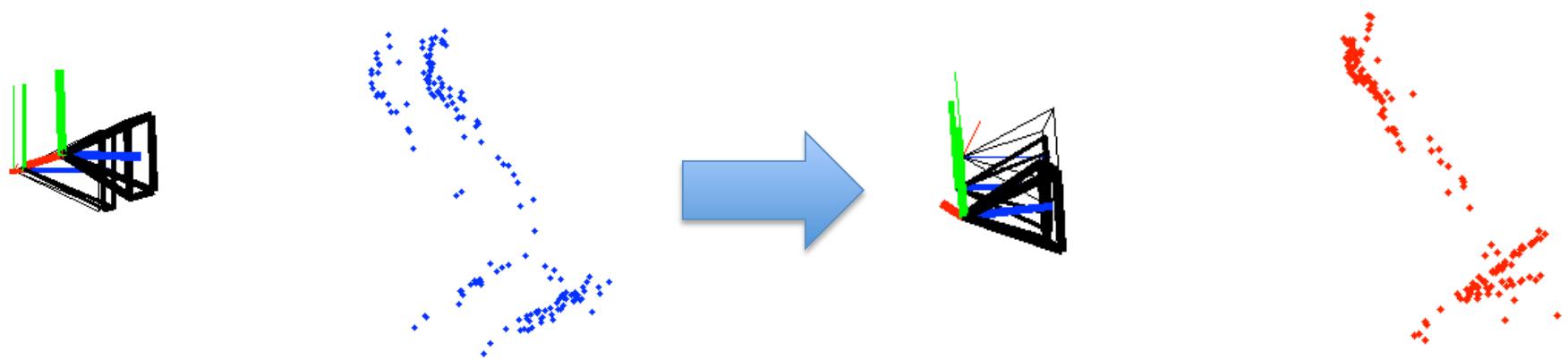


Oops

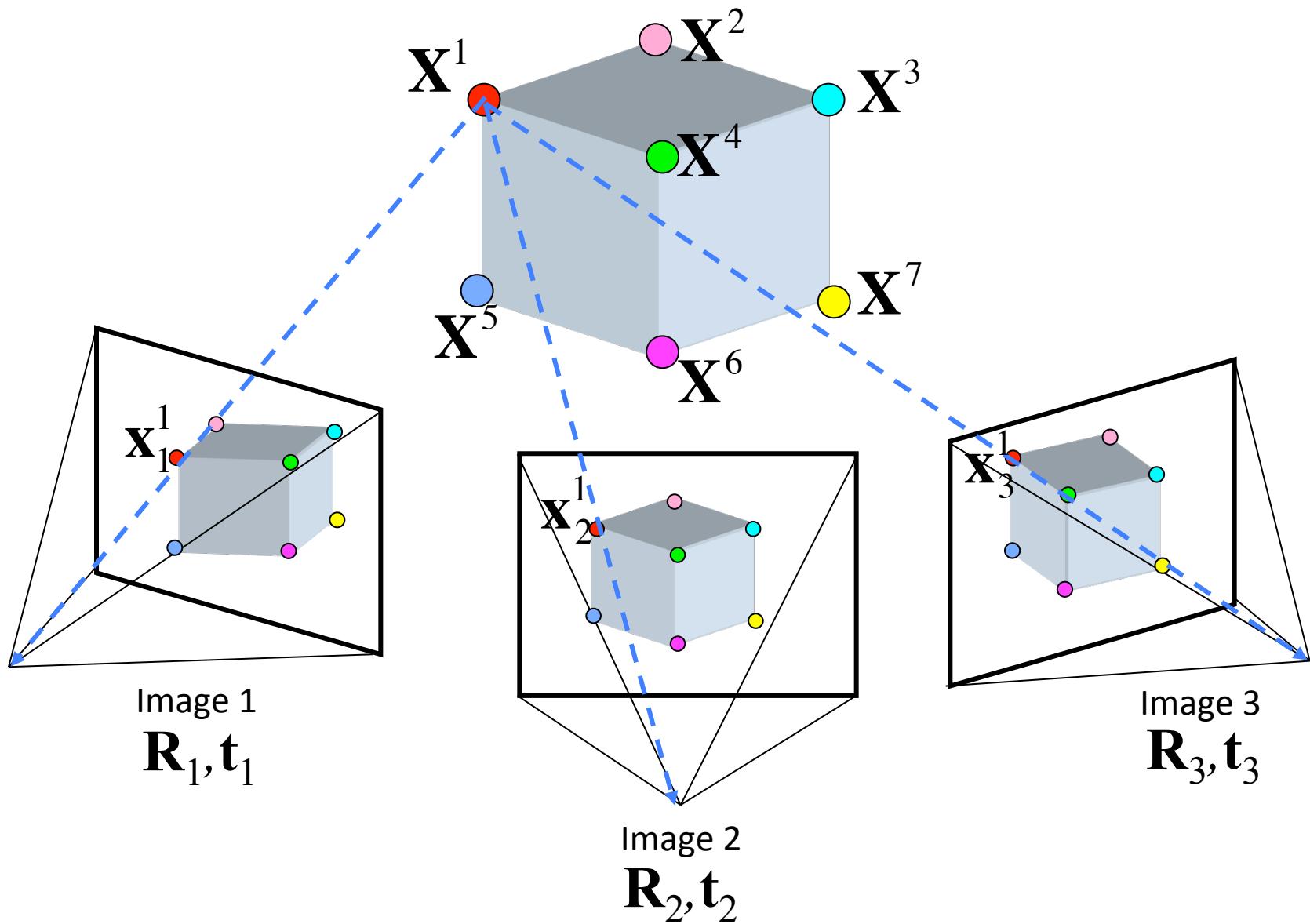


See From a Different Angle

Bundle Adjustment



“America’s Next Top Model”



“America's Next Top Model”

	Point 1	Point 2	Point 3
Image 1	$\mathbf{x}_1^1 = \mathbf{K}[\mathbf{R}_1 \mathbf{t}_1] \mathbf{X}^1$	$\mathbf{x}_1^2 = \mathbf{K}[\mathbf{R}_1 \mathbf{t}_1] \mathbf{X}^2$	
Image 2	$\mathbf{x}_2^1 = \mathbf{K}[\mathbf{R}_2 \mathbf{t}_2] \mathbf{X}^1$	$\mathbf{x}_2^2 = \mathbf{K}[\mathbf{R}_2 \mathbf{t}_2] \mathbf{X}^2$	$\mathbf{x}_2^3 = \mathbf{K}[\mathbf{R}_2 \mathbf{t}_2] \mathbf{X}^3$
Image 3	$\mathbf{x}_3^1 = \mathbf{K}[\mathbf{R}_3 \mathbf{t}_3] \mathbf{X}^1$		$\mathbf{x}_3^3 = \mathbf{K}[\mathbf{R}_3 \mathbf{t}_3] \mathbf{X}^3$

Rethinking the SFM problem

- Input: Observed 2D image position

$$\tilde{\mathbf{x}}_1^1 \quad \tilde{\mathbf{x}}_1^2$$

$$\tilde{\mathbf{x}}_2^1 \quad \tilde{\mathbf{x}}_2^2 \quad \tilde{\mathbf{x}}_2^3$$

- Output:

$$\tilde{\mathbf{x}}_3^1 \quad \tilde{\mathbf{x}}_3^3$$

Unknown Camera Parameters (with some guess)

$$[\mathbf{R}_1|\mathbf{t}_1], [\mathbf{R}_2|\mathbf{t}_2], [\mathbf{R}_3|\mathbf{t}_3]$$

Unknown Point 3D coordinate (with some guess)

$$\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \dots$$

Bundle Adjustment

A valid solution $[\mathbf{R}_1|\mathbf{t}_1], [\mathbf{R}_2|\mathbf{t}_2], [\mathbf{R}_3|\mathbf{t}_3]$ and $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \dots$

must let

Re-projection $\left[\begin{array}{ll} \mathbf{x}_1^1 = \mathbf{K}[\mathbf{R}_1|\mathbf{t}_1]\mathbf{X}^1 & \mathbf{x}_1^2 = \mathbf{K}[\mathbf{R}_1|\mathbf{t}_1]\mathbf{X}^2 \\ \mathbf{x}_2^1 = \mathbf{K}[\mathbf{R}_2|\mathbf{t}_2]\mathbf{X}^1 & \mathbf{x}_2^2 = \mathbf{K}[\mathbf{R}_2|\mathbf{t}_2]\mathbf{X}^2 & \mathbf{x}_2^3 = \mathbf{K}[\mathbf{R}_2|\mathbf{t}_2]\mathbf{X}^3 \\ \mathbf{x}_3^1 = \mathbf{K}[\mathbf{R}_3|\mathbf{t}_3]\mathbf{X}^1 & & \mathbf{x}_3^3 = \mathbf{K}[\mathbf{R}_3|\mathbf{t}_3]\mathbf{X}^3 \end{array} \right]$

$=$

Observation $\left[\begin{array}{ll} \tilde{\mathbf{x}}_1^1 & \tilde{\mathbf{x}}_1^2 \\ \tilde{\mathbf{x}}_2^1 & \tilde{\mathbf{x}}_2^2 & \tilde{\mathbf{x}}_2^3 \\ \tilde{\mathbf{x}}_3^1 & & \tilde{\mathbf{x}}_3^3 \end{array} \right]$

Bundle Adjustment

A valid solution $[\mathbf{R}_1|\mathbf{t}_1], [\mathbf{R}_2|\mathbf{t}_2], [\mathbf{R}_3|\mathbf{t}_3]$ and $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \dots$ must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

$$\min \sum_i \sum_j \left(\tilde{\mathbf{x}}_i^j - \mathbf{K}[\mathbf{R}_i|\mathbf{t}_i] \mathbf{X}^j \right)^2$$

Bundle Adjustment

A valid solution $[\mathbf{R}_1|\mathbf{t}_1], [\mathbf{R}_2|\mathbf{t}_2], [\mathbf{R}_3|\mathbf{t}_3]$ and $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \dots$ must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

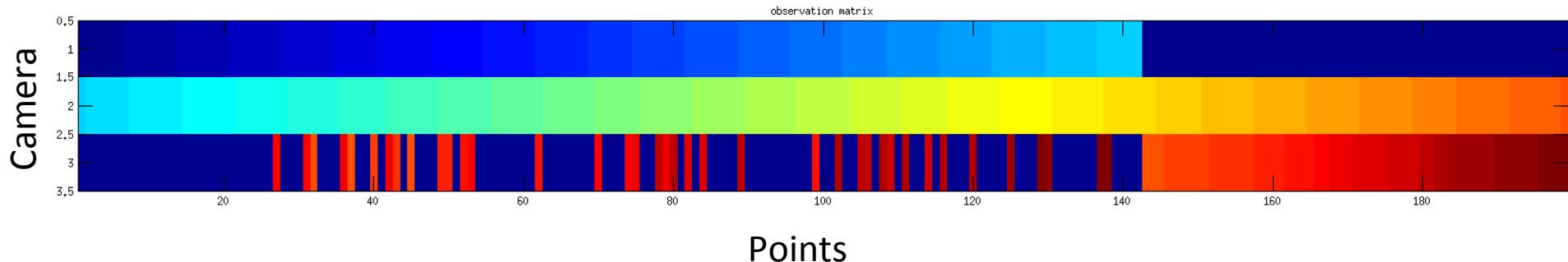
$$\min \sum_i \sum_j \left(\tilde{\mathbf{x}}_i^j - \mathbf{K}[\mathbf{R}_i|\mathbf{t}_i] \mathbf{X}^j \right)^2$$

Question: What is the unit of this objective function?

Bundle Adjustment

A valid solution $[\mathbf{R}_1|\mathbf{t}_1], [\mathbf{R}_2|\mathbf{t}_2], [\mathbf{R}_3|\mathbf{t}_3]$ and $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \dots$ must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

$$\min \sum_i \sum_j \left(\tilde{\mathbf{x}}_i^j - \mathbf{K} [\mathbf{R}_i | \mathbf{t}_i] \mathbf{X}^j \right)^2$$



Linking

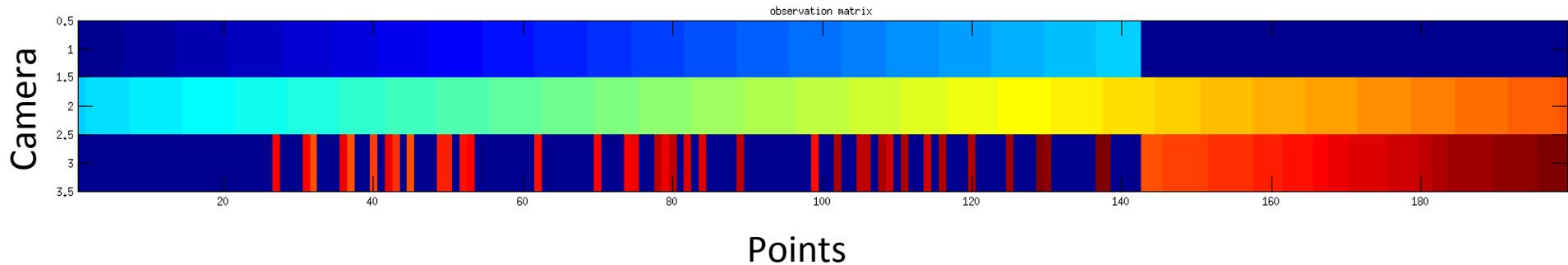


Linking



SIFT Matching

SIFT Matching



Solving This Optimization Problem

- Theory:

The Levenberg–Marquardt algorithm

http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm

- Practice:

The Ceres-Solver from Google

<http://code.google.com/p/ceres-solver/>

Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve $\min(10 - x)^2$

```
class SimpleCostFunction
    : public ceres::SizedCostFunction<1 /* number of residuals */,
                                         1 /* size of first parameter */ {
public:
    virtual ~SimpleCostFunction() {}
    virtual bool Evaluate(double const* const* parameters,
                          double* residuals,
                          double** jacobians) const {
        const double x = parameters[0][0];
        residuals[0] = 10 - x; // f(x) = 10 - x
        // Compute the Jacobian if asked for.
        if (jacobians != NULL && jacobians[0] != NULL) {
            jacobians[0][0] = -1;
        }
        return true;
    }
};
```

Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve $\min(10 - x)^2$

```
int main(int argc, char** argv) {
    double x = 5.0;
    ceres::Problem problem;

    // The problem object takes ownership of the newly allocated
    // SimpleCostFunction and uses it to optimize the value of x.
    problem.AddResidualBlock(new SimpleCostFunction, NULL, &x);

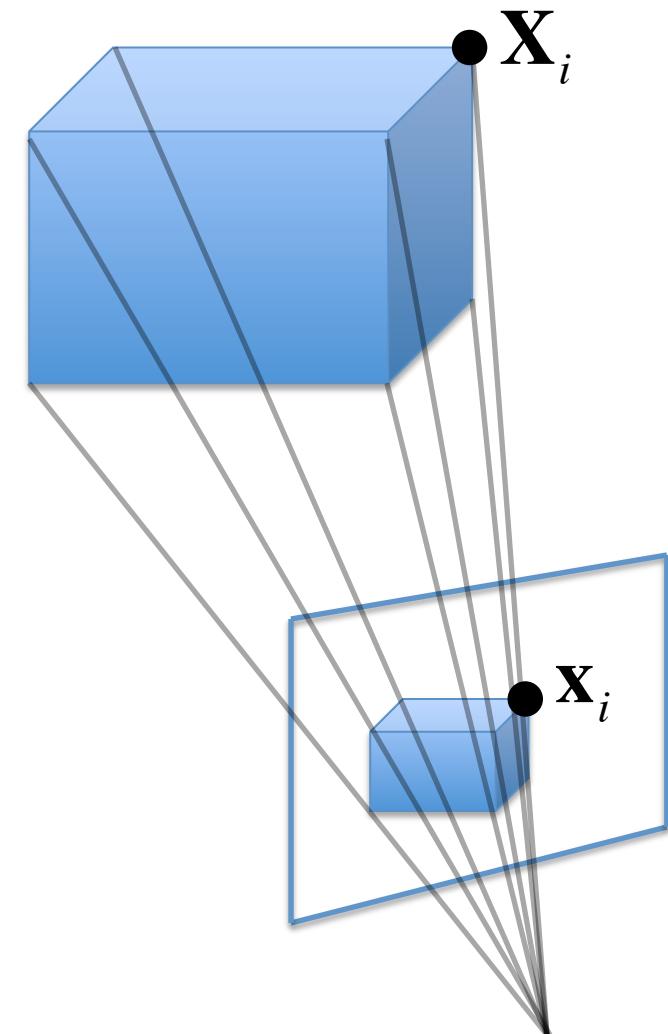
    // Run the solver!
    Solver::Options options;
    options.max_num_iterations = 10;
    options.linear_solver_type = ceres::DENSE_QR;
    options.minimizer_progress_to_stdout = true;
    Solver::Summary summary;
    Solve(options, &problem, &summary);
    std::cout << summary.BriefReport() << "\n";
    std::cout << "x : 5.0 -> " << x << "\n";
    return 0;
}
```

Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve $\min(10 - x)^2$

```
0: f: 1.250000e+01 d: 0.00e+00 g: 5.00e+00 h: 0.00e+00 rho: 0.00e+00 mu: 1.00e-04 li: 0
1: f: 1.249750e-07 d: 1.25e+01 g: 5.00e-04 h: 5.00e+00 rho: 1.00e+00 mu: 3.33e-05 li: 1
2: f: 1.388518e-16 d: 1.25e-07 g: 1.67e-08 h: 5.00e-04 rho: 1.00e+00 mu: 1.11e-05 li: 1
Ceres Solver Report: Iterations: 2, Initial cost: 1.250000e+01, \
Final cost: 1.388518e-16, Termination: PARAMETER_TOLERANCE.
x : 5 -> 10
```

Non-linear Least Square Cuboid Reconstruction



Observation:

Pixel location of the 7 (or 6) corners

Parameters to be estimated:

Cuboid: Height $h \times$ Width $w \times 1$

Camera: Intrinsic K and Extrinsic R, t

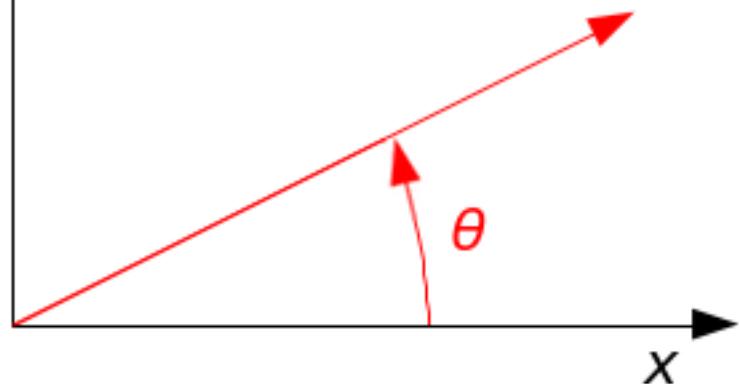
$$\min \sum_i \|\mathbf{x}_i - \mathbf{K}(\mathbf{R}\mathbf{X}_i + \mathbf{t})\|^2$$

$$\mathbf{X} = \begin{bmatrix} h & h & h & h & -h & -h & -h & -h \\ w & w & -w & -w & w & w & -w & -w \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

Parameterizing Rotation Matrix

- 2D Rotation

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



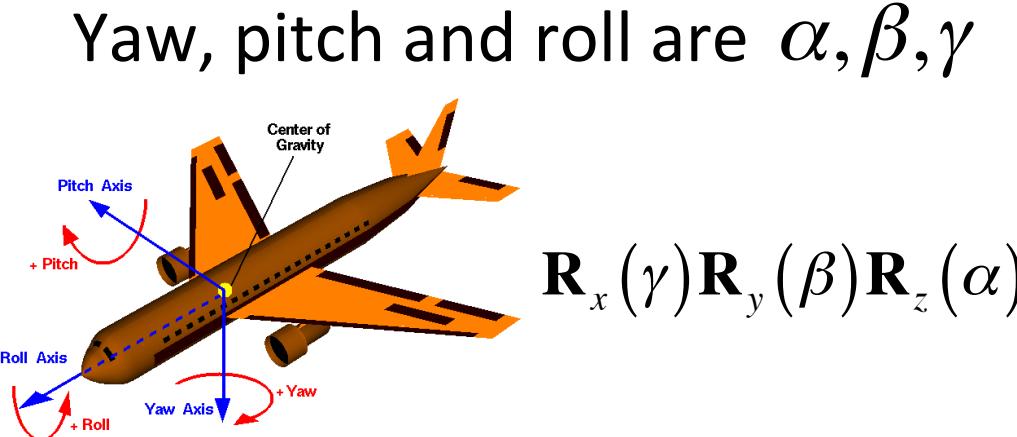
$$\mathbf{R}^T = \mathbf{R}^{-1}, \det \mathbf{R} = 1$$

3D Rotation

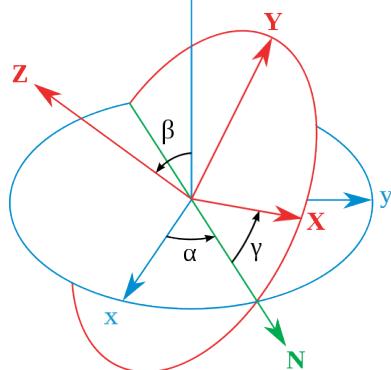
$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



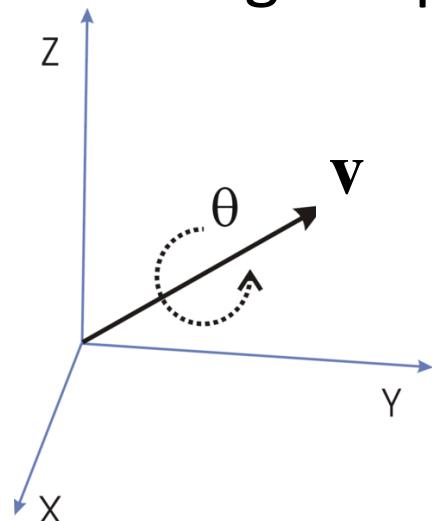
Euler angles are α, β, γ



$$\mathbf{R}_z(\gamma)\mathbf{R}_x(\beta)\mathbf{R}_y(\alpha)$$

3D Rotation

Axis-angle representation



Quaternions

$$\mathbf{q} = \left(\mathbf{v} \sin\left(\frac{\theta}{2}\right), \cos\left(\frac{\theta}{2}\right) \right)^T$$

Avoid Gimbal Lock!

Recommendation!

Triplet Representation $\mathbf{v}\theta$ (3 dof) ← Not over-parameterized

Rodrigues' rotation formula

$$\mathbf{k}_{rot} = \mathbf{k} \cos \theta + (\mathbf{v} \times \mathbf{k}) \sin \theta + \mathbf{v} (\mathbf{v} \cdot \mathbf{k}) (1 - \cos \theta)$$

http://en.wikipedia.org/wiki/Axis-angle_representation

http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

http://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula

Matlab Symbolic Operation

How to derive complicated equations
(after passing math classes)?

```
syms h w K alpha beta gamma tx ty tz
X = [h; w; f];
Y = [...] * X;
Z = K* Y;
x = Z(1:2,:)/ Z([3,3],:);
ccode(x,'file','x.cpp'); % ←matlab will generate C code for x
```

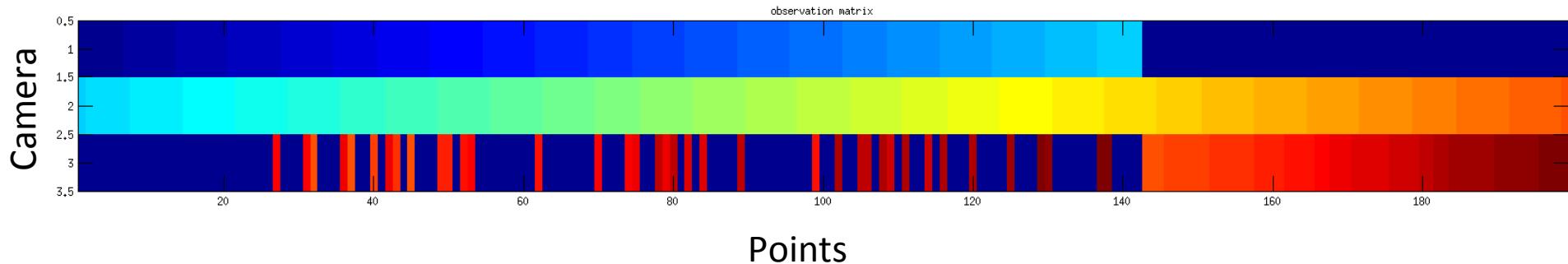
<http://www.mathworks.com/help/symbolic/sym.html>

<http://mit.edu/jxiao/Public/software/fitCuboid/fitCuboid/derive4cuboid.m>

Ceres for Bundle Adjustment

A valid solution $[\mathbf{R}_1|\mathbf{t}_1], [\mathbf{R}_2|\mathbf{t}_2], [\mathbf{R}_3|\mathbf{t}_3]$ and $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \dots$ must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

$$\min \sum_i \sum_j \left(\tilde{\mathbf{x}}_i^j - \mathbf{K} [\mathbf{R}_i | \mathbf{t}_i] \mathbf{X}^j \right)^2$$



ba2D.cc

```
struct AlignmentError2D {
    AlignmentError2D(double* observed_in): observed(observed_in) {}
    template <typename T>
    bool operator()(const T* const camera_extrinsic,
                     const T* const point,
                     T* residuals) const {
        T p[3];
        // camera_extrinsic[0,1,2] are the angle-axis rotation.
        ceres::AngleAxisRotatePoint(camera_extrinsic, point, p); ← RX
        // camera_extrinsic[3,4,5] are the translation.
        p[0] += camera_extrinsic[3];
        p[1] += camera_extrinsic[4];
        p[2] += camera_extrinsic[5]; ← RX + t = [R|t]X

        // let p[2] ~ 0
        if (T(0.0) <= p[2]){
            if(p[2]<EPS){
                p[2] = EPS; ← To avoid divide by 0
            }
        }else{
            if (p[2]>-EPS){
                p[2] = -EPS;
            }
        }
        // project it
        p[0] = T(fx) * p[0] / p[2] + T(px);
        p[1] = T(fy) * p[1] / p[2] + T(py); ← x = K[R|t]X
        // reprojection error
        residuals[0] = p[0] - T(observed[0]);
        residuals[1] = p[1] - T(observed[1]);
        return true;
    }
    double* observed;
};
```

$$\begin{aligned} \text{RX} \\ \text{RX} + \mathbf{t} = [\mathbf{R} | \mathbf{t}] \mathbf{X} \\ \text{To avoid divide by 0} \\ \mathbf{x} = \mathbf{K}[\mathbf{R} | \mathbf{t}] \mathbf{X} \\ \mathbf{x} - \tilde{\mathbf{X}} \end{aligned}$$

Initialization Matters

- Input: Observed 2D image position
- Output:

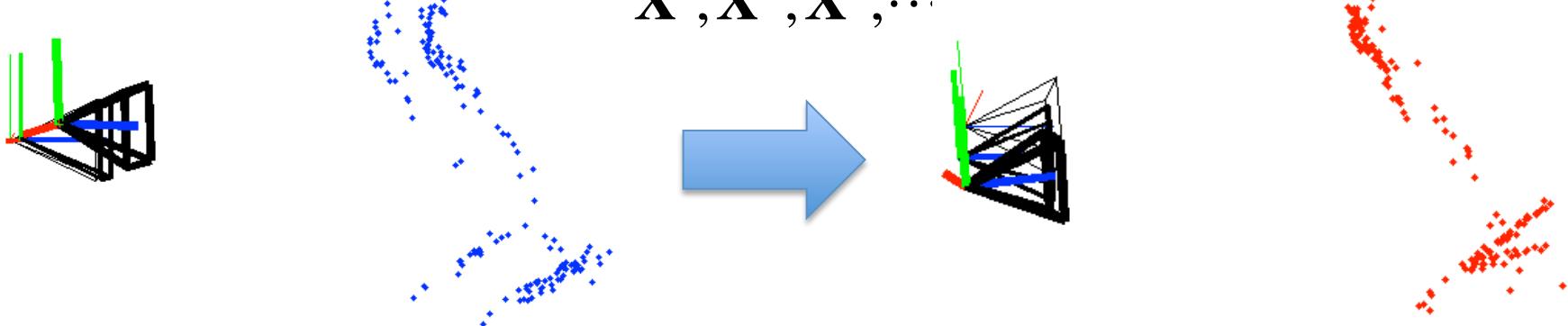
$$\begin{array}{cc} \tilde{\mathbf{x}}_1^1 & \tilde{\mathbf{x}}_1^2 \\ \tilde{\mathbf{x}}_2^1 & \tilde{\mathbf{x}}_2^2 & \tilde{\mathbf{x}}_2^3 \\ \tilde{\mathbf{x}}_3^1 & \tilde{\mathbf{x}}_3^3 \end{array}$$

Unknown Camera Parameters (with some guess)

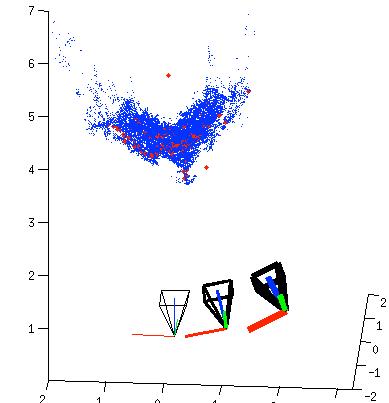
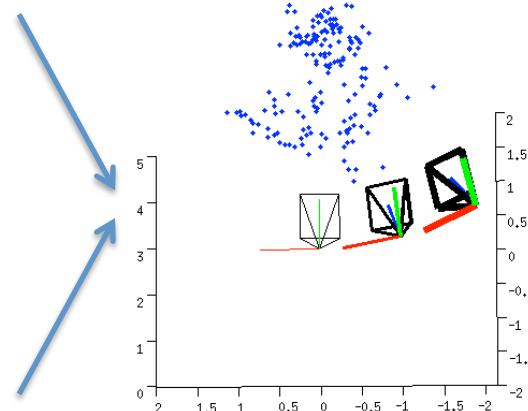
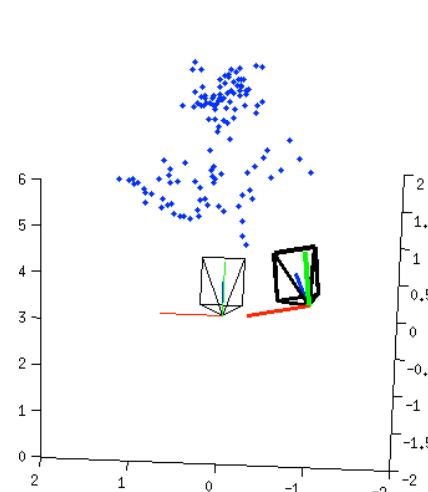
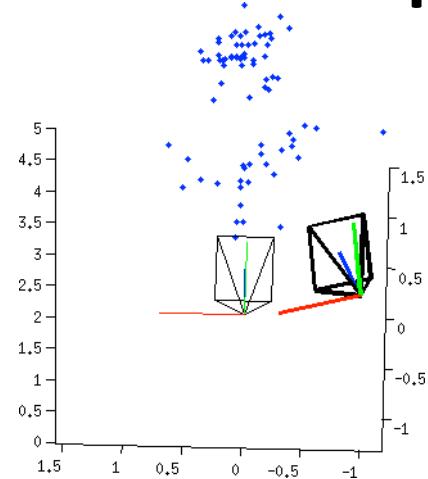
$$[\mathbf{R}_1|\mathbf{t}_1], [\mathbf{R}_2|\mathbf{t}_2], [\mathbf{R}_3|\mathbf{t}_3]$$

Unknown Point 3D coordinate (with some guess)

$$\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \dots$$



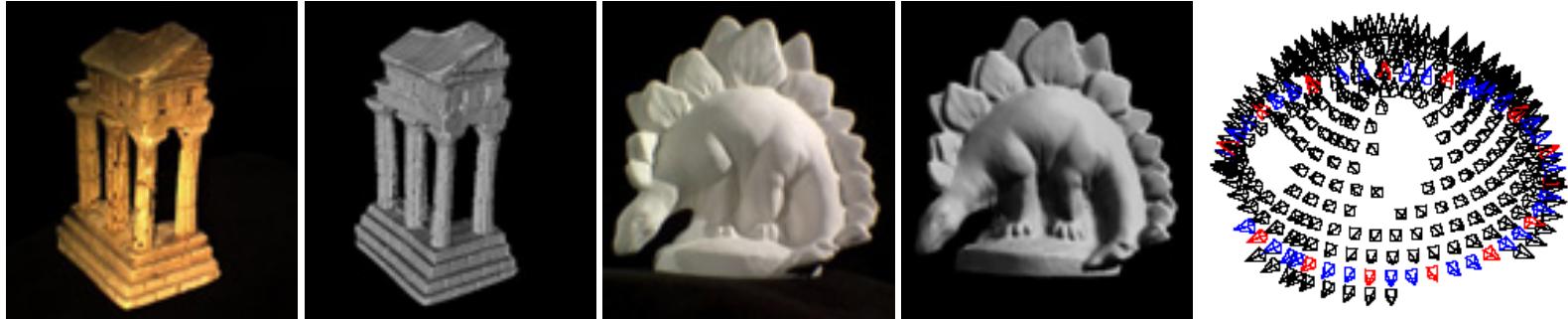
Pipeline



Taught

Next

Multiple View Stereo



State-of-the-art:

PMVS: <http://grail.cs.washington.edu/software/pmv/>

Accurate, Dense, and Robust Multi-View Stereopsis, Y Furukawa and J Ponce, 2007.

Benchmark:

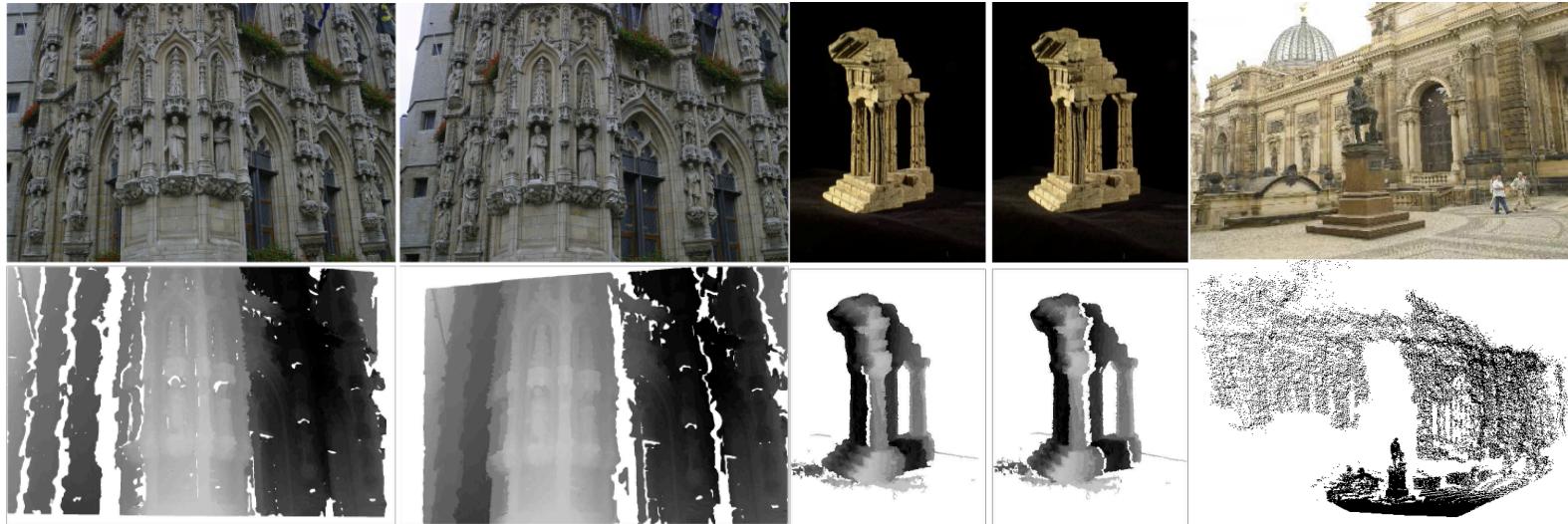
<http://vision.middlebury.edu/mview/>

A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms.
SM Seitz, B Curless, J Diebel, D Scharstein, R Szeliski. 2006.

Baseline:

Multi-view stereo revisited. M Goesele, B Curless, SM Seitz. 2006.

Key idea: Matching Propagation



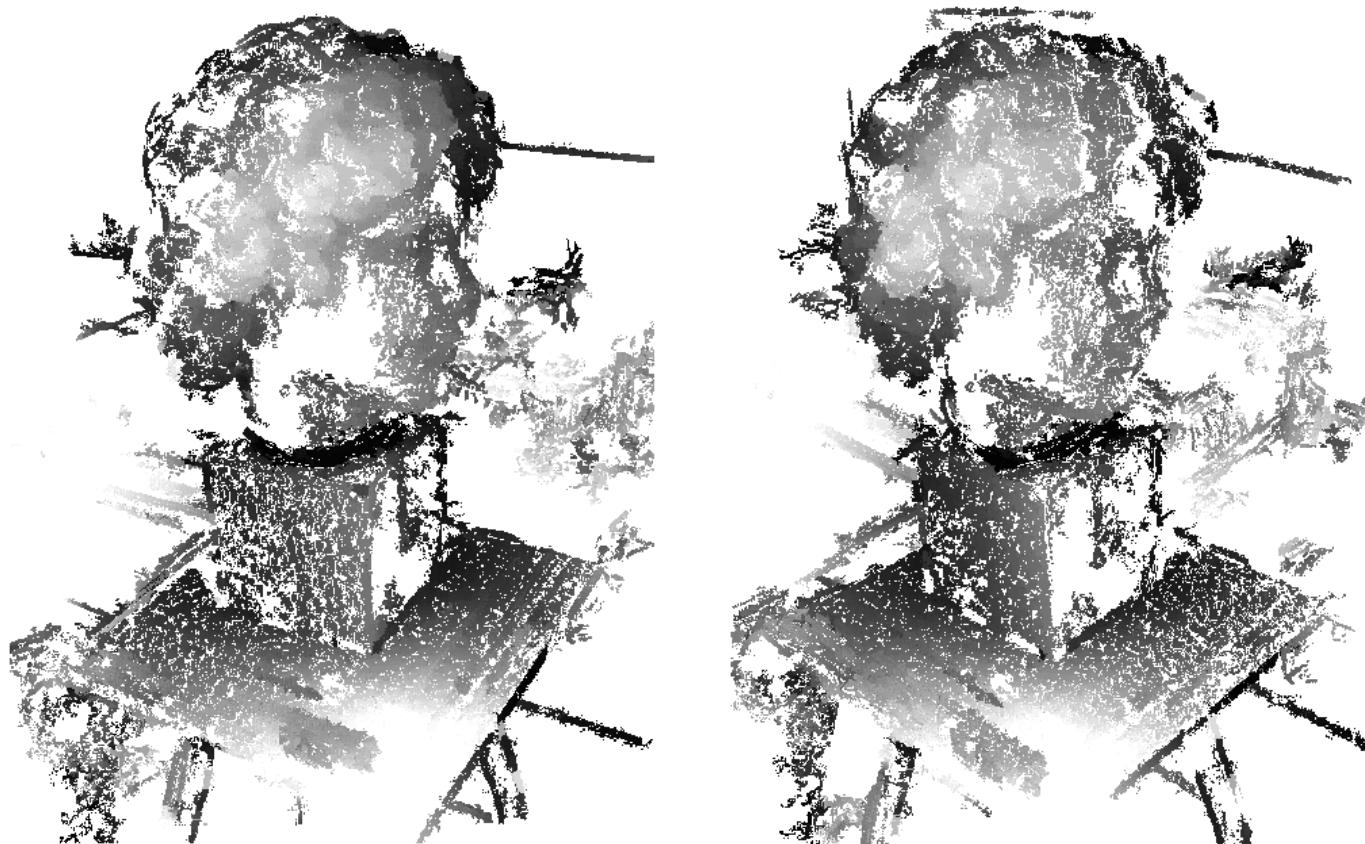
- [1] Learning Two-view Stereo Matching, J Xiao, J Chen, DY Yeung, and L Quan, 2008.
- [2] Accurate, Dense, and Robust Multi-View Stereopsis, Y Furukawa and J Ponce, 2007.
- [3] Multi-view stereo revisited. M Goesele, B Curless, SM Seitz. 2006.
- [4] Robust Dense Matching Using Local and Global Geometric Constraints, M Lhuillier & L Quan, 2000.

In another context:

- [5] PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing, C Barnes, E Shechtman, A Finkelstein, and DB Goldman, 2009.

Simplest Matching Propagation

We are going to learn a very simple algorithm that is implemented in the SFMedu program.



Descriptor: ZNCC (Zero-mean Normalized Cross-Correlation)

- Invariant to linear radiometric changes
- More conservative than others such as sum of absolute or square differences in uniform regions
- More tolerant in textured areas where noise might be important

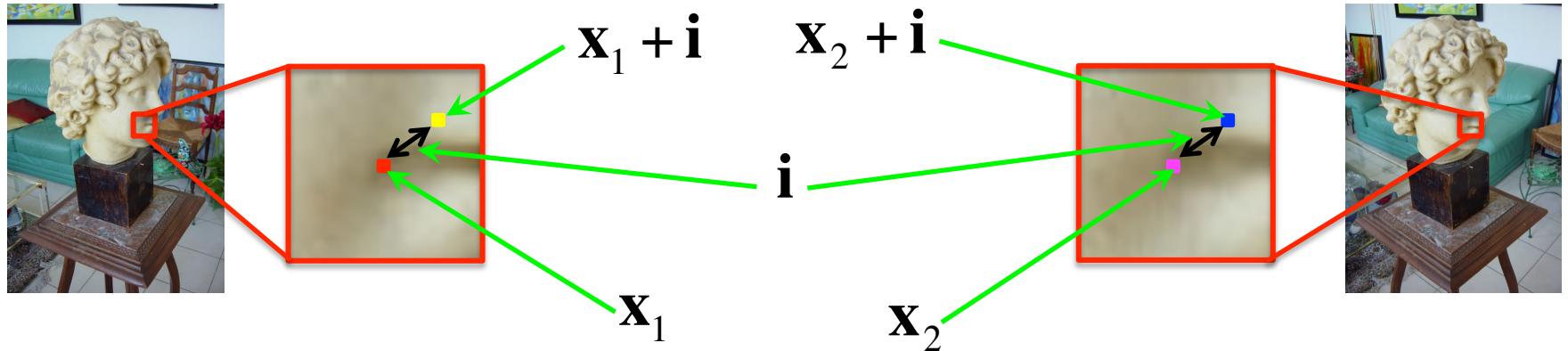
$$\text{ZNCC}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\sum_{\mathbf{i}} (I(\mathbf{x}_1 + \mathbf{i}) - \bar{I}(\mathbf{x}_1)) (I(\mathbf{x}_2 + \mathbf{i}) - \bar{I}(\mathbf{x}_2))}{\sqrt{\sum_{\mathbf{i}} (I(\mathbf{x}_1 + \mathbf{i}) - \bar{I}(\mathbf{x}_1))^2} \sum_{\mathbf{i}} (I(\mathbf{x}_2 + \mathbf{i}) - \bar{I}(\mathbf{x}_2))^2}$$



Descriptor: ZNCC (Zero-mean Normalized Cross-Correlation)

- Invariant to linear radiometric changes
- More conservative than others such as sum of absolute or square differences in uniform regions
- More tolerant in textured areas where noise might be important

$$\text{ZNCC}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\sum_{\mathbf{i}} (I(\mathbf{x}_1 + \mathbf{i}) - \bar{I}(\mathbf{x}_1)) (I(\mathbf{x}_2 + \mathbf{i}) - \bar{I}(\mathbf{x}_2))}{\sqrt{\sum_{\mathbf{i}} (I(\mathbf{x}_1 + \mathbf{i}) - \bar{I}(\mathbf{x}_1))^2} \sum_{\mathbf{i}} (I(\mathbf{x}_2 + \mathbf{i}) - \bar{I}(\mathbf{x}_2))^2}$$



Seed for propagation

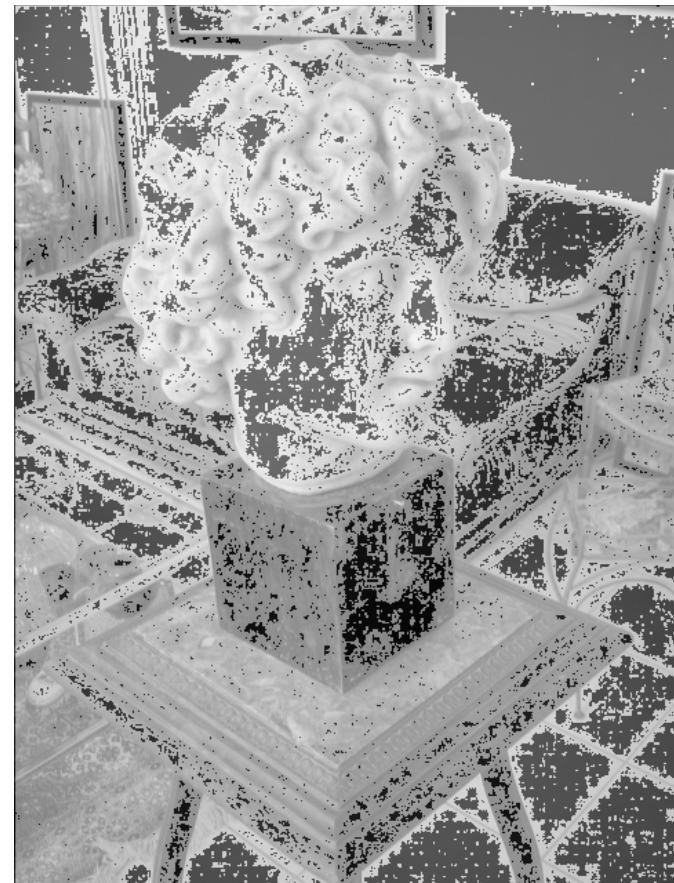


Matching Propagation (propagate.m)

- Maintain a priority queue Q
- Initialize: Put all seeds into Q with their ZNCC values as scores
- For each iteration:
 - Pop the match with best ZNCC score from Q
 - Add new potential matches in their immediate spatial neighborhood into Q
- Safety: handle uniqueness, and propagate only on matchable area

Matchable Area

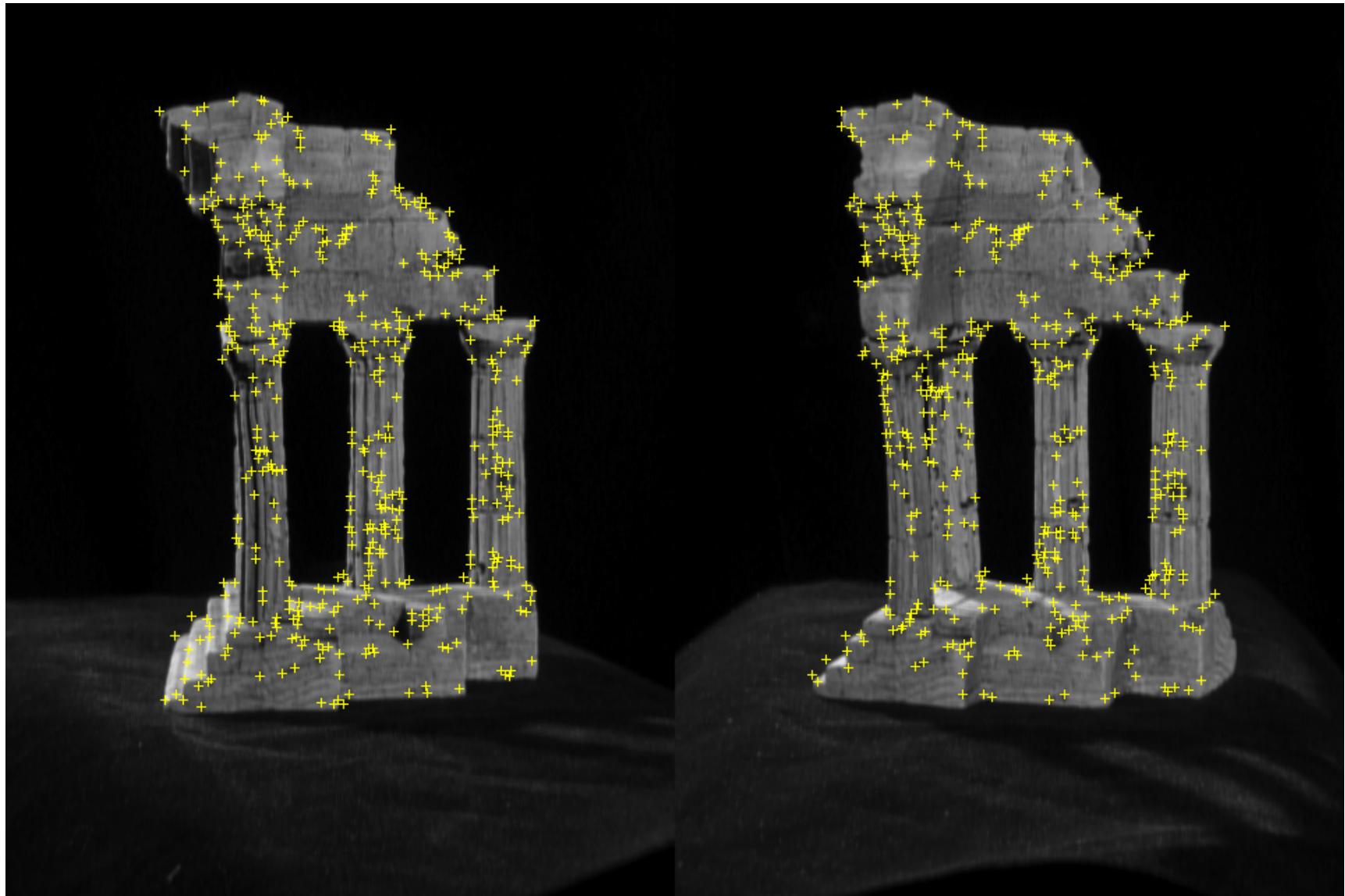
the area with maximal gradience > threshold



Result (denseMath/run.m)



Another Example



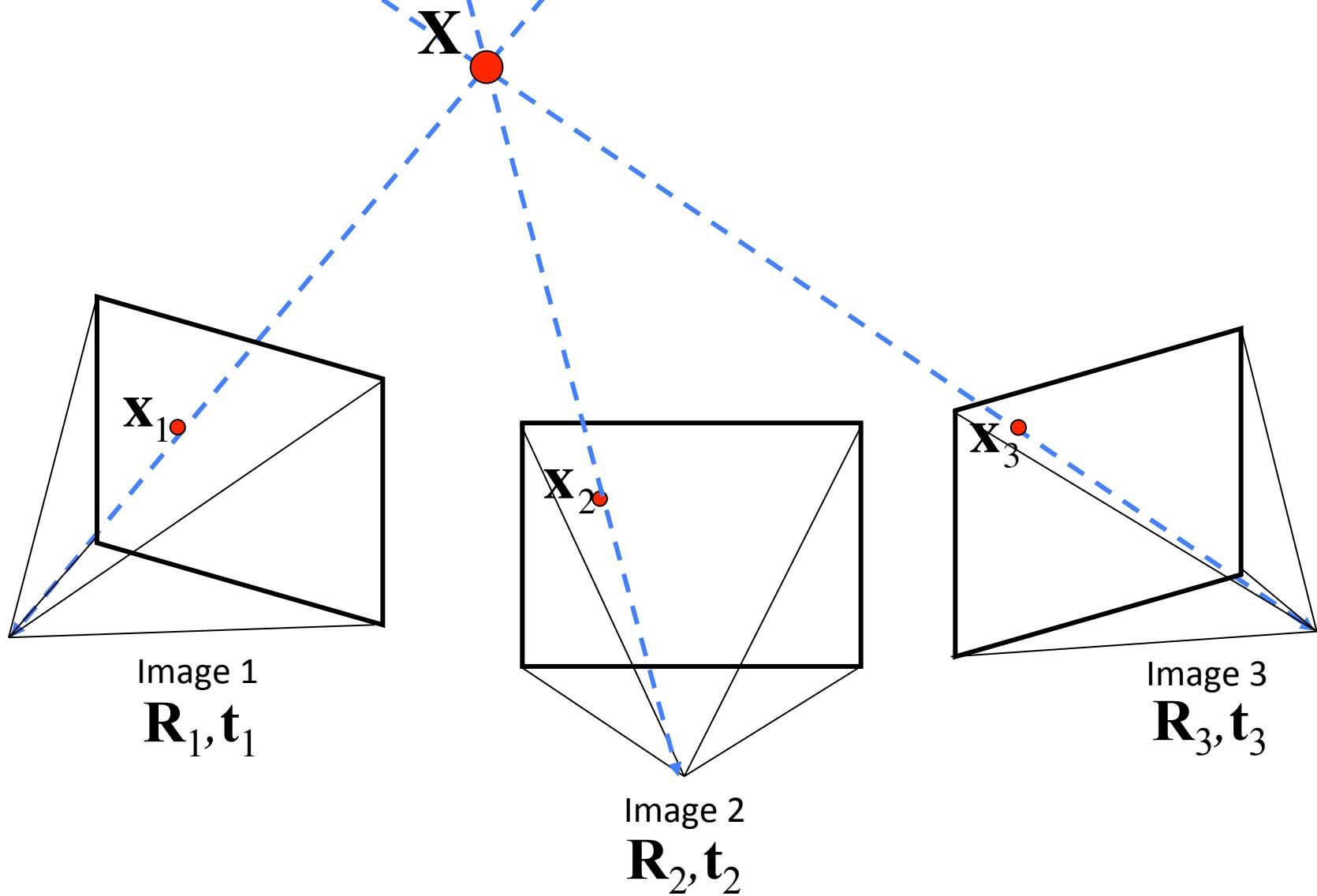
Another Example



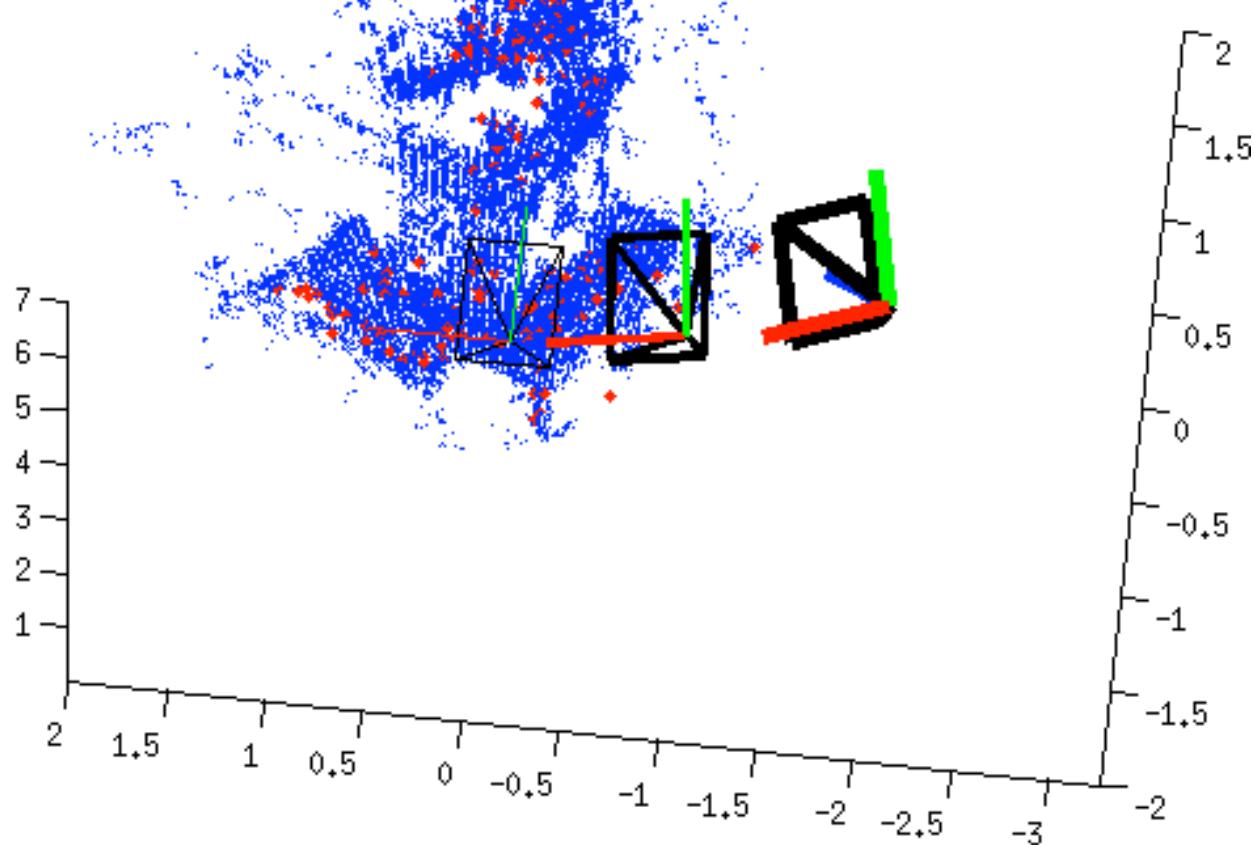
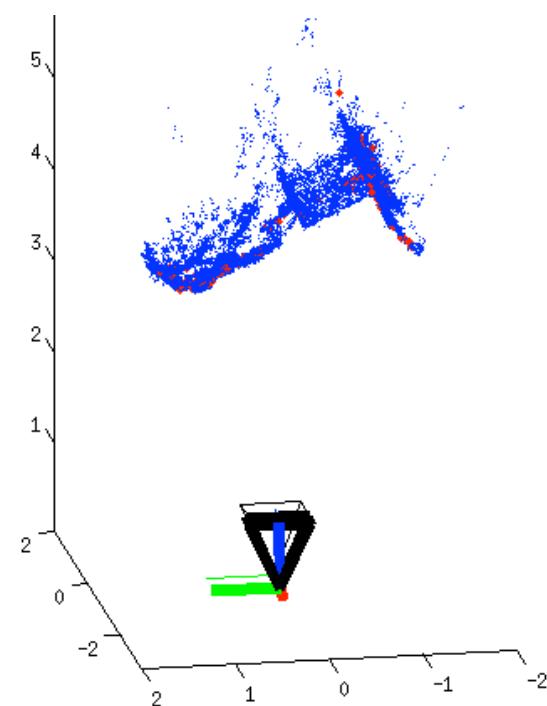
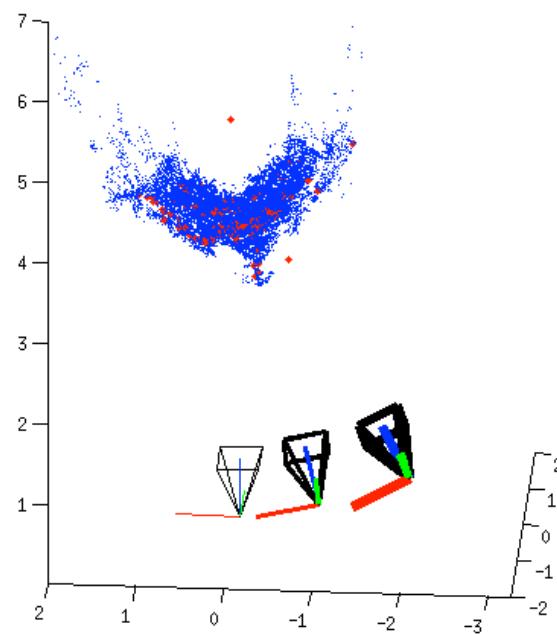
Another Example



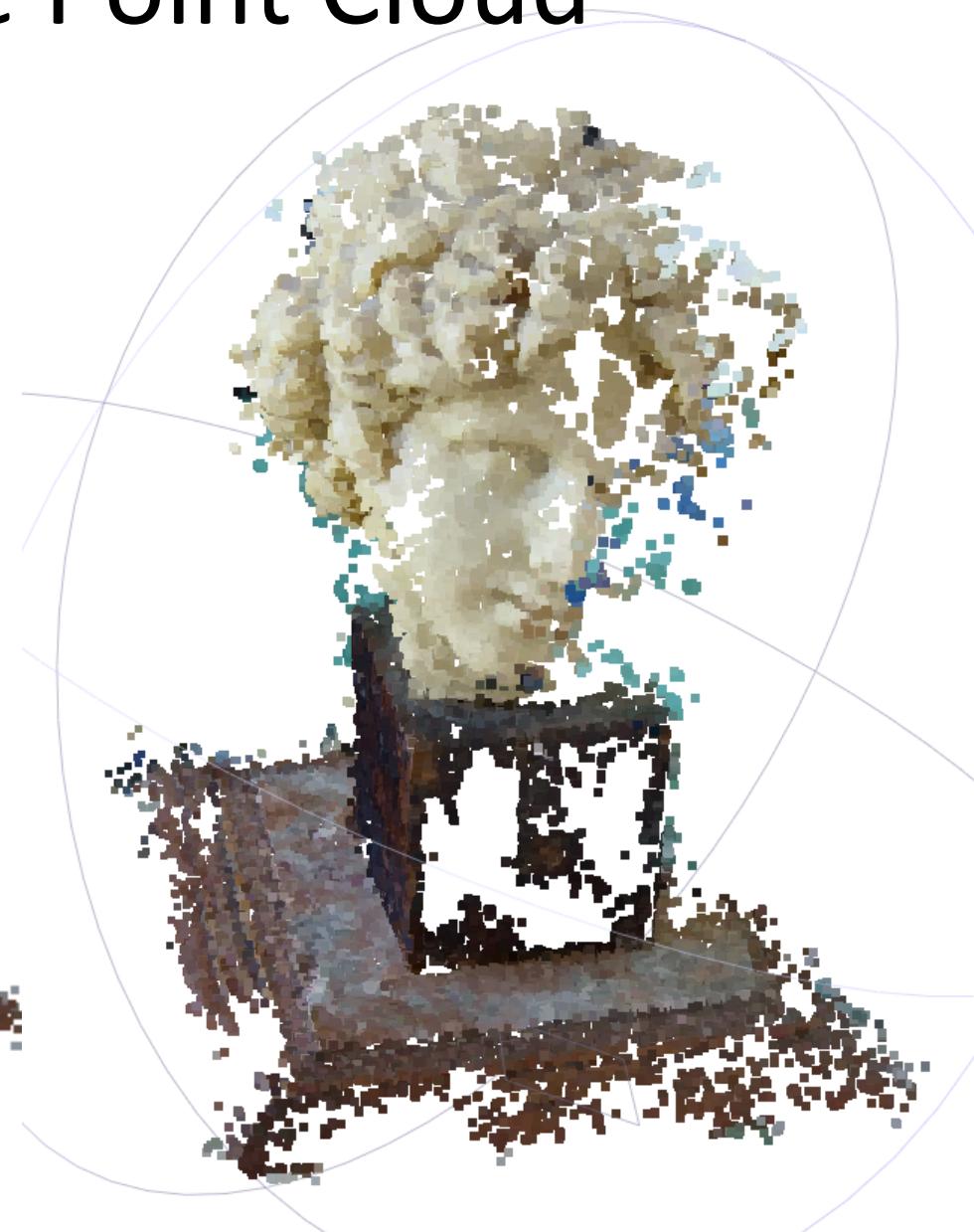
Triangulation



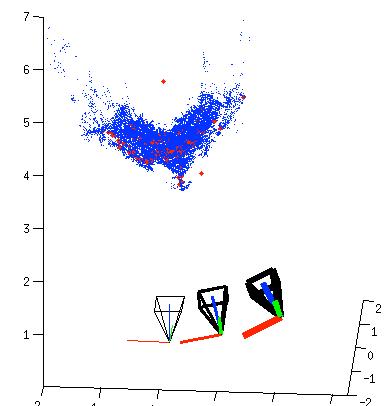
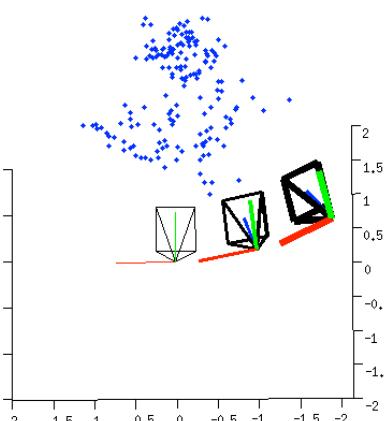
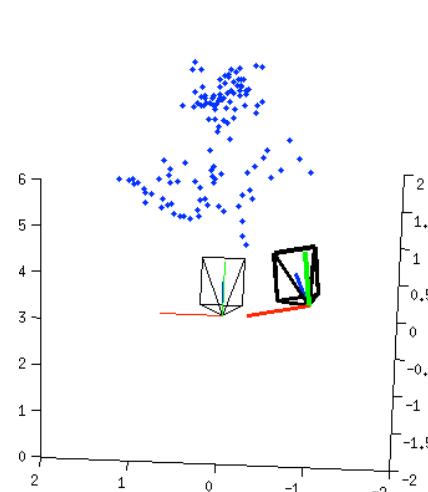
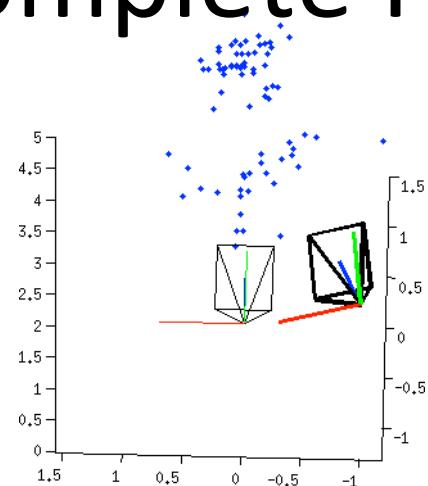
Final Result



Colorize the Point Cloud



Complete Pipeline in SFMedu



Structure from Motion (SfM)

Multi-view Stereo (MVS)

Wait: How to get the focal length?

- Auto-calibration

Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters, M Pollefeys, R Koch and L Van Gool, 1998.

http://mit.edu/jxiao/Public/software/autocalibrate/autocalibration_lin.m

- Grid Search to look for the solution with minimal reprojection error

for $f = \min_f : \max_f$

do everything, then obtain reprojection error after bundle adjustment

- Optimize for this value in bundle adjustment

- Camera Calibration (with checkerboard)

http://www.vision.caltech.edu/bouguetj/calib_doc/

- EXIF of JPEG file recorded from digital camera

Read the code of Bundler to understand how to convert EXIF into focal length value

<http://phototour.cs.washington.edu/bundler/>

Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



+



+



=



Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



+



+



+



Real World Applications

- Streetview Reconstruction and Recognition
<http://people.csail.mit.edu/jxiao/StreetSeg/>
<http://people.csail.mit.edu/jxiao/CityRec/>
- Photo Tourism <http://phototour.cs.washington.edu/>
- Microsoft Photosynth <http://photosynth.net/>
- 2d3, boujor (Matchmovers) and movies
<http://www.2d3.com/> <http://www.vicon.com/boujou/>
- Robotics: SLAM <http://openslam.org/>
Simultaneous Localization And Mapping

Steps

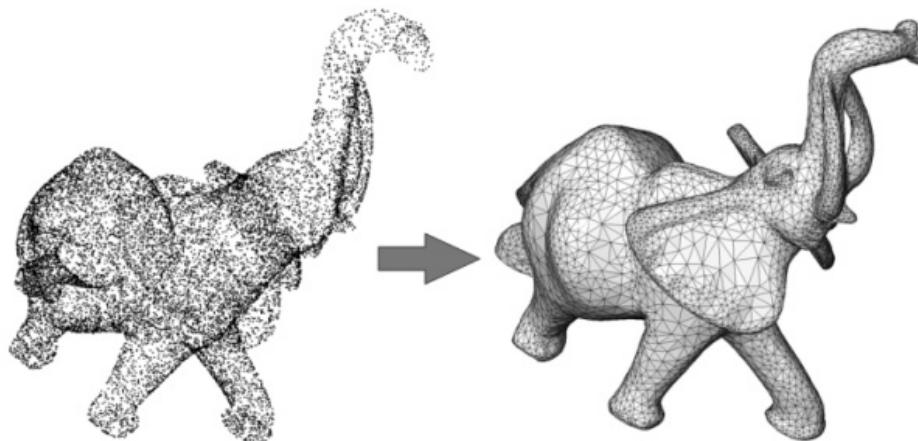
Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

+ Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling



Point Cloud → 3D Mesh Model

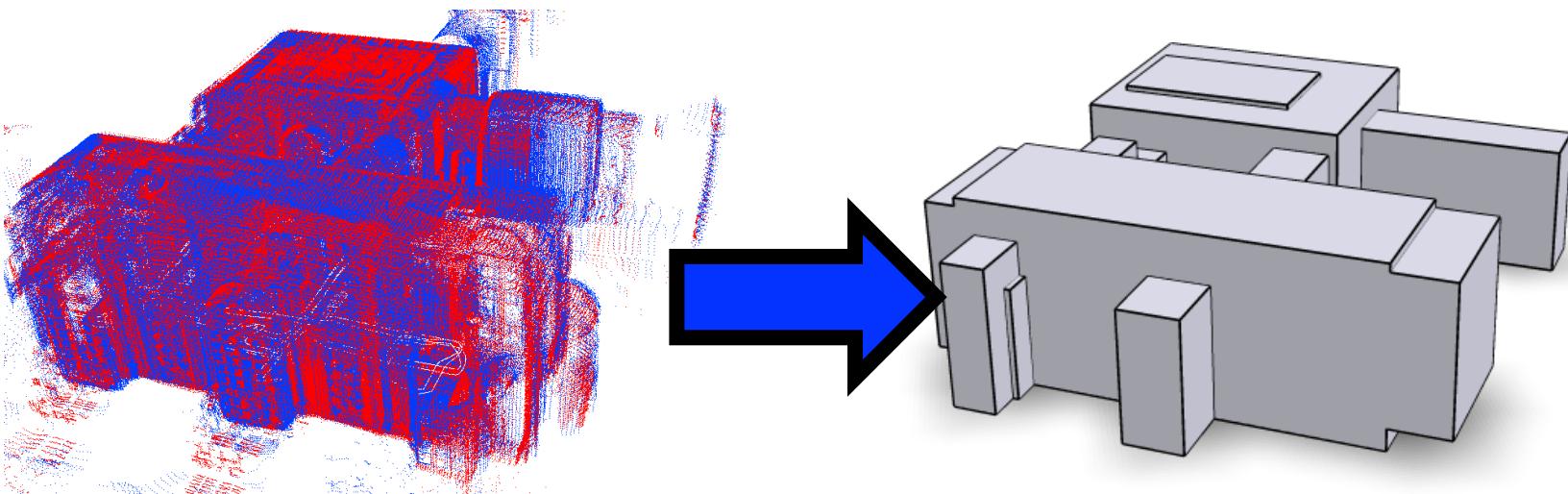
- Surface Reconstruction
 - Marching Cubes: http://en.wikipedia.org/wiki/Marching_cubes
 - Poisson Surface Reconstruction
<http://research.microsoft.com/en-us/um/people/hoppe/proj/poissonrecon/>
- Model Fitting
 - RANSAC & J-linkage <http://www.diegm.uniud.it/fusiello/demo/jlk/>
 - InverseCSG
 - GlobFit <http://code.google.com/p/globfit/>
 - Face Model Fitting <http://research.microsoft.com/apps/pubs/default.aspx?id=73211>

InverseCSG Algorithm

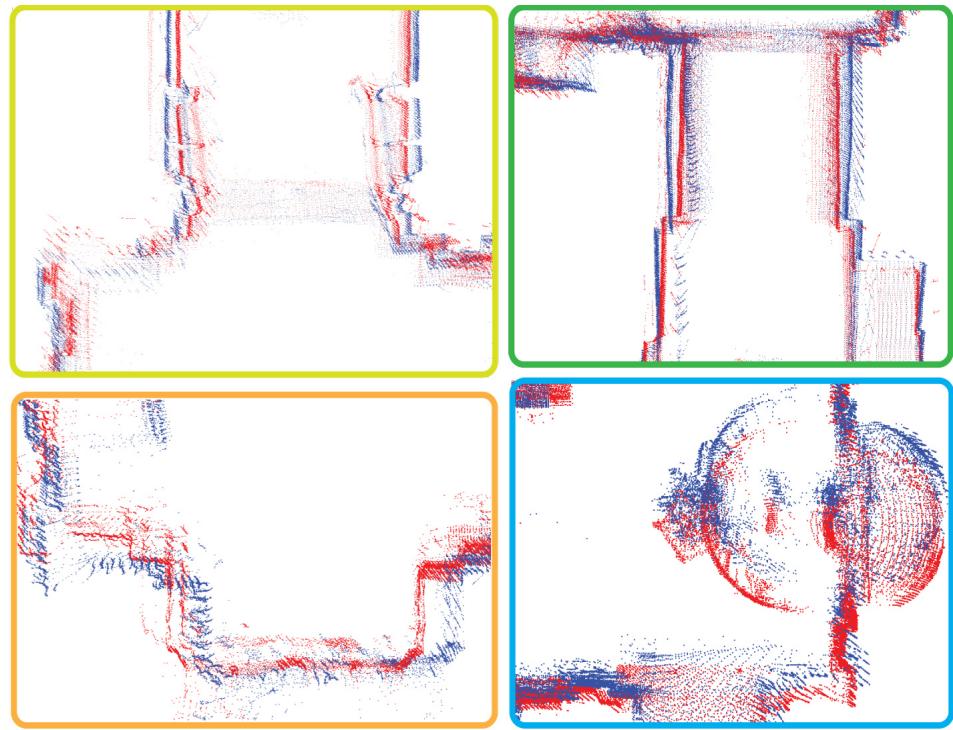
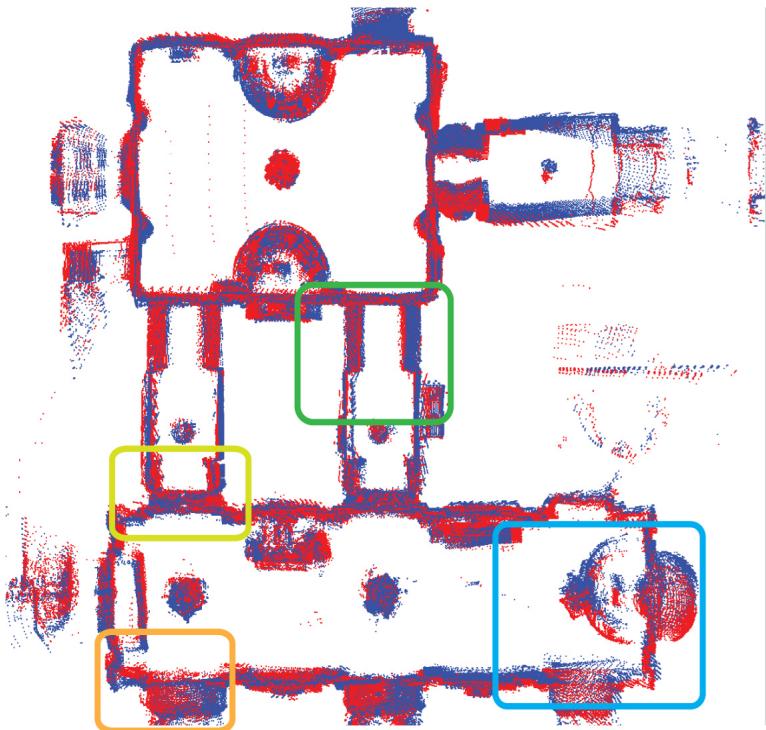


Reconstructing the World's Museums, J. Xiao and Y. Furukawa, 2012.

InverseCSG Algorithm



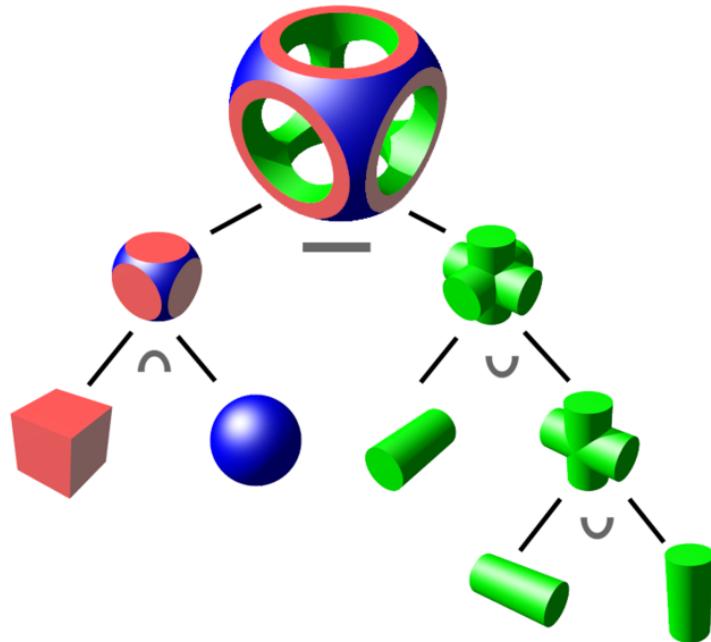
Noisy Points



top-down view of input points

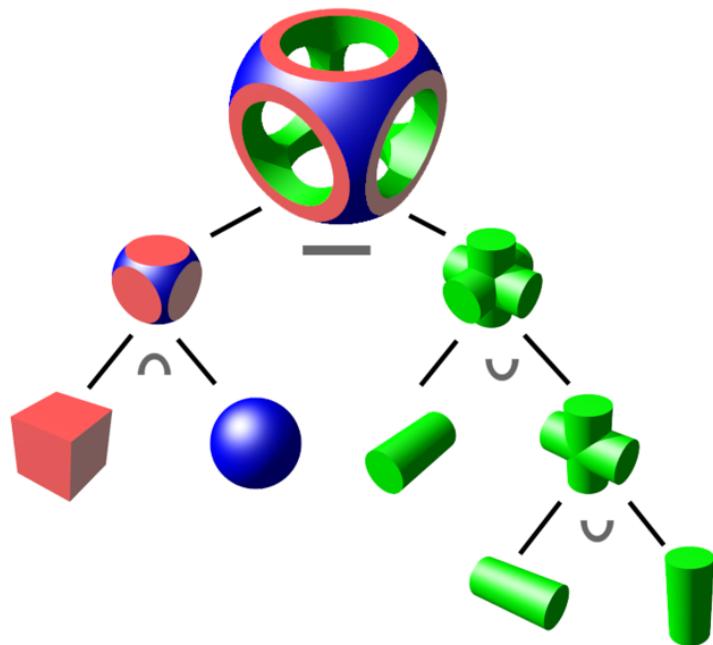
InverseCSG Algorithm

Constructive Solid Geometry (CSG)



InverseCSG Algorithm

Constructive Solid
Geometry (CSG)

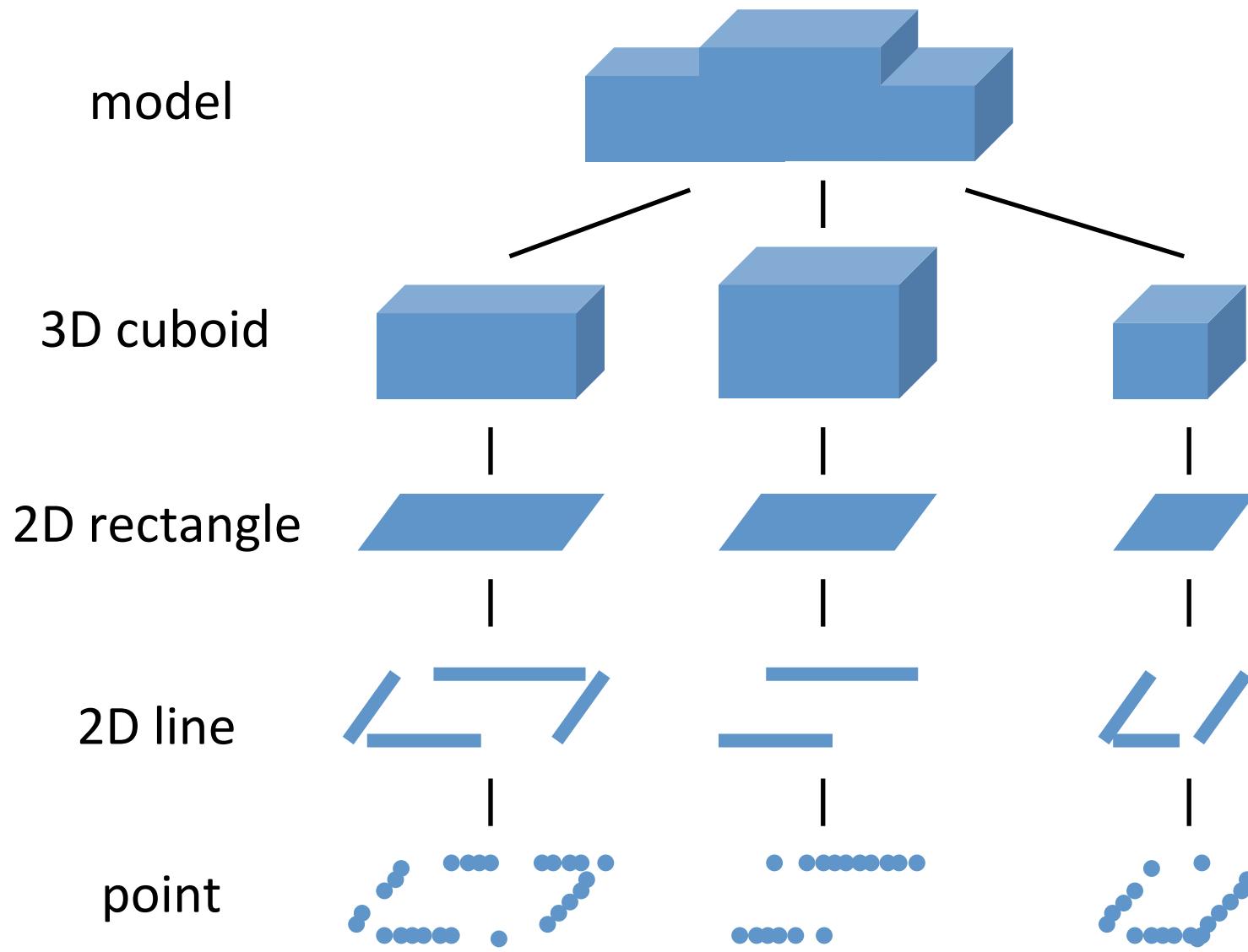


Cuboids as primitives

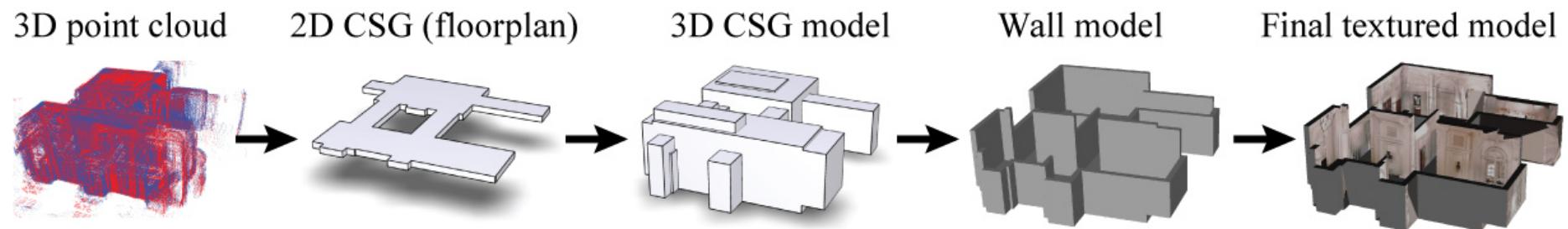


Xiao et al. NIPS 2012
Xiao et al. Siggraph Asia 2012a

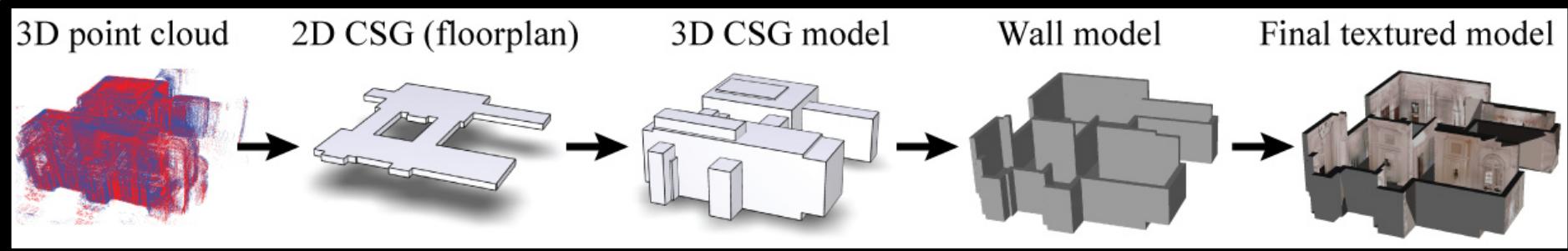
Bottom-up & Top-down



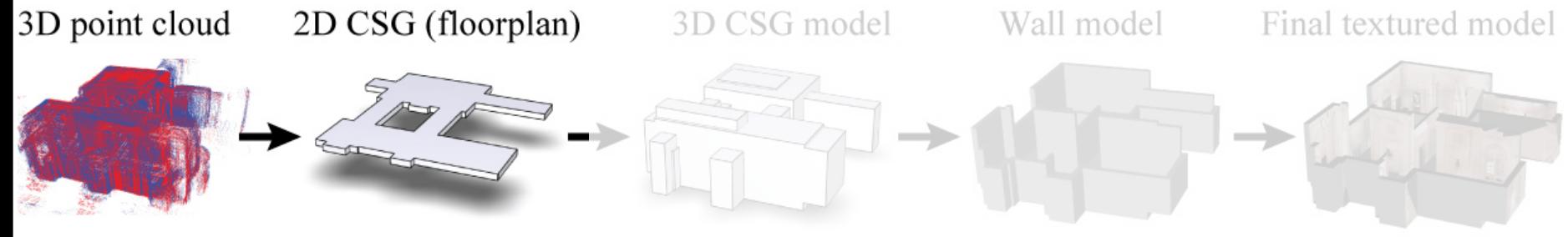
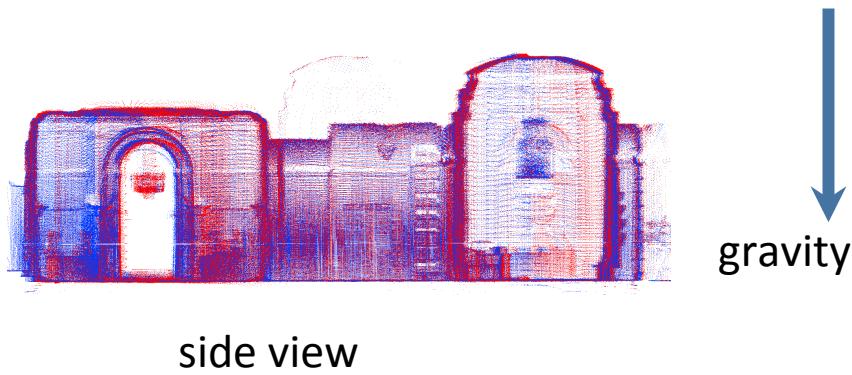
System Overview



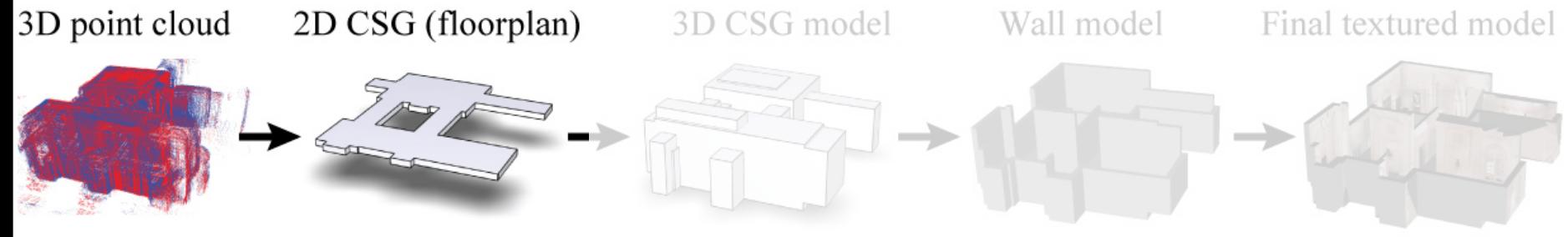
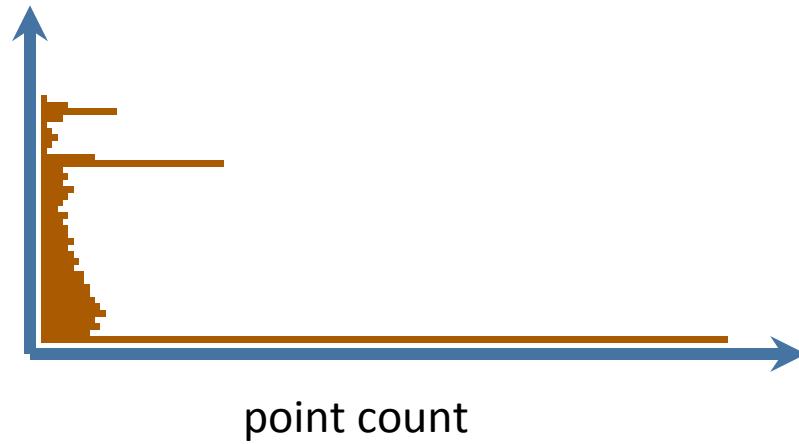
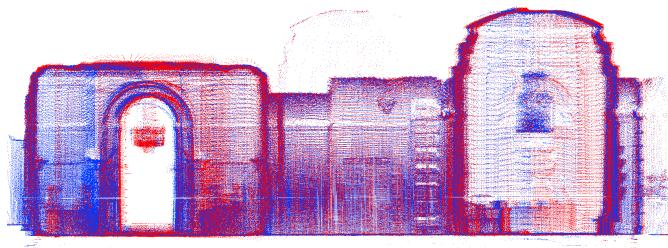
System Overview



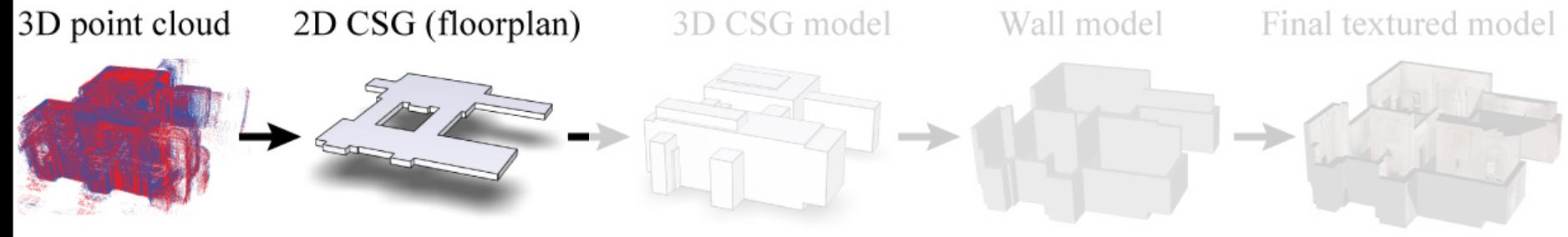
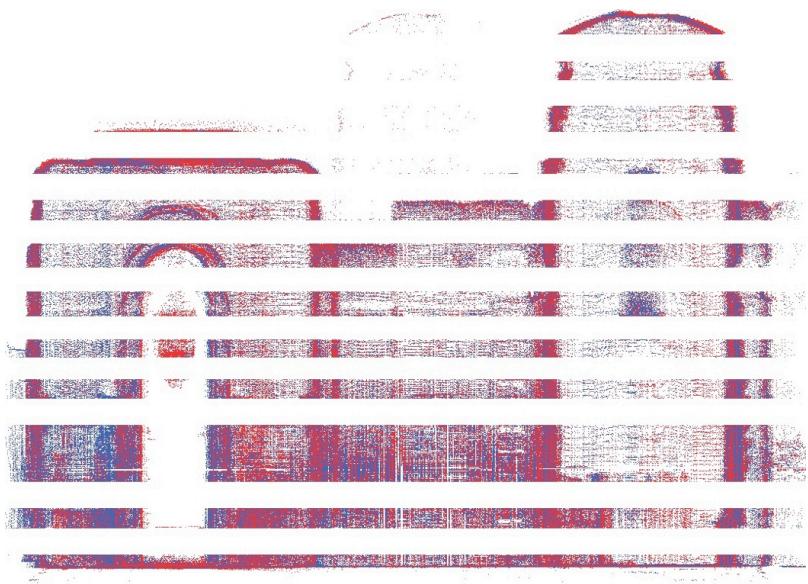
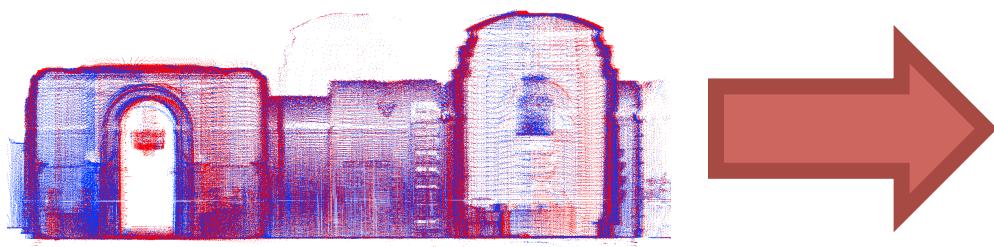
Cut into Slices



Cut into Slices

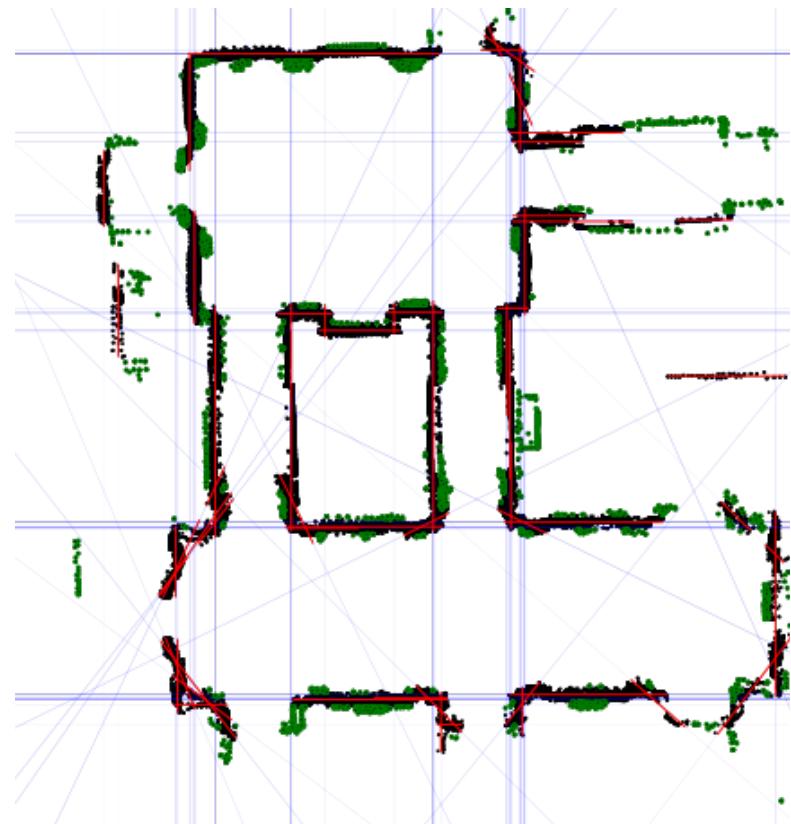


Cut into Slices

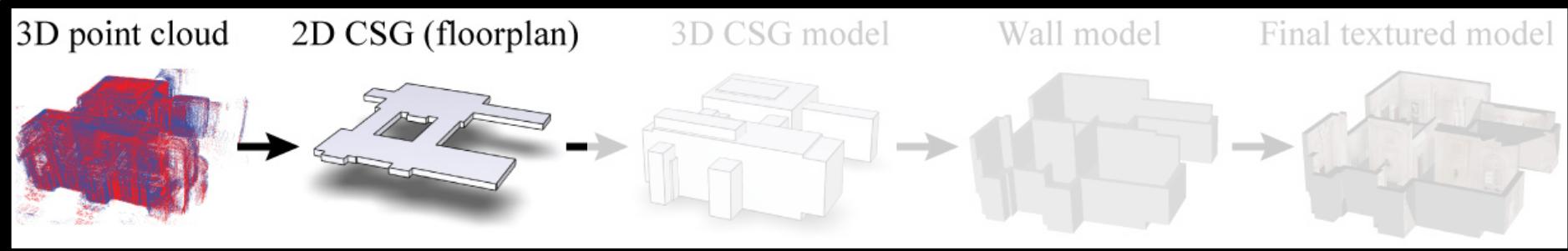


2D CSG Reconstruction

1. Generate primitives



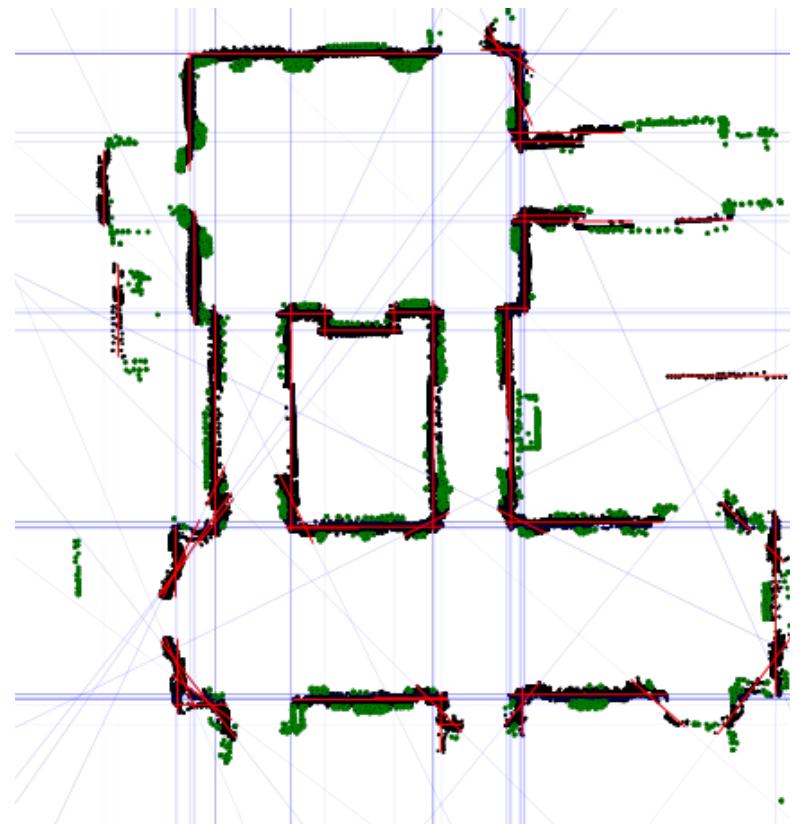
2. Choose a subset



2D CSG Reconstruction

1. Generate primitives

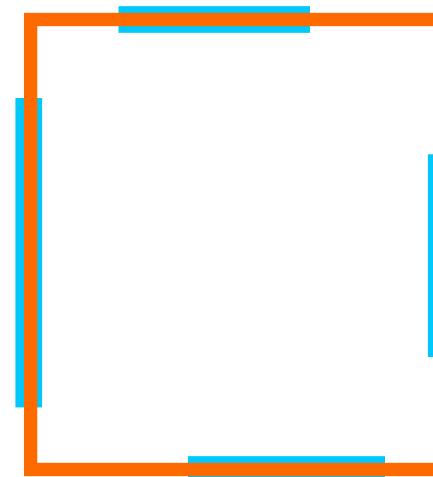
point → line



2D CSG Reconstruction

1. Generate primitives

From 4 line segments



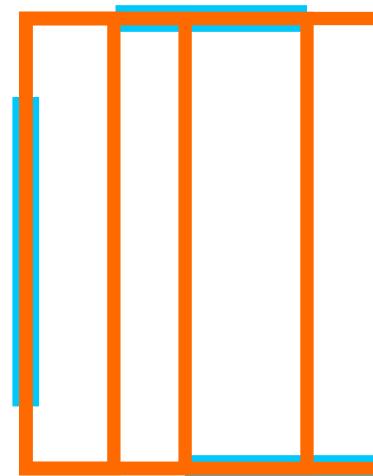
point → line

line → rectangle

2D CSG Reconstruction

1. Generate primitives

From 3 line segments

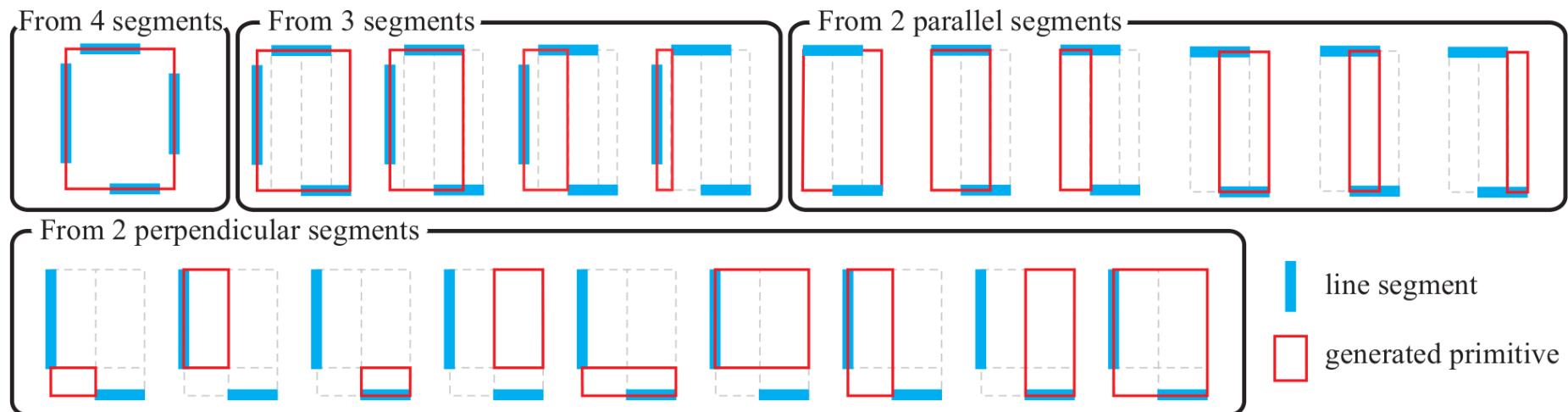


point → line

line → rectangle

2D CSG Reconstruction

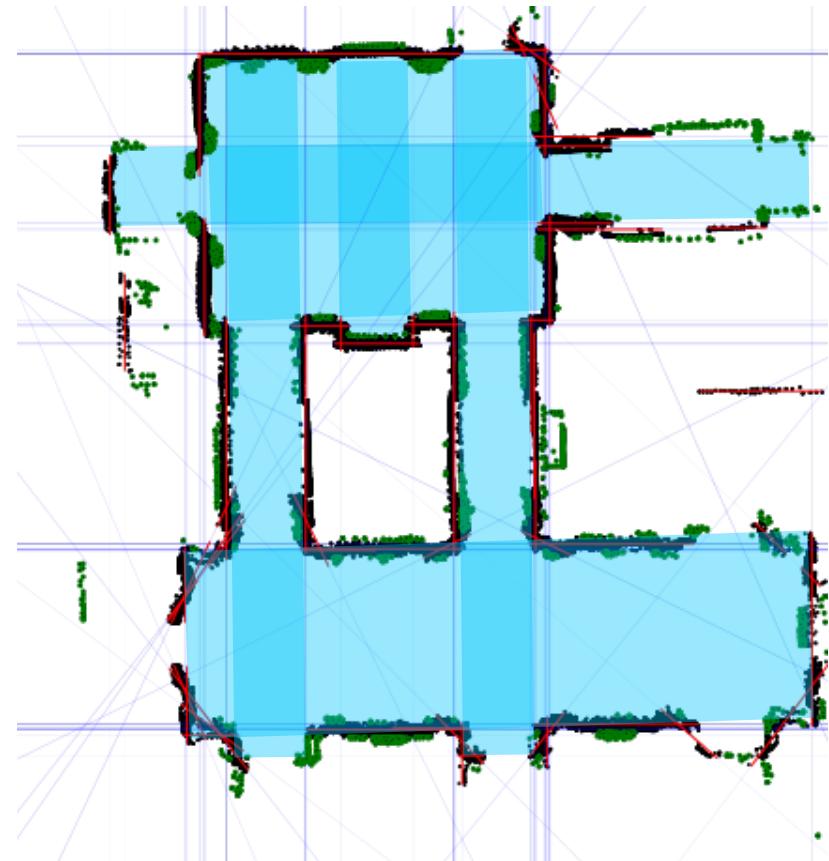
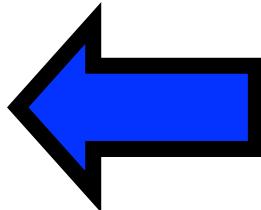
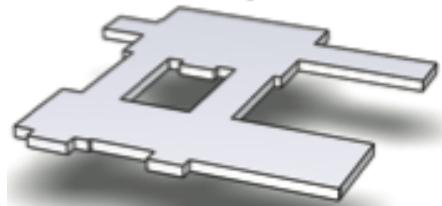
1. Generate primitives



2D CSG Reconstruction

1. Generate primitives

2. Choose a subset

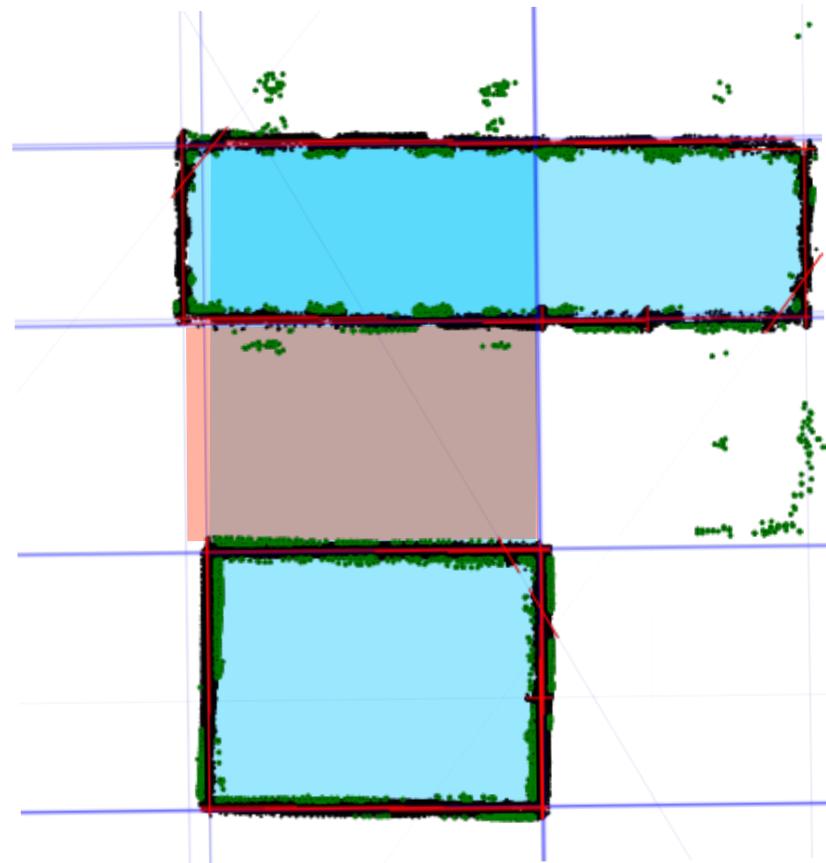


2D CSG Reconstruction

1. Generate primitives

2. Choose a subset

Repeat in each slice



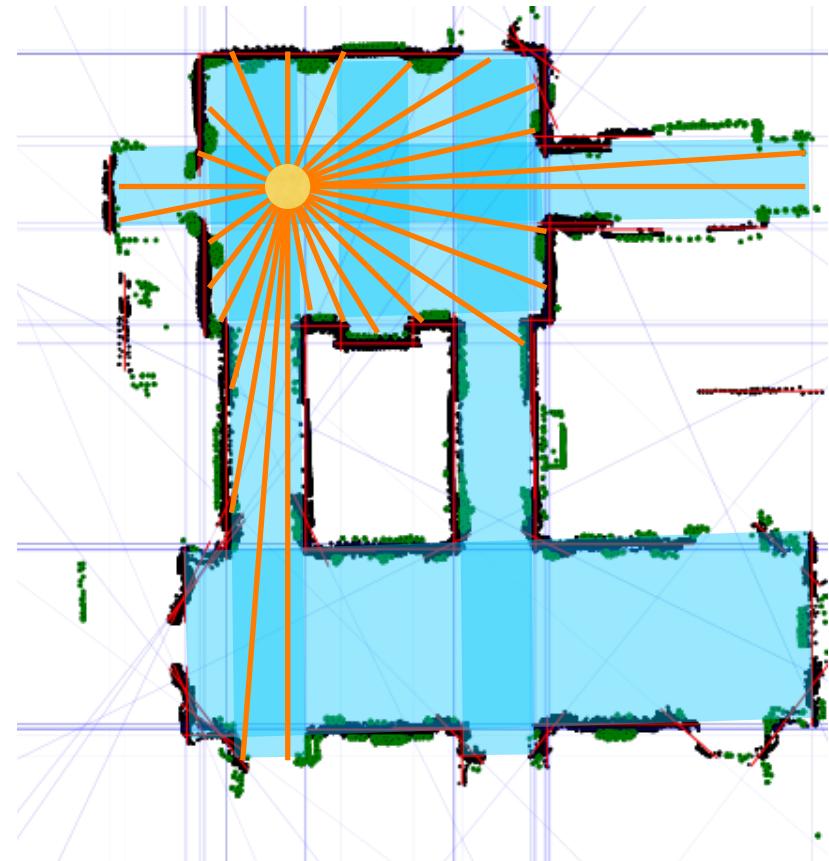
2D CSG Reconstruction

Explain the data

- Free space
- Laser points

Simple

- Regularization



Objective Function

$$E_1(T) = \frac{\{\text{Sum of free-space scores inside } T\}}{\{\text{Total sum in the domain without negative scores}\}}$$

$$E_2(T) = \frac{\{\#\text{ of points on the surface of } T\}}{\{\text{total # of points}\}}$$

$$E_3(T) = \frac{\{\text{perimeter of } T \text{ near laser points (within 0.2 meters)}\}}{\{\text{total perimeter of } T\}}$$

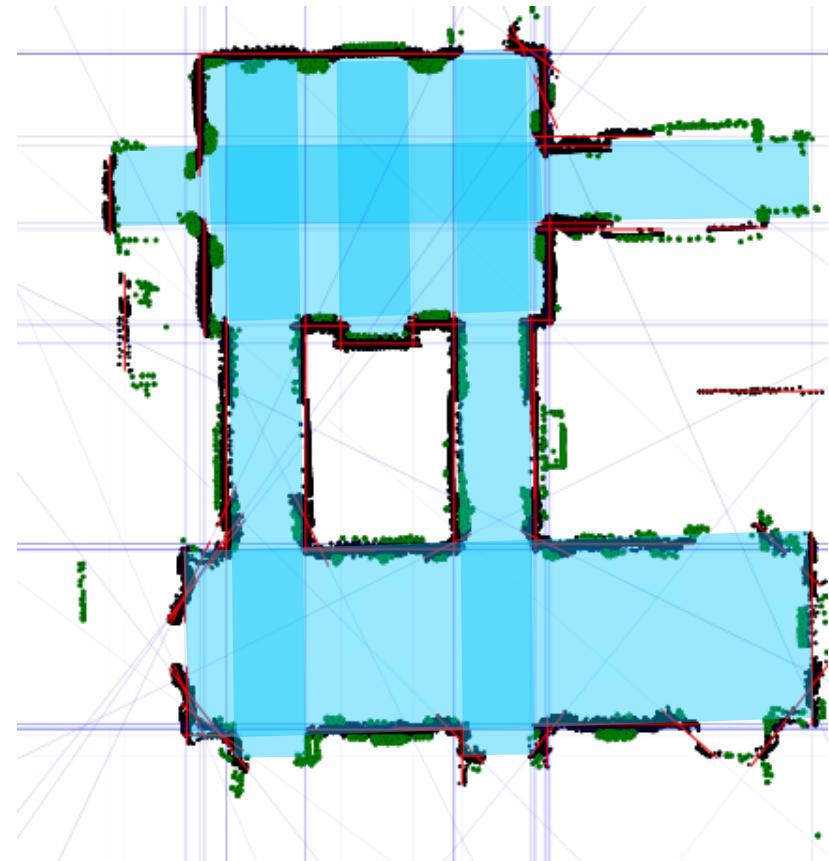
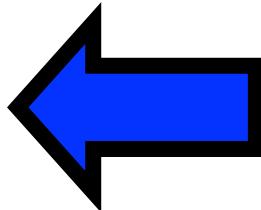
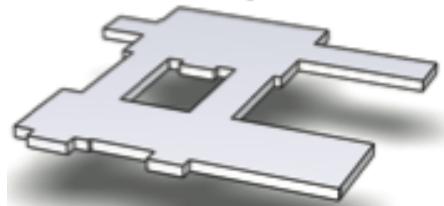
$$E(T) = w_1 E_1(T) + w_2 E_2(T) + w_3 E_3(T)$$

Free space Laser points Regularization

2D CSG Reconstruction

1. Generate primitives

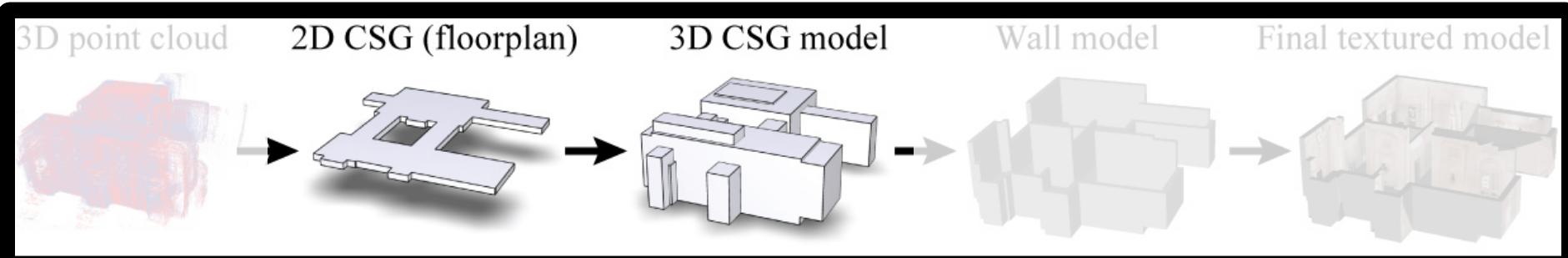
2. Choose a subset



3D CSG Reconstruction

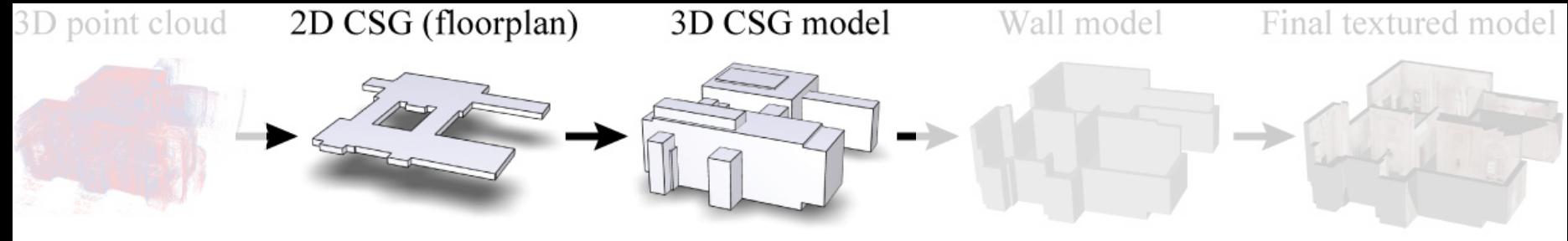
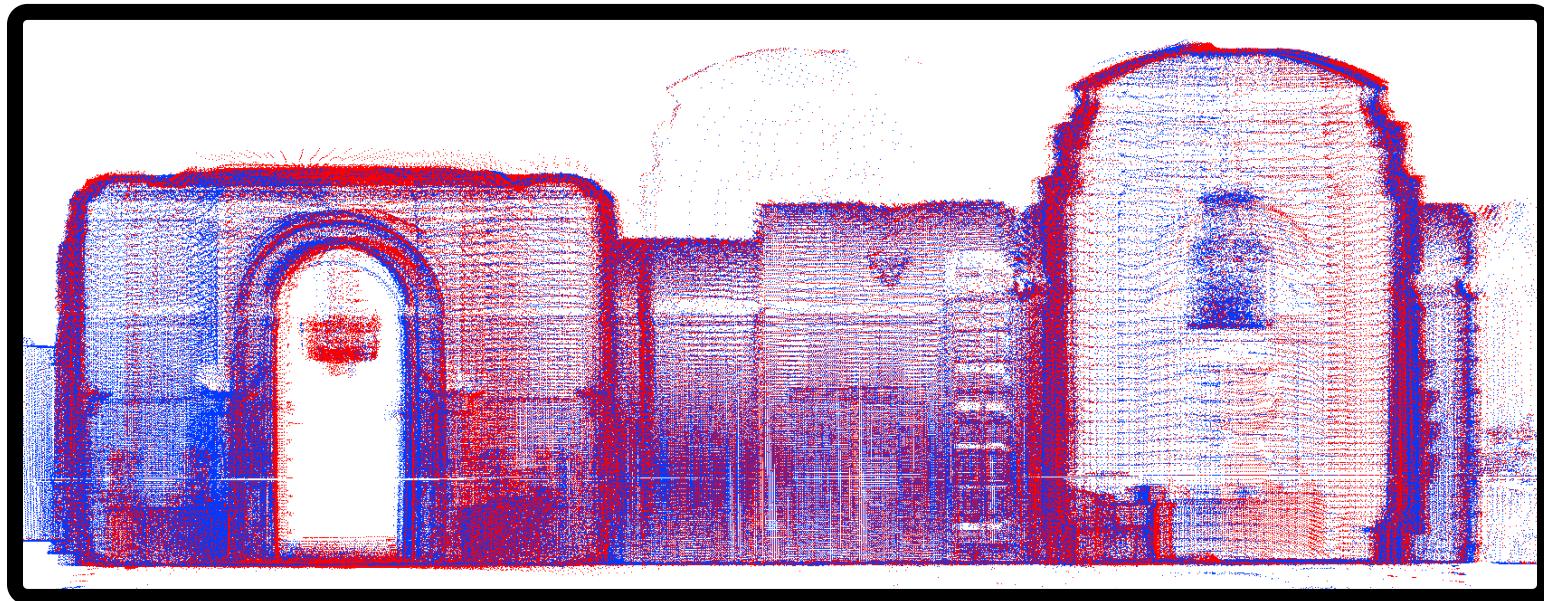
1. Generate primitives (cuboids)

2. Choose a subset (out of primitive candidates)



3D CSG Reconstruction

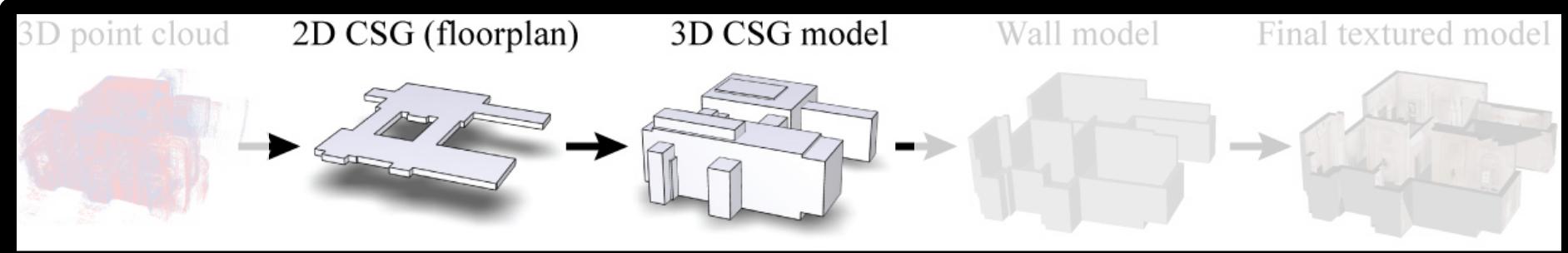
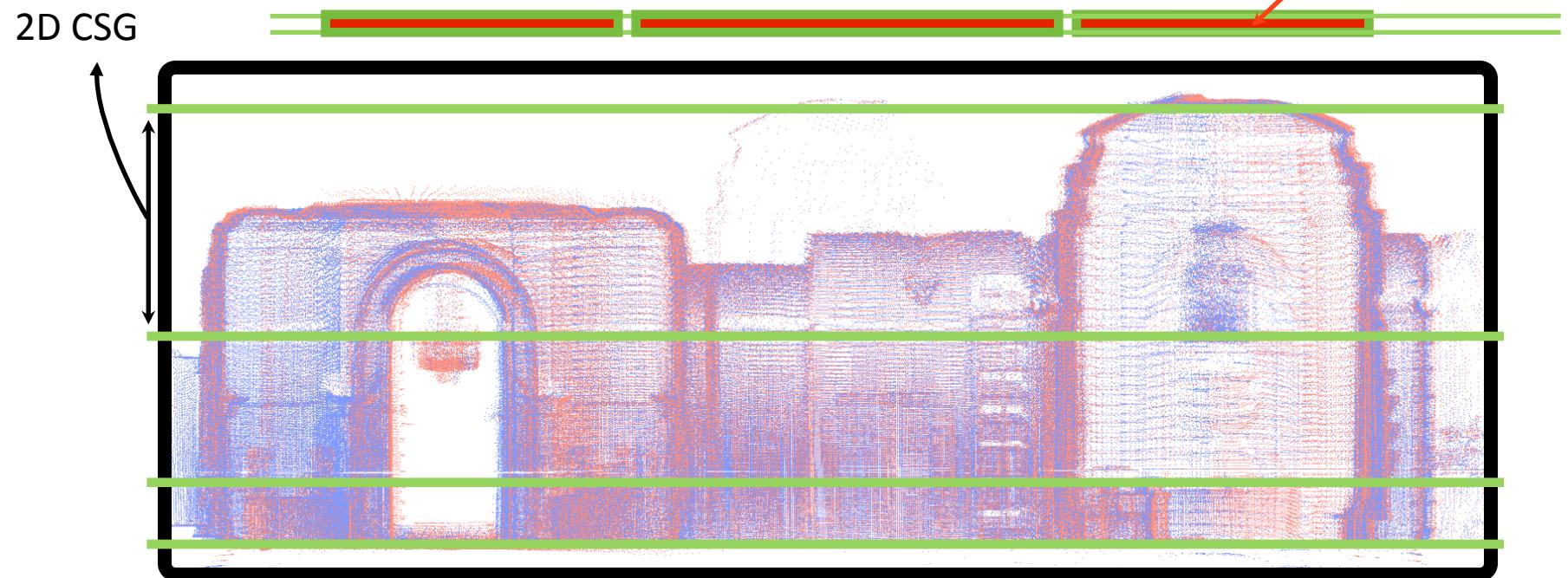
1. Generate primitives (cuboids)



3D CSG Reconstruction

1. Generate primitives (cuboids)

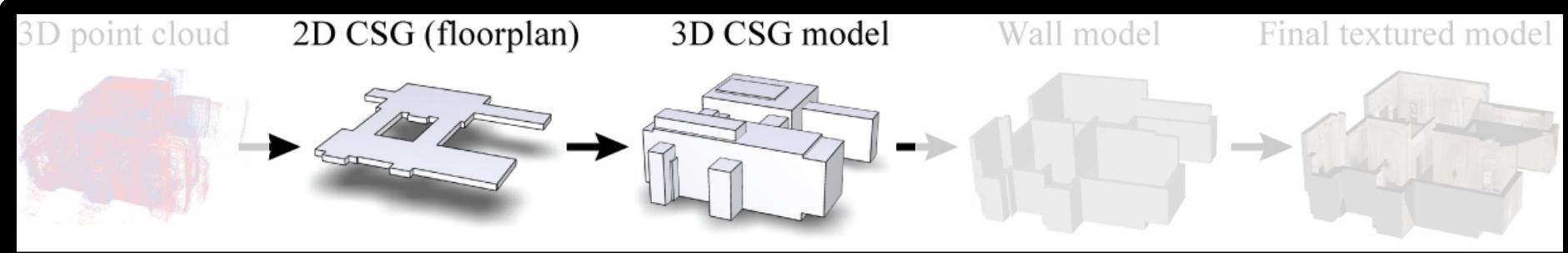
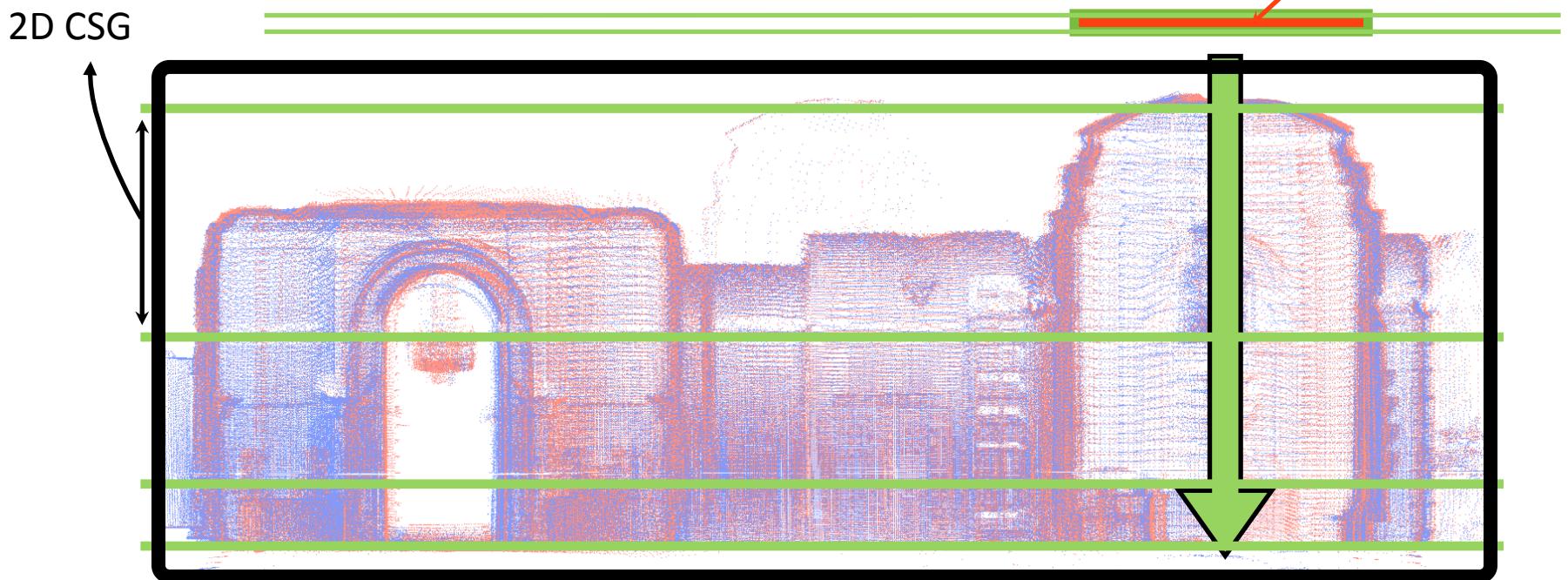
Rectangle primitive



3D CSG Reconstruction

1. Generate primitives (cuboids)

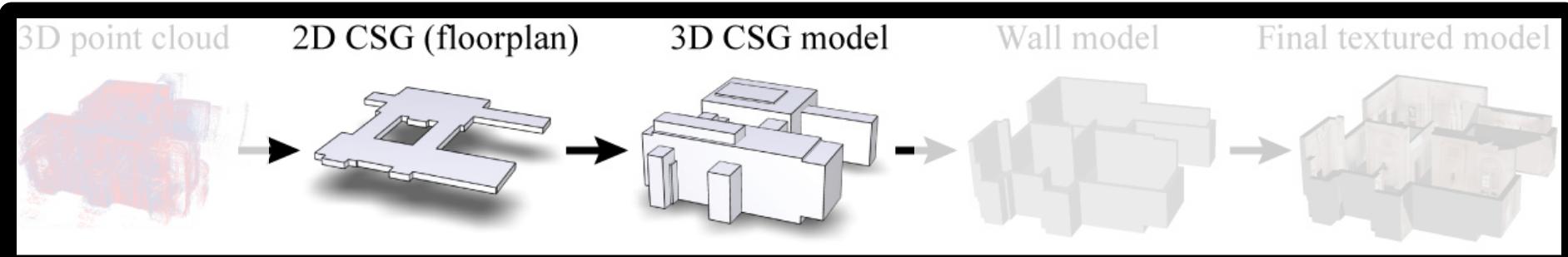
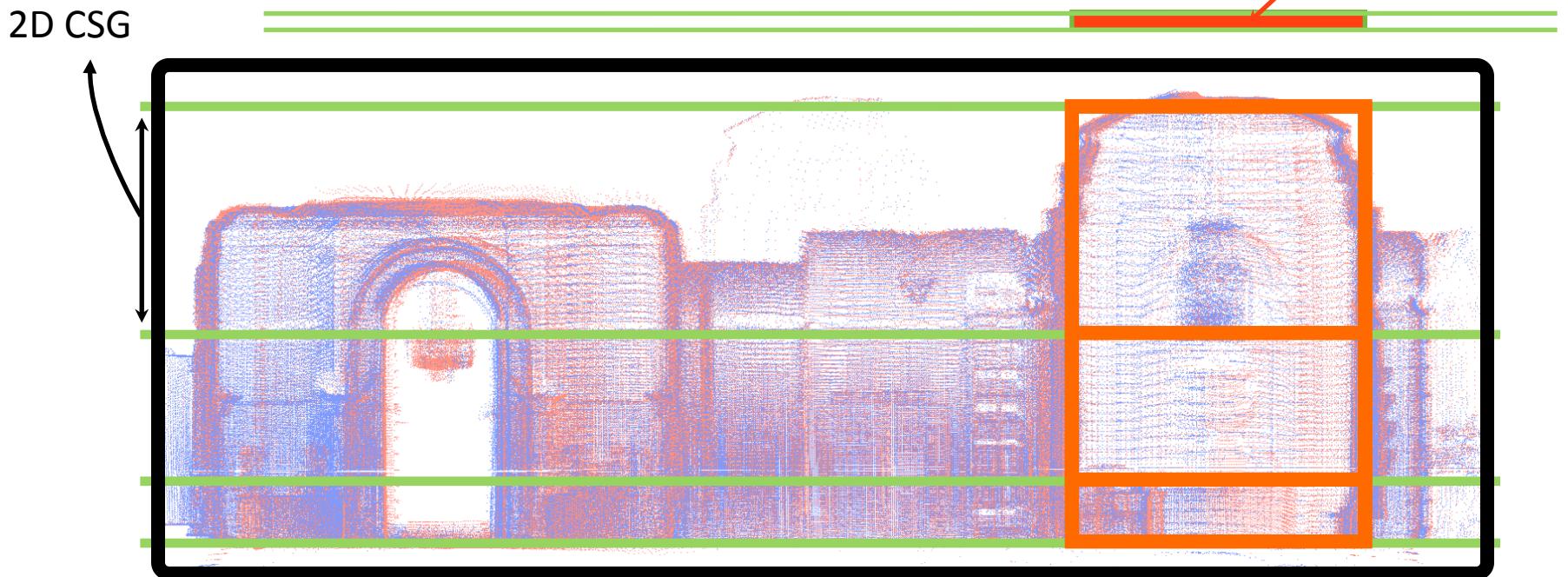
Rectangle primitive



3D CSG Reconstruction

1. Generate primitives (cuboids)

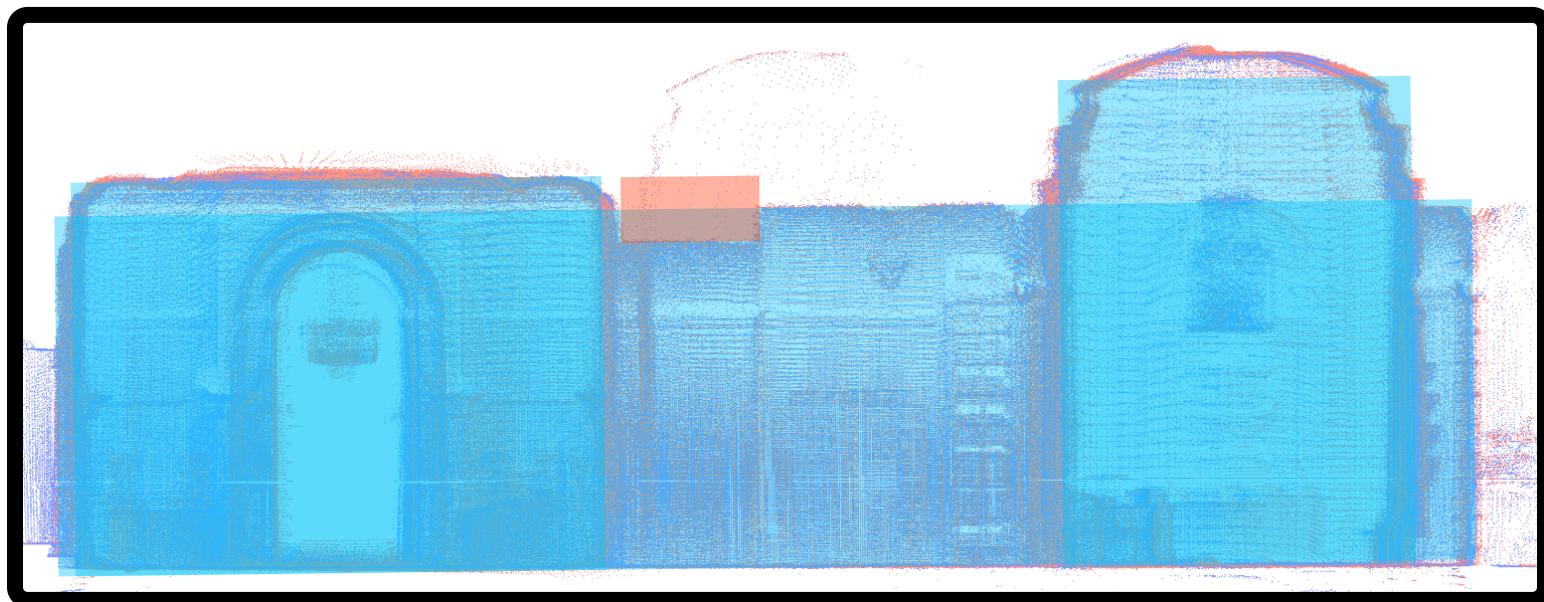
Rectangle primitive



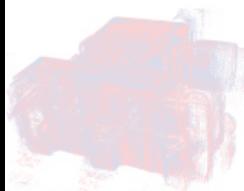
3D CSG Reconstruction

1. Generate primitives (**cuboids**)

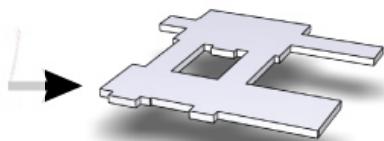
2. Choose a subset



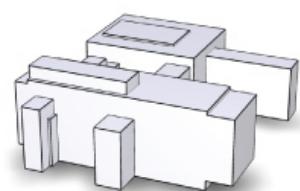
3D point cloud



2D CSG (floorplan)



3D CSG model



Wall model



Final textured model

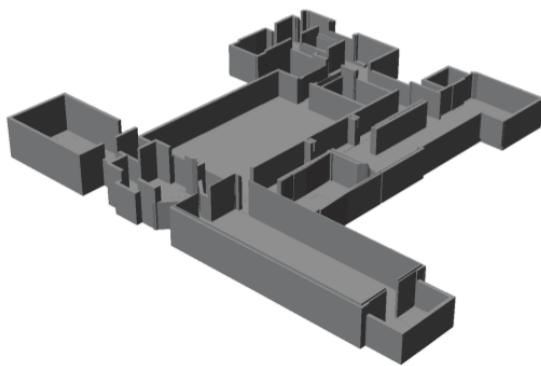


Algorithm on Run

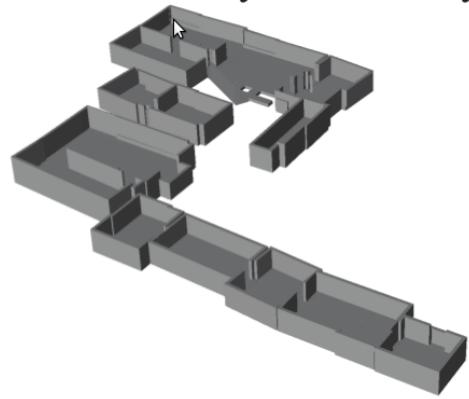
Step-by-step visualization
of
3D CSG model reconstruction

Result

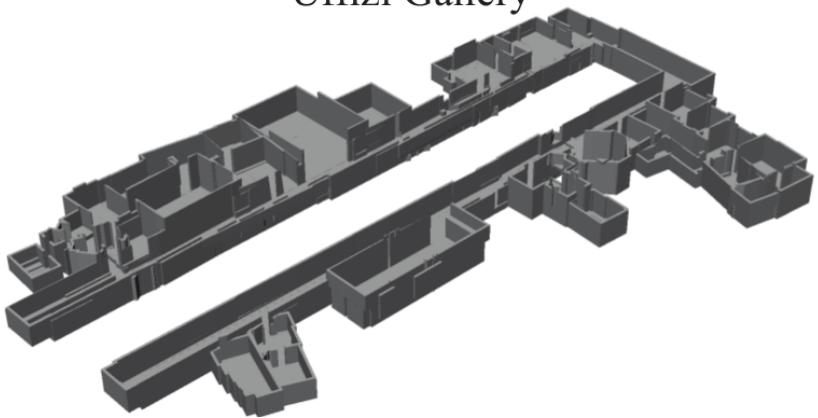
The Frick Collection



The State Tretyakov Gallery



Uffizi Gallery



Shaded mesh

Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

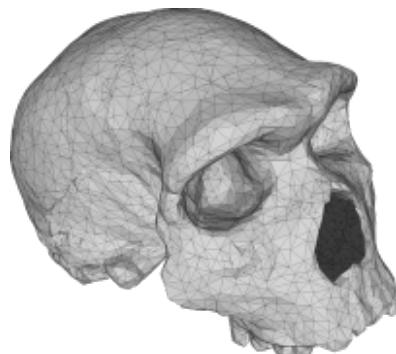
Points → Meshes: Model Fitting

Meshes → Models: Texture Mapping

+

=

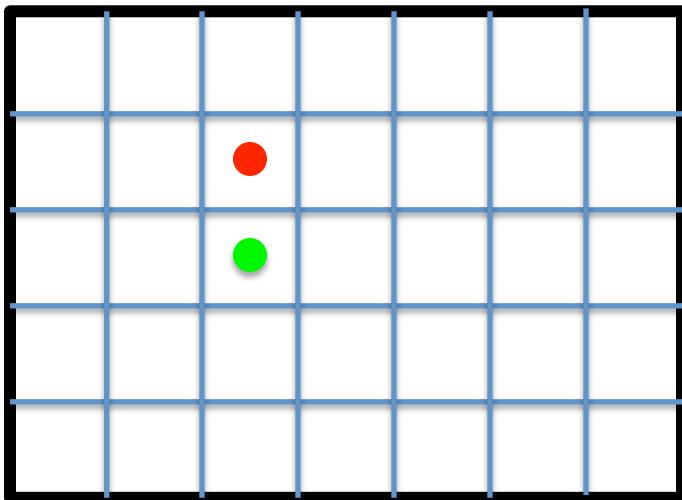
Images → Models: Image-based Modeling



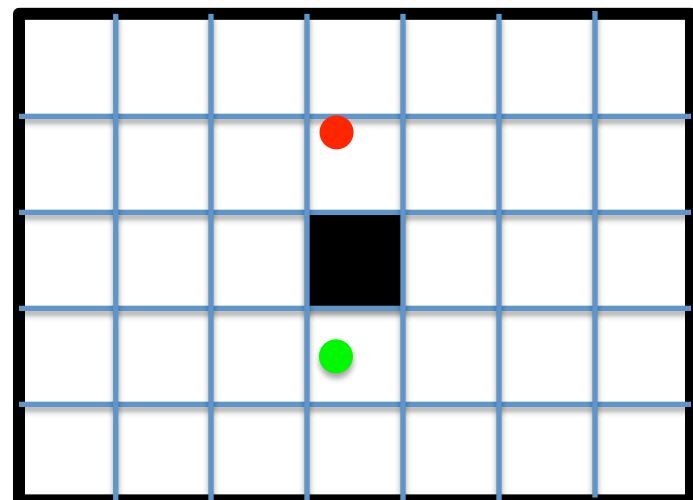
Texture Mapping

- Texture Stitching
- View-based Texture Mapping
- View Interpolation and Warping
- Interactive Visualization

Image Warping: Forward Warping



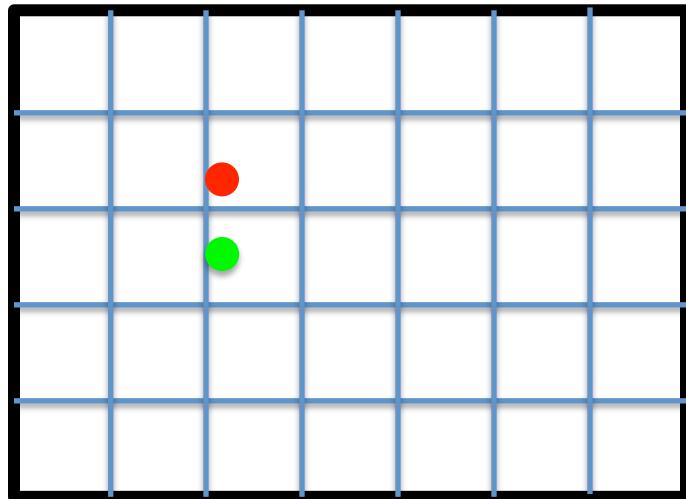
Source Image



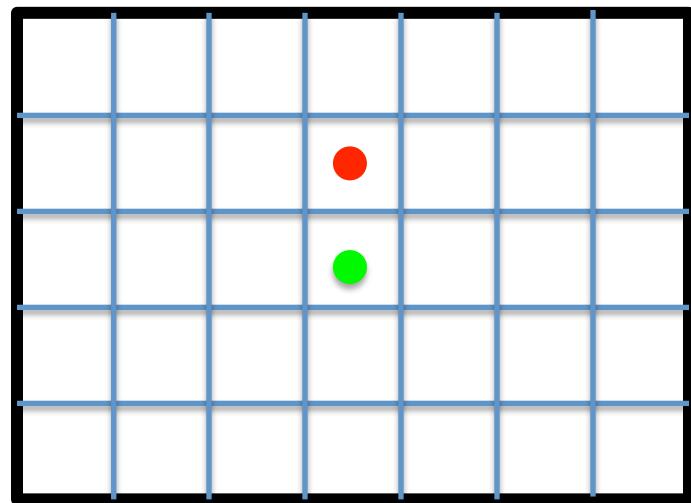
Target Image

Matlab interp2 example: <http://www.cse.psu.edu/~rcollins/CSE486/lecture14.pdf>

Image Warping: Backward Warping



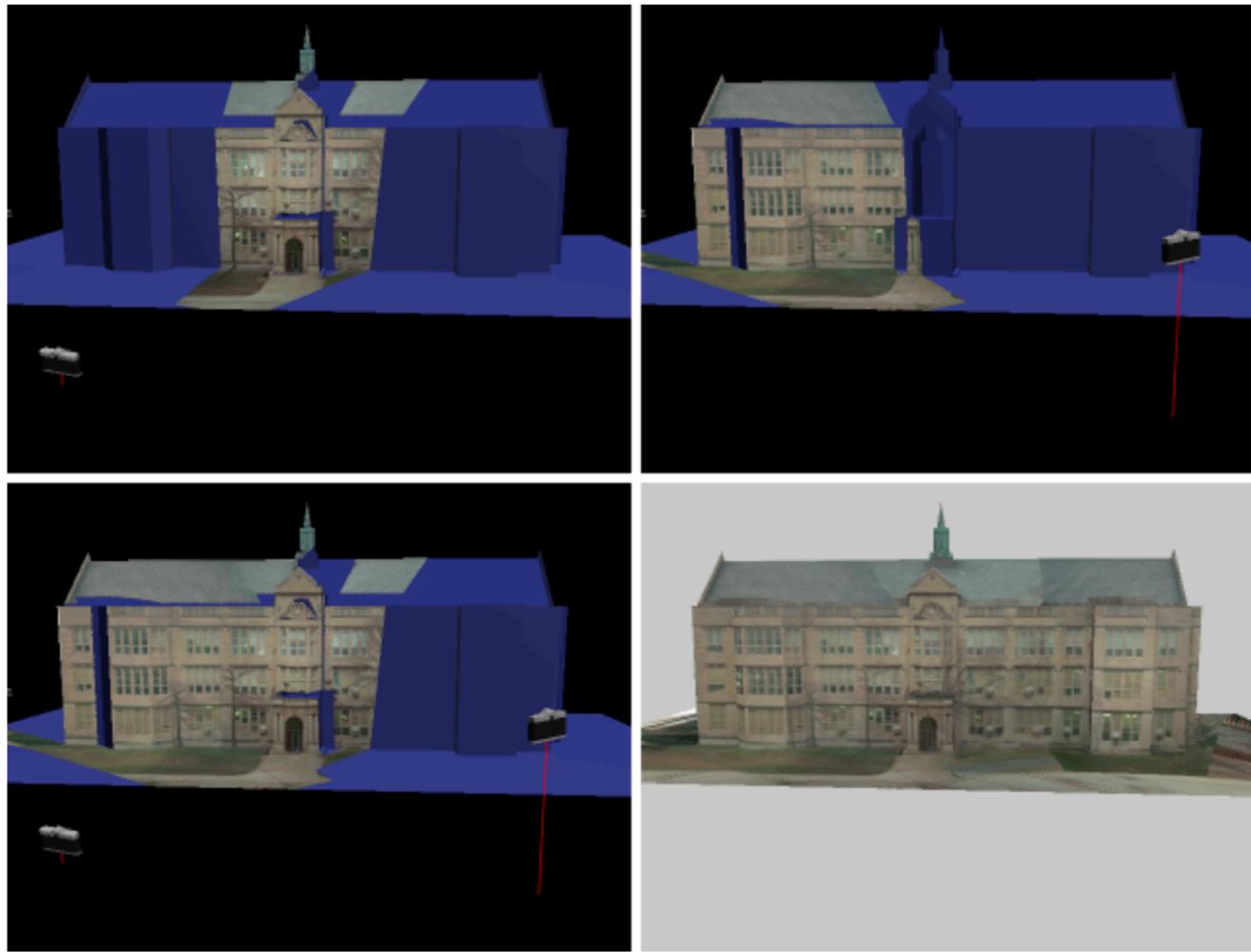
Source Image



Target Image

Matlab interp2 example: <http://www.cse.psu.edu/~rcollins/CSE486/lecture14.pdf>

Texture Stitching



The process of assembling projected images to form a composite rendering
Image from Paul Debevec's Thesis

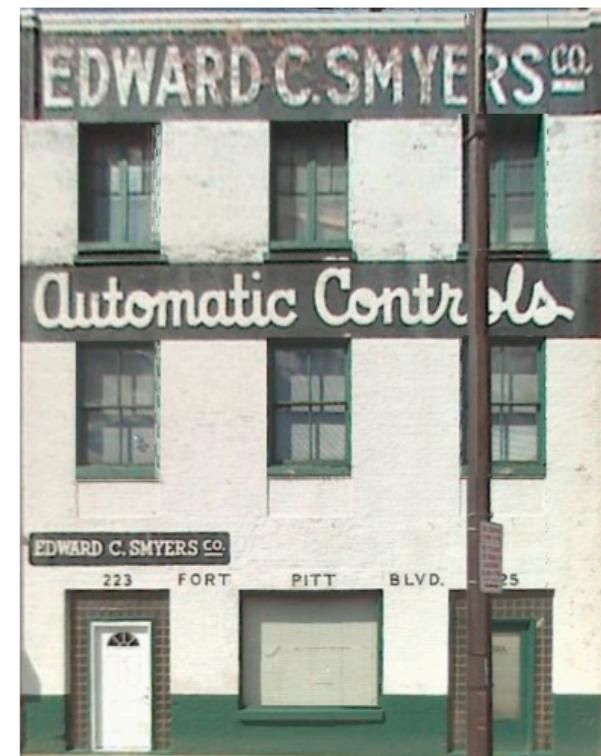
Texture Stitching



(a)



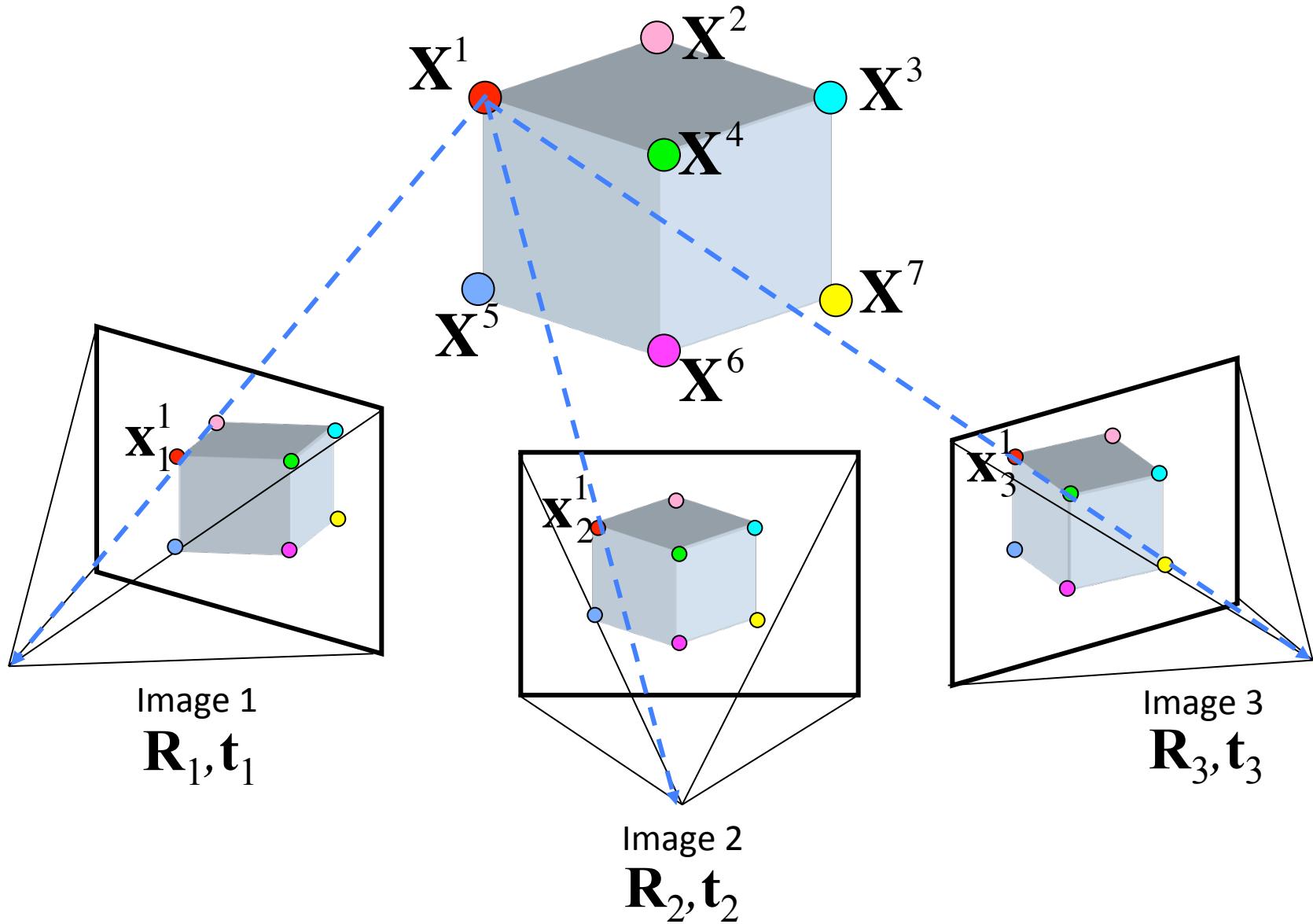
(b)



(c)

Image-based Street-side City Modeling. Xiao et al, 2009.

Texture Stitching



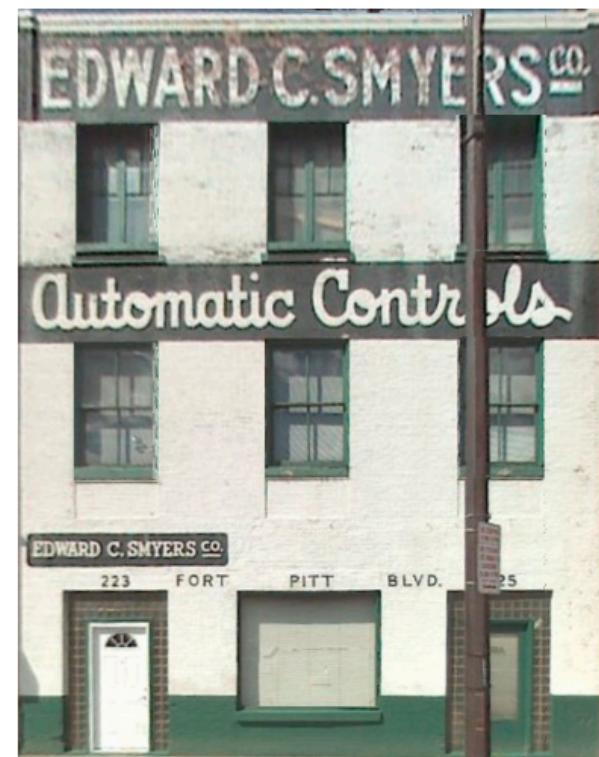
Texture Stitching



(a)



(b)



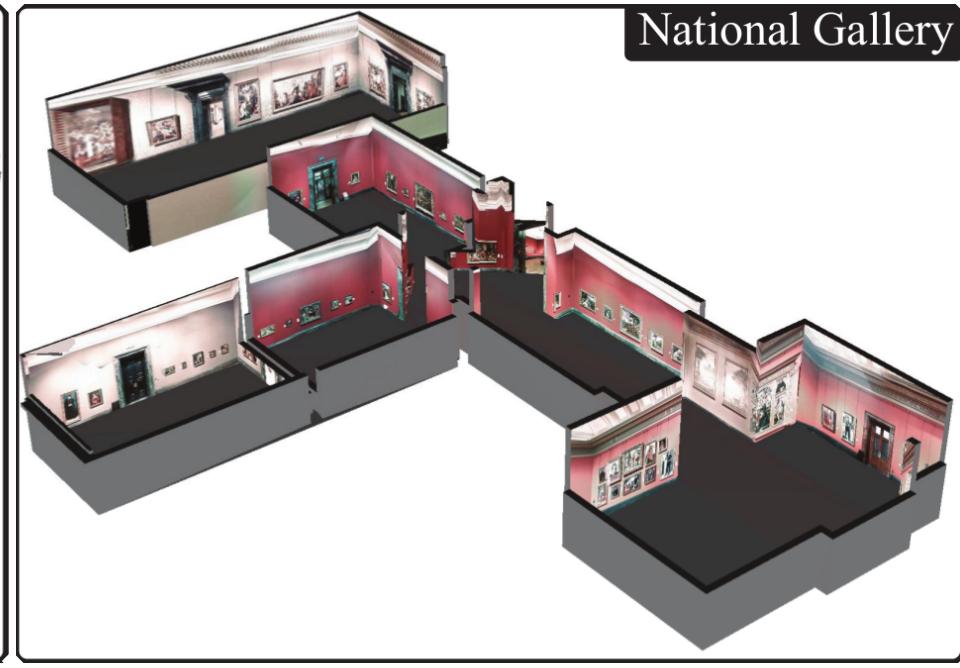
(c)

Image-based Street-side City Modeling. Xiao et al, 2009.

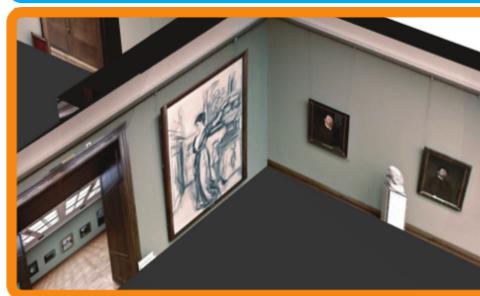
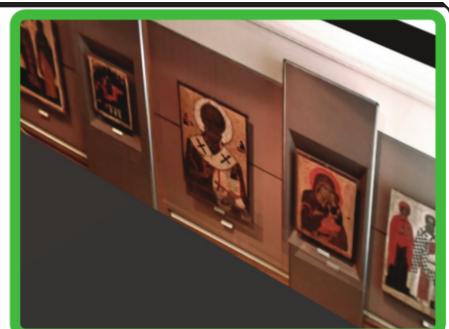
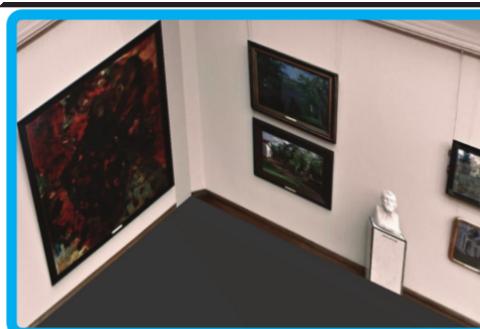
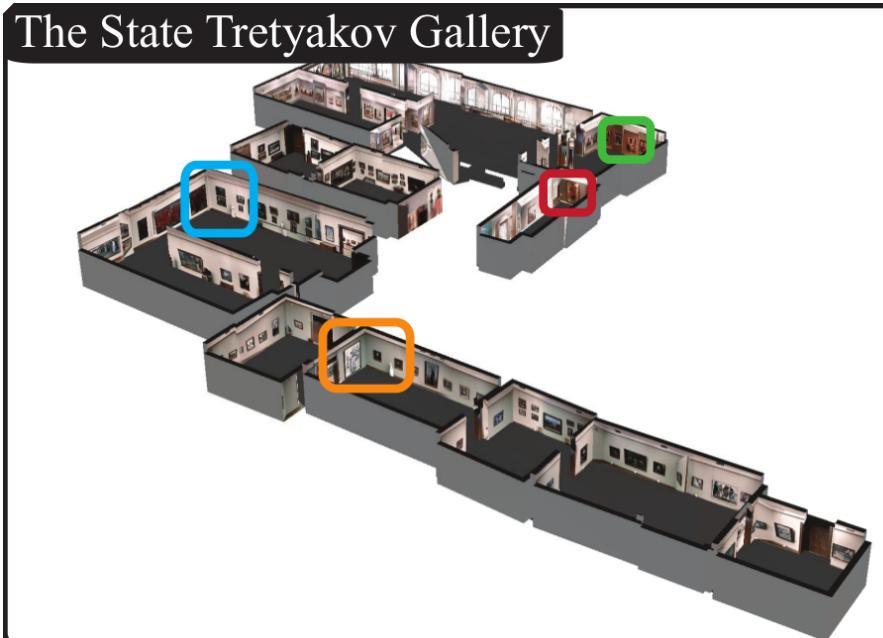
The Frick Collection



National Gallery



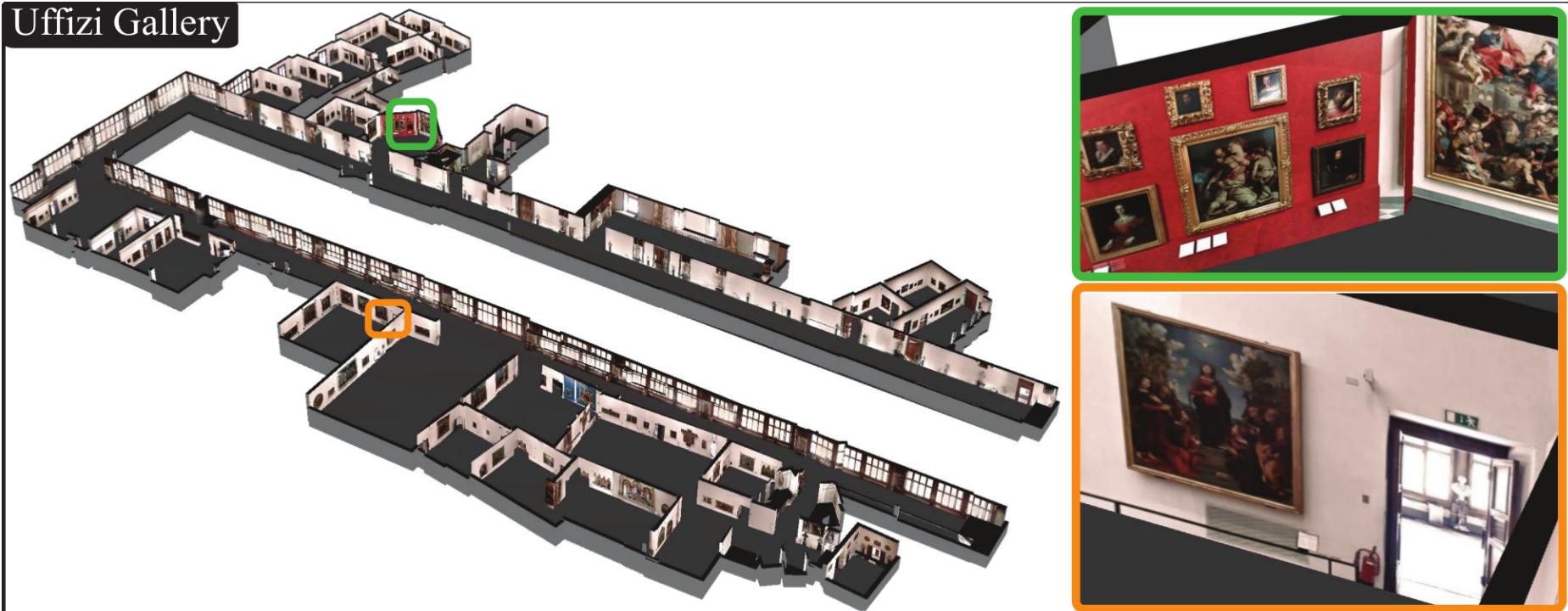
The State Tretyakov Gallery



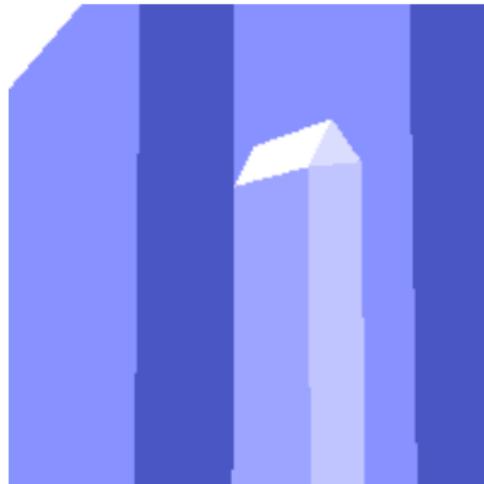
The State Hermitage Museum



Uffizi Gallery



View-Dependent Texture Mapping



(a)



(b)



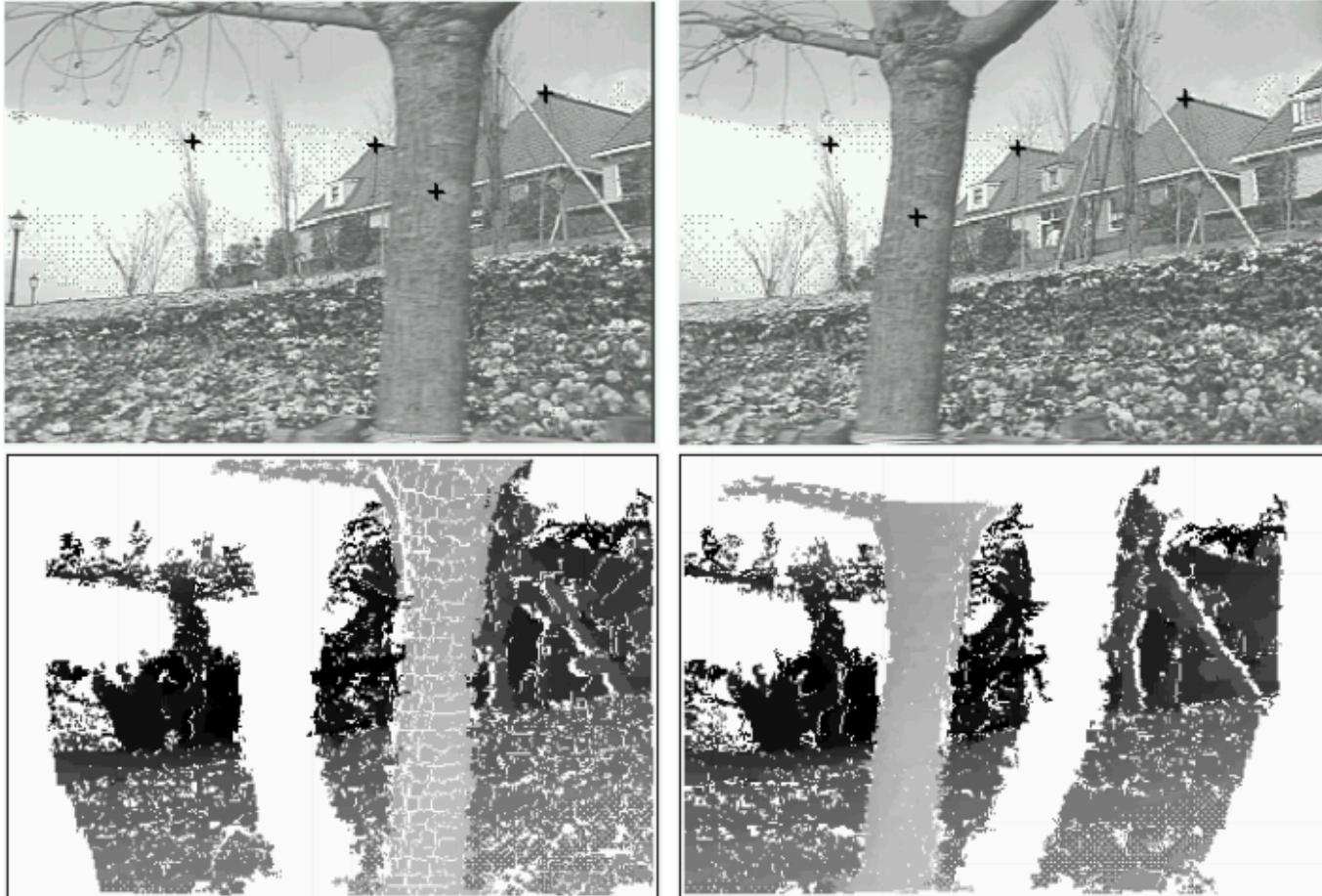
(c)



(d)

Modeling and Rendering Architecture from Photographs,
Paul Debevec, PhD Thesis, UC Berkeley, 1996

View Interpolation



Match Propagation from Image-based Modeling and Rendering
M Lhuillier and L Quan, 2012

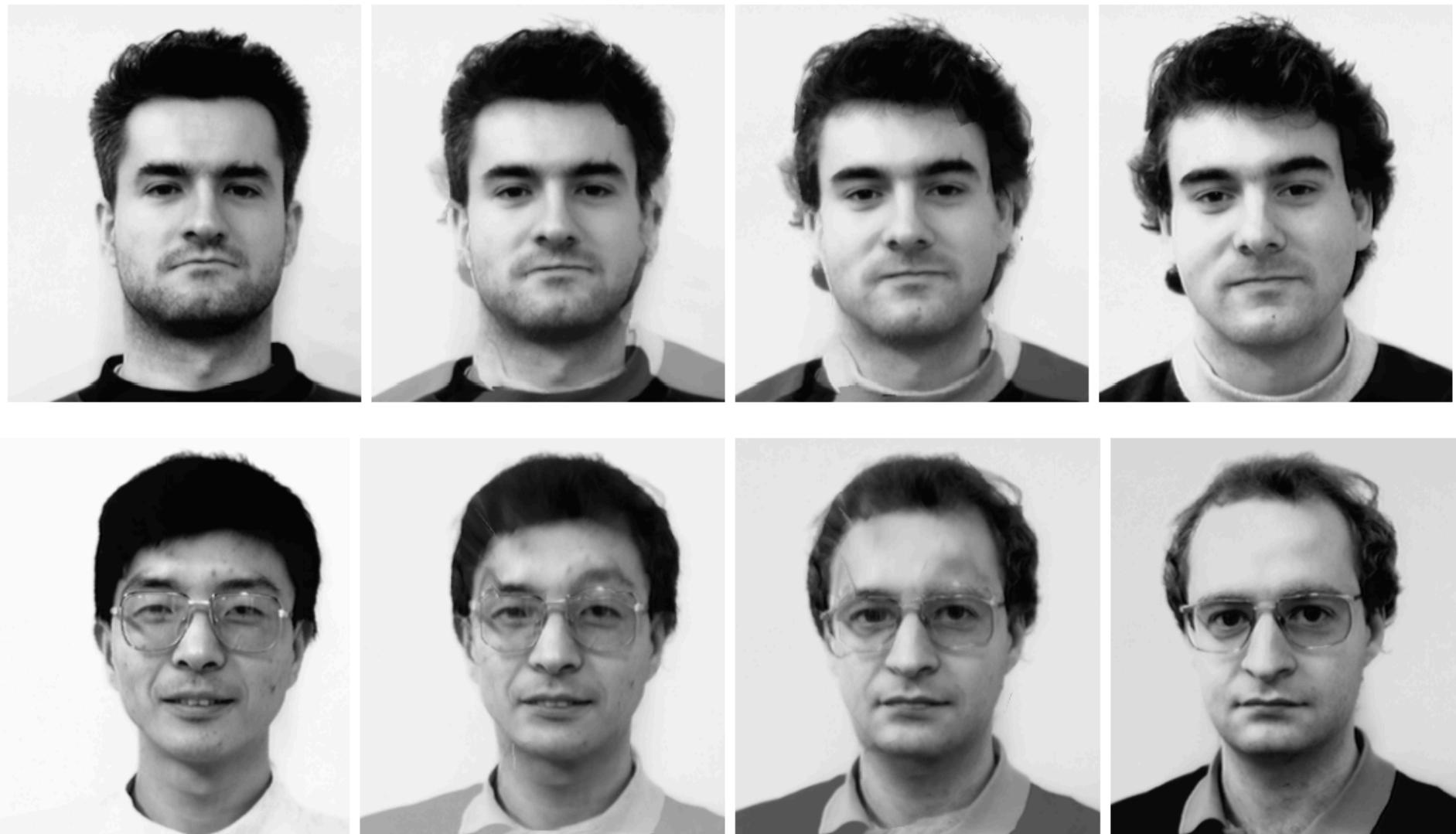
View Interpolation



View Interpolation



View Interpolation



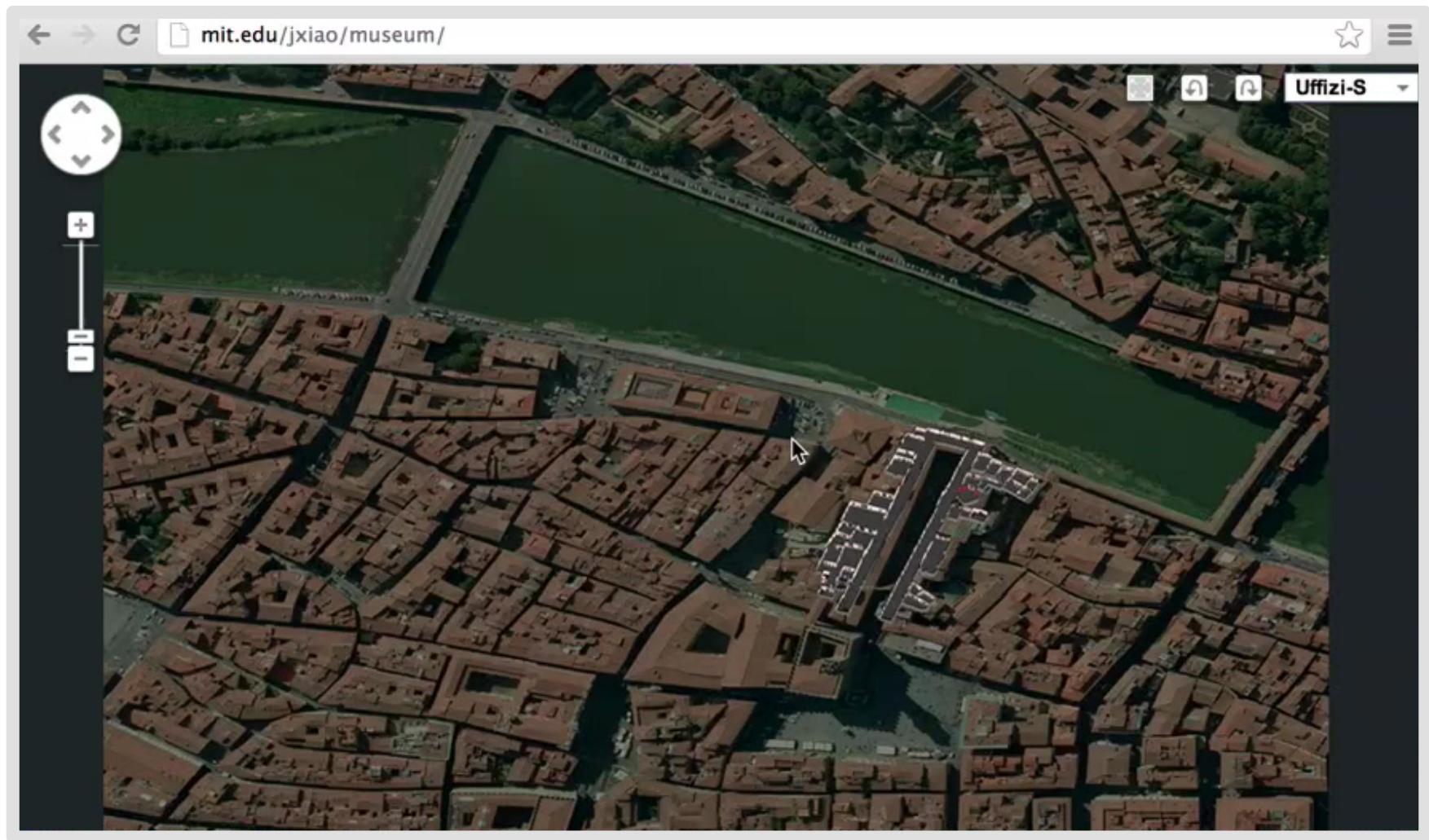
Interactive Visualization

Gallery: 492 images - 2 Mpixels

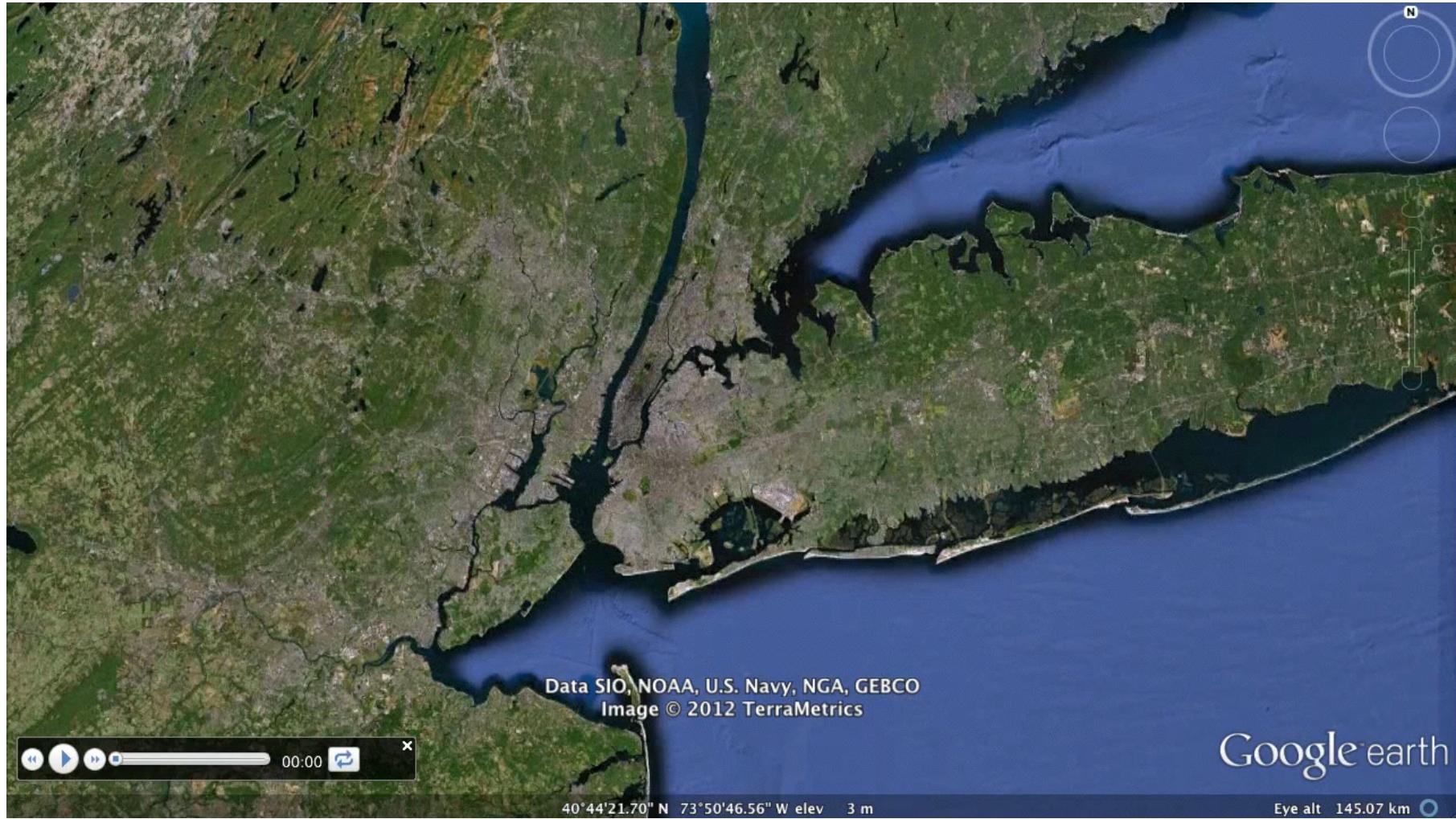


Reconstructing Building Interiors from Images, Y Furukawa, B Curless, SM Seitz, R Szeliski, 2009
Download the Viewer from: <http://grail.cs.washington.edu/projects/interior/>

Bird's-eye View Indoor Maps

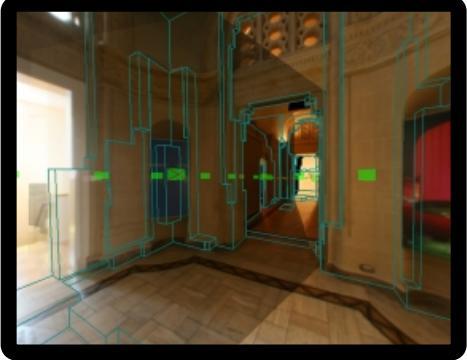
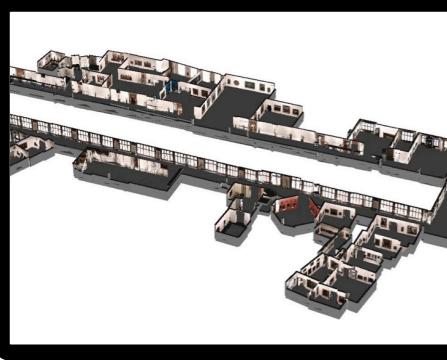


Aerial+Ground Visualization



Reconstructing the World's Museums. J. Xiao and Y. Furukawa, 2012. <http://mit.edu/jxiao/museum/>

Ground vs. Aerial → Ground + Aerial

	Ground	Aerial	Ground+Aerial
Outdoor			
Indoor			

Finish our Journey!

Images → Points:

Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes:

Model Fitting

+ Meshes → Models:

Texture Mapping

= Images → Models:

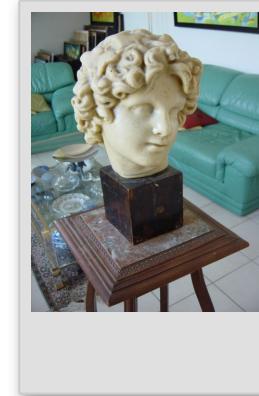
Image-based Modeling



+



+



=

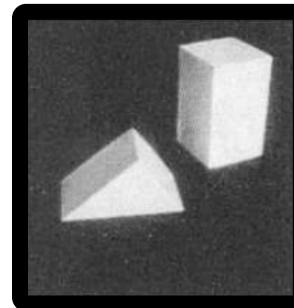


History of 3D Reconstruction (an 1-min over-simplified view)

50 years of 3D reconstruction

1960s

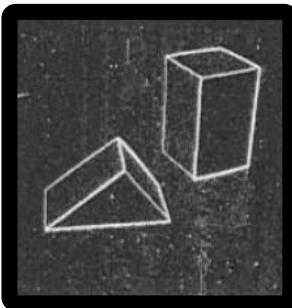
Problem Definition



Input

1970s

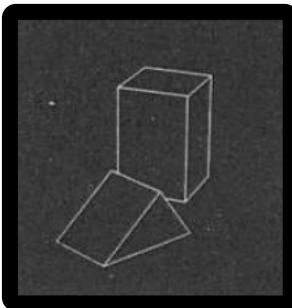
Image Formulation



Gradient

1980s

Geometry



Output

1990s

Machine Perception of Three-Dimensional Solids,
Roberts, PhD Thesis, MIT, 1963.

2000s

Reconstruction

2010s

Visualization



50 years of 3D reconstruction

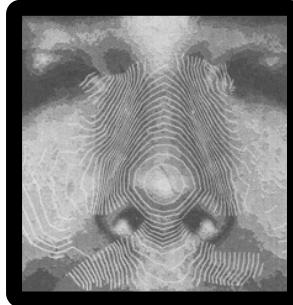
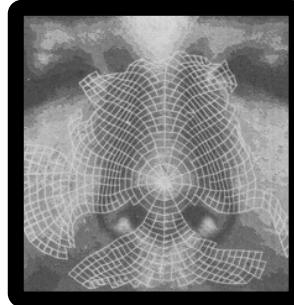
1960s

Problem Definition



1970s

Image Formulation



1980s

Geometry

Shape from Shading,
Horn, MIT AI Memos 232, 1970.

1990s

2000s Reconstruction

2010s Visualization



50 years of 3D reconstruction

1960s 1970s 1980s 1990s 2000s 2010s

Problem Definition

Image Formulation

Geometry

Reconstruction

Visualization

Essential Matrix

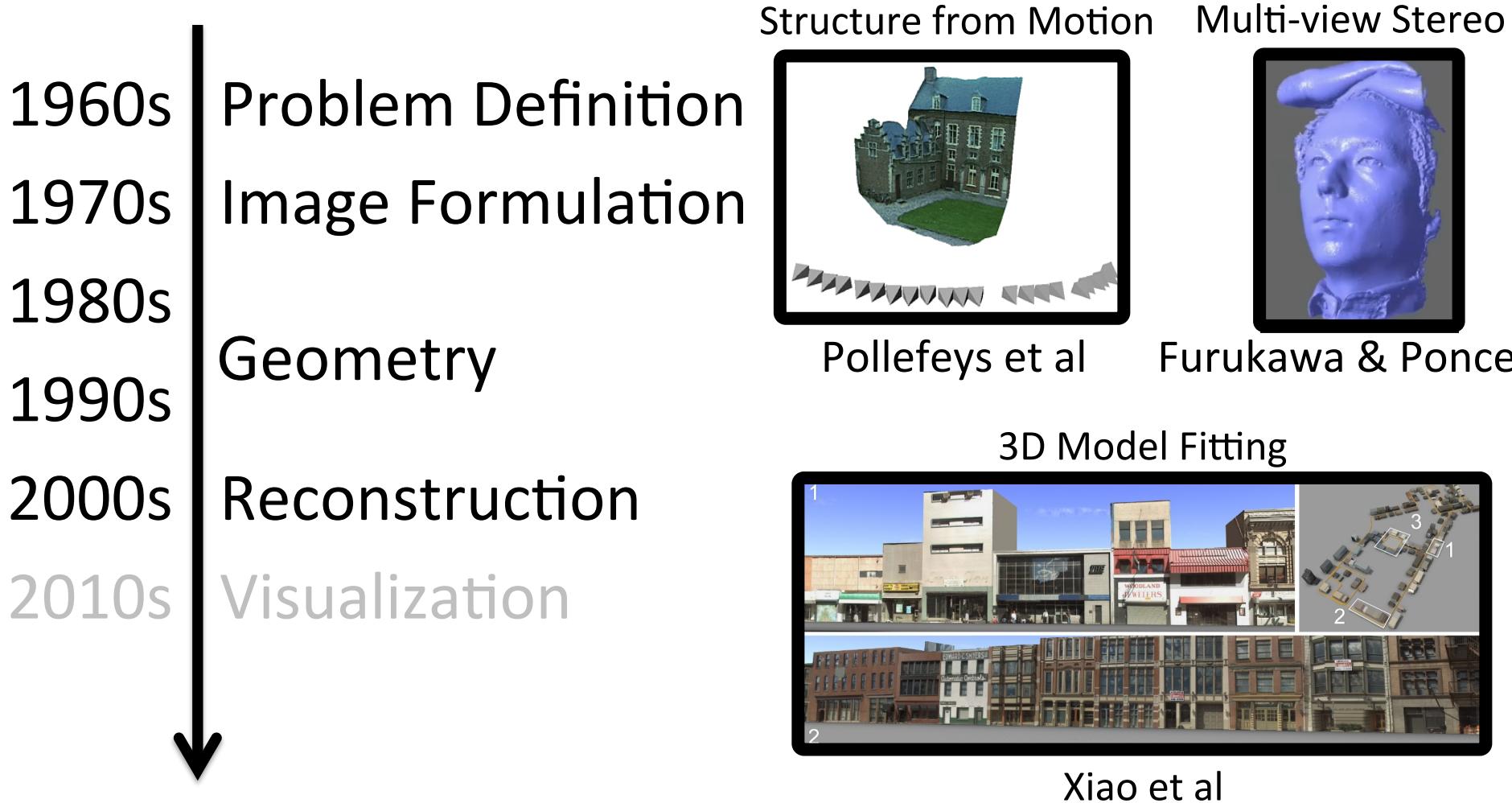
$$x'_\mu Q_{\mu\nu} x_\nu = 0 \quad (11)$$

Dividing equation (11) by $x'_3 x_3$, we arrive at the desired relationship between the image coordinates:

$$x'_\mu Q_{\mu\nu} x_\nu = 0 \quad (12)$$

A computer algorithm for reconstructing a scene from two projections, Longuet-Higgins, Nature, 1981.

50 years of 3D reconstruction



50 years of 3D reconstruction

1960s Problem Definition
1970s Image Formulation
1980s Geometry
1990s Reconstruction
2000s Visualization



Snavely et al



Furukawa et al



Xiao et al

50 years of 3D reconstruction

1960s Problem Definition
1970s Image Formulation
1980s Geometry
1990s Reconstruction
2000s Visualization

Goals of Computer Vision:

- Let machines see
- Let humans see better



Longer Summary of the History

Research Landmarks for 3D Reconstruction

Steve Seitz

"History of 3D Computer Vision"

NSF Frontiers in Computer Vision, 2011

<http://www.youtube.com/watch?v=kylzMr917Rc>

<http://homes.cs.washington.edu/~seitz/talks/3Dhistory.pdf>

Homework: Play with SFMedu

- Run the SFMedu program
- Debug the code step by step in Matlab
- Link the code and the theory we cover in this lecture
- Try some other images
- Locate the unstable part of the algorithm

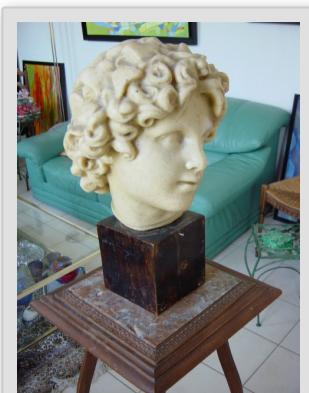
Final Project Ideas: Improve SFMedu

- 5-point algorithm
- Estimate focal length on the fly
- Beyond pairwise neighboring matching
- Tri-focal tensor
- Make it stable for different images input
- Make it work for a video input
- Make it work faster
- Check all the theories and fix all bugs ☺

Q & A



+



+



=

