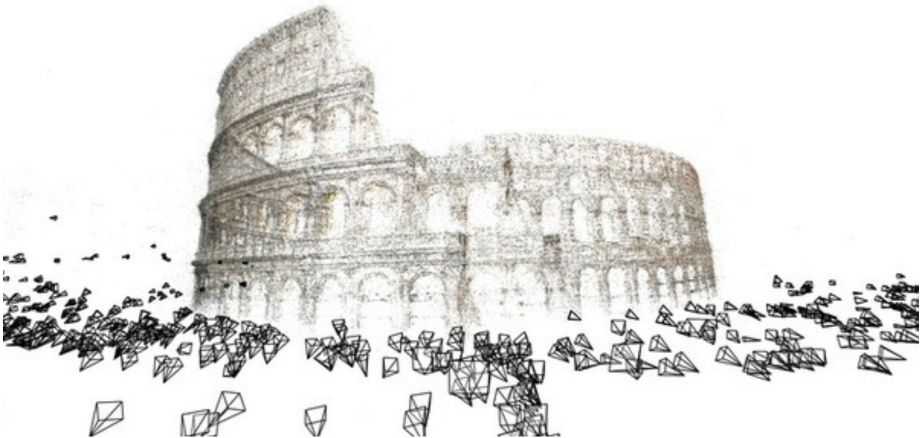
Structure from Motion & Radiance Fields

feature matching, bundle adjustment, volume rendering



CSC420
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/420
Slide credit: Kris Kitani, Noah Snavely, Rob Fergus, Yannis Gkioulekas (CMU 16-385)

TCIG
Toronto Computational Imaging Group

Announcements

- HW 3 due on Nov 13

- midterm debrief
  - option: allow midterm weight to be shifted to final exam

Course Overview

Two-view structure from motion

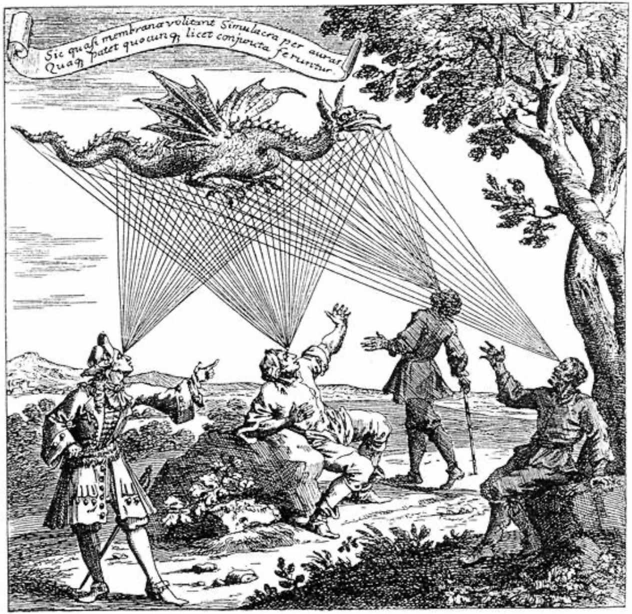
	Structure (scene geometry)	Motion (camera geometry)	Measurements
Pose Estimation	?	?	?
Triangulation		known or unknown?	
Reconstruction			

	Structure (scene geometry)	Motion (camera geometry)	Measurements
Pose Estimation	known	estimate	3D to 2D correspondences
Triangulation	?	?	?
Reconstruction			

	Structure (scene geometry)	Motion (camera geometry)	Measurements
Pose Estimation	known	estimate	3D to 2D correspondences
Triangulation	estimate	known	2D to 2D coorespondences
Reconstruction	?	?	?

	Structure (scene geometry)	Motion (camera geometry)	Measurements
Pose Estimation	known	estimate	3D to 2D correspondences
Triangulation	estimate	known	2D to 2D coorespondences
Reconstruction	estimate	estimate	2D to 2D coorespondences

Structure from motion



Драконъ, видимый подъ различными углами зрѣнія
По гравюрѣ на мѣди изъ „Oculus artificialis teledioptricus" Цана. 1702 года.

Camera calibration & triangulation

- Suppose we know 3D points
  - And have matches between these points and an image
  - How can we compute the camera parameters?

Camera calibration & triangulation

- Suppose we know 3D points
    - And have matches between these points and an image
    - How can we compute the camera parameters?
- Suppose we have know camera parameters, each of which observes a point
    - How can we compute the 3D location of that point?

Structure from motion

- SfM solves both of these problems *at once*
- A kind of chicken-and-egg problem
  - (but solvable)

Reconstruction

(2 view structure from motion)

Given a set of matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

Estimate the camera matrices

$$\mathbf{P}, \mathbf{P}'$$

Estimate the 3D point

$$\mathbf{X}$$

Reconstruction

(2 view structure from motion)

Given a set of matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

Estimate the camera matrices

$$\mathbf{P}, \mathbf{P}'$$

'motion'

(of the cameras)

Estimate the 3D point

$$\mathbf{X}$$

'structure'

Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences

$$\boldsymbol{x}_m'^{\top} \mathbf{F} \boldsymbol{x}_m = 0$$

Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

$$\boldsymbol{x}_m'^{\top} \mathbf{F} \boldsymbol{x}_m = 0$$

Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\, \mathbf{I} \mid \mathbf{0} \,]$ and $\mathbf{P}' = [\, [\mathbf{e}_x]\mathbf{F} \mid \mathbf{e}' \,]$

Camera matrices corresponding to the fundamental matrix **F** may be chosen as

$$\mathbf{P} = [\mathbf{I}|\mathbf{0}] \qquad \mathbf{P}' = [[\boldsymbol{e}_\times]\mathbf{F}|\boldsymbol{e}']$$

(See Hartley and Zisserman C.9 for proof)

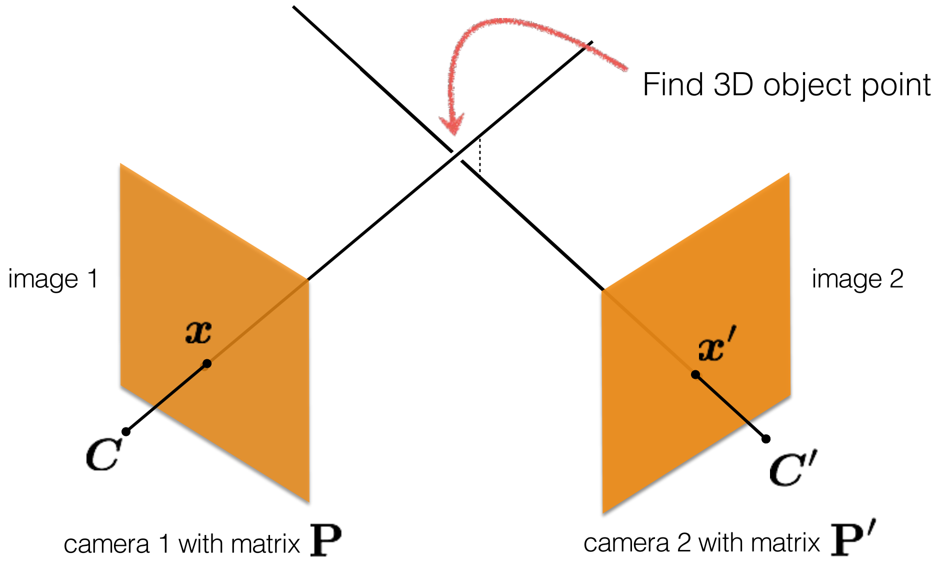
Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\,\mathbf{I} \mid \mathbf{0}\,]$ and $\mathbf{P}' = [\,[\mathbf{e}'_{\times}]\mathbf{F} \mid \mathbf{e}'\,]$

3. For each point correspondence, compute the point **X** in 3D space (triangulation)
   Triangulation with $x = \mathbf{P}\,X$ and $x' = \mathbf{P}'\,X$

Triangulation



Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\, \mathbf{I} \mid \mathbf{0} \,]$ and $\mathbf{P}' = [\, [\mathbf{e}'_{\times}]\mathbf{F} \mid \mathbf{e}' \,]$

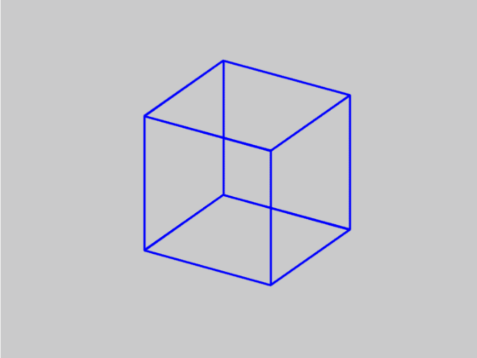
3. For each point correspondence, compute the point **X** in 3D space (triangularization)
   SVD with $x = \mathbf{P}\, X$ and $x' = \mathbf{P}'\, X$

Is SfM always uniquely solvable?

Ambiguities in structure from motion

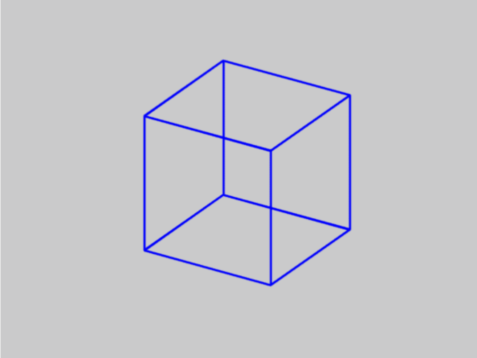
Is SfM always uniquely solvable?

- No…



Is SfM always uniquely solvable?

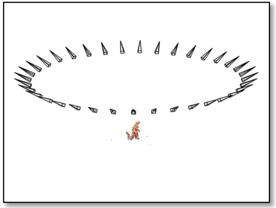
- No…

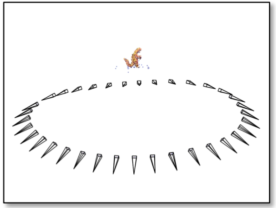


- do you see rotating left to right? Or right to left?

SfM – Failure cases

- Necker reversal





Projective Ambiguity

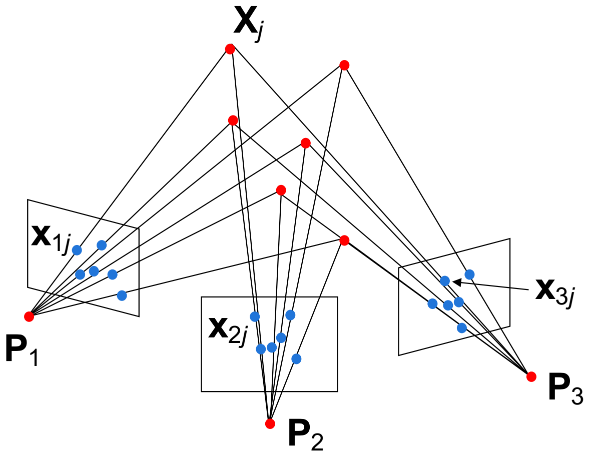
- Reconstruction is ambiguous by an arbitrary 3D projective transformation without prior knowledge of camera parameters

Structure from motion

- Given: *m* images of *n* fixed 3D points

$$\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j , \qquad i = 1, \ldots , m, \quad j = 1, \ldots , n$$

- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* correspondences $\mathbf{x}_{ij}$



Structure from motion ambiguity

- If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same:

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\frac{1}{k}\mathbf{P}\right)(k\,\mathbf{X})$$

Structure from motion ambiguity

- If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same:

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\frac{1}{k}\mathbf{P}\right)(k\,\mathbf{X})$$

Can we recover the absolute scale of a scene?

Structure from motion ambiguity

- If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same:

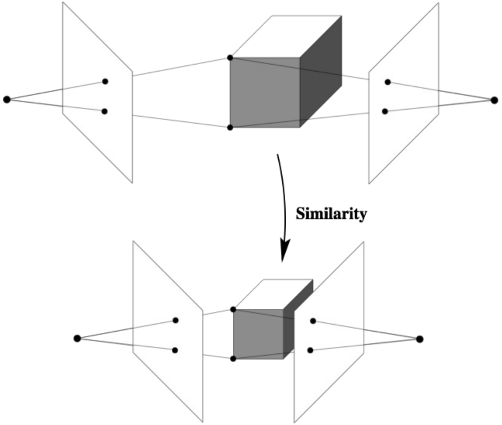
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\frac{1}{k}\mathbf{P}\right)(k\,\mathbf{X})$$

Can we recover the absolute scale of a scene?
No!

Structure from motion ambiguity

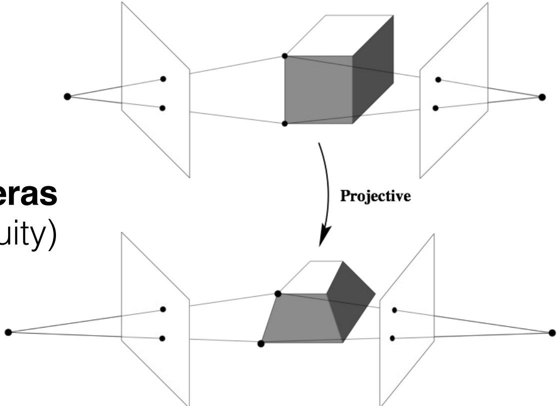
- If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same
- More generally: if we transform the scene using a transformation **Q** and apply the inverse transformation to the camera matrices, then the images do not change

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)\left(\mathbf{Q}\mathbf{X}\right)$$

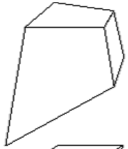
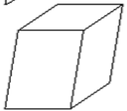
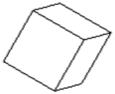
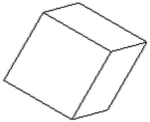


Calibrated cameras
(similarity projection ambiguity)

Uncalibrated cameras
(projective projection ambiguity)



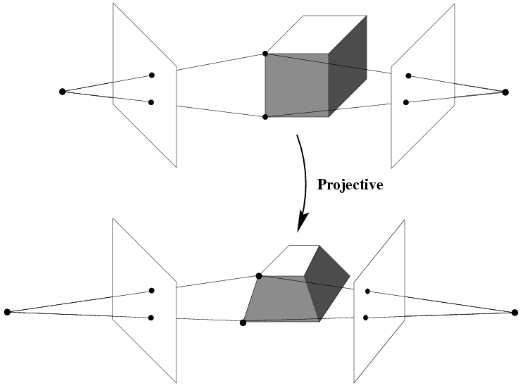
Types of ambiguity

Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, ratios of length
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, lengths

- With no constraints on the camera calibration matrix or on the scene, we get a *projective* reconstruction
- Need additional information to *upgrade* the reconstruction to affine, similarity, or Euclidean

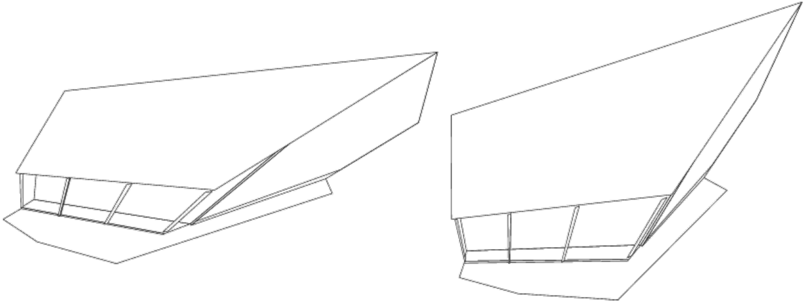
Slide: S. Lazebnik

Projective ambiguity

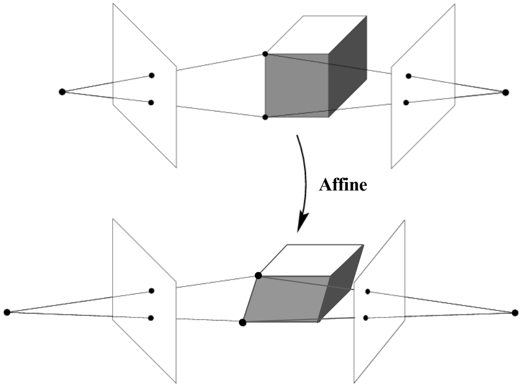


$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_{\mathbf{P}}^{-1}\right)\left(\mathbf{Q}_{\mathbf{P}}\,\mathbf{X}\right)$$

Projective ambiguity



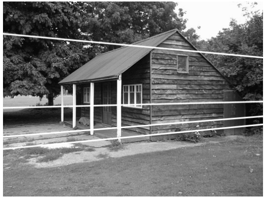
Affine ambiguity

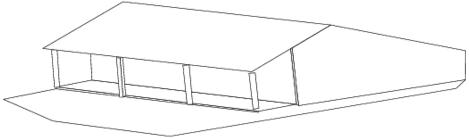


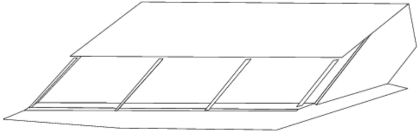
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_{\mathbf{A}}^{-1}\right)\left(\mathbf{Q}_{\mathbf{A}}\,\mathbf{X}\right)$$

Affine ambiguity

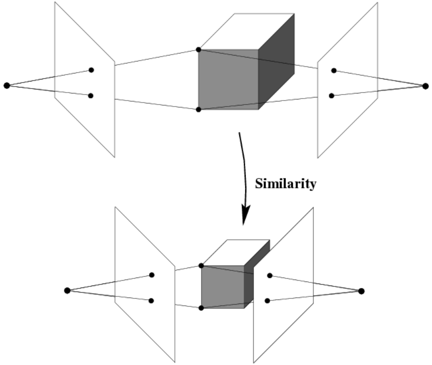






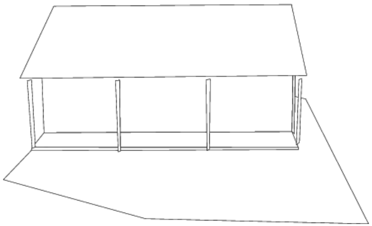
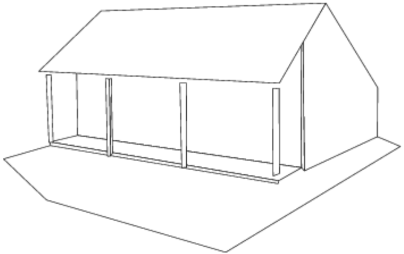
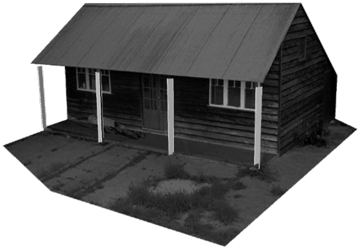


Similarity ambiguity



$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_S^{-1}\right)\left(\mathbf{Q}_S\mathbf{X}\right)$$

Similarity ambiguity

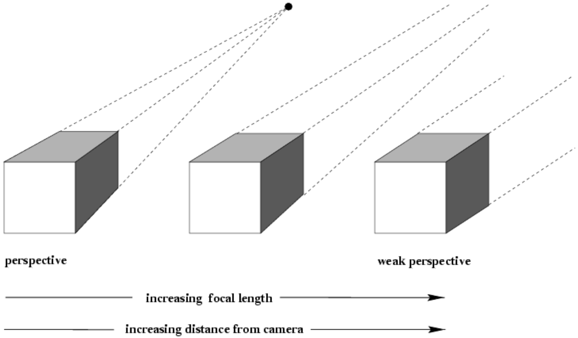





What can we do to remove ambiguities?

Affine structure from motion

Structure from motion

- Let's start with *affine cameras* (the math is easier)

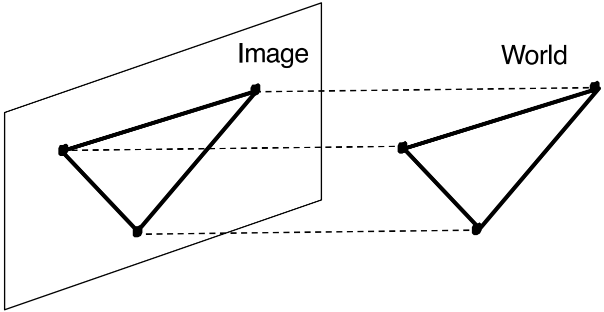


center at infinity

Recall: Orthographic Projection

Special case of perspective projection

- Distance from center of projection to image plane is infinite

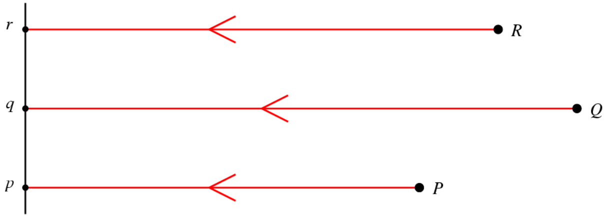


- Projection matrix:

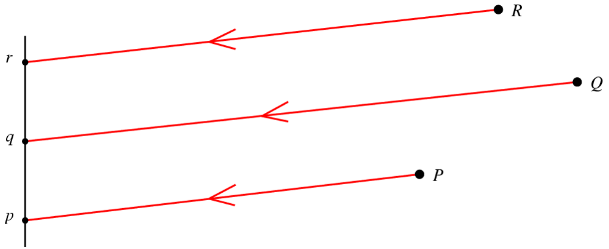
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

Affine cameras

Orthographic Projection



Parallel Projection



Affine cameras

- A general affine camera combines the effects of an affine transformation of the 3D space, orthographic projection, and an affine transformation of the image:

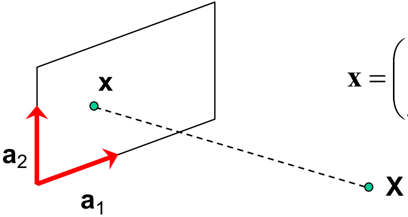
$$\mathbf{P} = [3\times 3\,\text{affine}]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}[4\times 4\,\text{affine}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Affine cameras

- A general affine camera combines the effects of an affine transformation of the 3D space, orthographic projection, and an affine transformation of the image:

$$\mathbf{P} = [3\times 3\,\text{affine}]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}[4\times 4\,\text{affine}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

- Affine projection is a linear mapping + translation in inhomogeneous coordinates



$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \mathbf{AX} + \mathbf{b}$$

Projection of world origin

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \dots, m, \ j = 1, \dots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \dots, m, \; j = 1, \dots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix} \rightarrow \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix}$$

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \dots, m, \; j = 1, \dots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \to \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \to \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$$

how many knowns?

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \dots, m, \; j = 1, \dots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix} \rightarrow \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix}$$

- We have 2*mn* knowns — *n* measured 2D point locations in each of the *m* images

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \dots, m, \; j = 1, \dots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix} \rightarrow \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix}$$

- We have 2*mn* knowns — *n* measured 2D point locations in each of the *m* images

how many unknowns?

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1,\dots, m, \; j = 1, \dots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix} \rightarrow \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix}$$

- We have 2*mn* knowns — *n* measured 2D point locations in each of the *m* images
- 8*m* + 3*n* - 12 unknowns — 8 affine transformations for each of the m images and *n* 3D points minus 12 dof for affine ambiguity

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i\,, \qquad i = 1, \ldots, m, \;\; j = 1, \ldots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix} \rightarrow \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix}$$

- We have 2*mn* knowns and 8*m* + 3*n* -12 unknowns

Affine structure from motion

- Given: *m* images of *n* fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \dots, m, \; j = 1, \dots, n$$

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \to \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix} \to \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ \mathbf{1} \end{pmatrix}$$

- We have 2*mn* knowns and 8*m* + 3*n* -12 unknowns
- Thus, we must have 2*mn* >= 8*m* + 3*n* – 12
- For *m* = two views, we need *n* = four point correspondences
  2 * 2 * 4 = 16 >= 8*2 + 3*4 – 12 = 16

Affine structure from motion

- Centering: subtract the centroid of the image points

$$\hat{\mathbf{x}}_{ij} = \mathbf{x}_{ij} - \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_{ik} = \mathbf{A}_i\mathbf{X}_j + \mathbf{b}_i - \frac{1}{n}\sum_{k=1}^{n}\left(\mathbf{A}_i\mathbf{X}_k + \mathbf{b}_i\right)$$

$$= \mathbf{A}_i\left(\mathbf{X}_j - \frac{1}{n}\sum_{k=1}^{n}\mathbf{X}_k\right) = \mathbf{A}_i\hat{\mathbf{X}}_j$$

- For simplicity, assume that the origin of the world coordinate system is at the centroid of the 3D points

Affine structure from motion

- Centering: subtract the centroid of the image points

$$\hat{\mathbf{x}}_{ij} = \mathbf{x}_{ij} - \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_{ik} = \mathbf{A}_i\mathbf{X}_j + \mathbf{b}_i - \frac{1}{n}\sum_{k=1}^{n}\left(\mathbf{A}_i\mathbf{X}_k + \mathbf{b}_i\right)$$

$$= \mathbf{A}_i\left(\mathbf{X}_j - \frac{1}{n}\sum_{k=1}^{n}\mathbf{X}_k\right) = \mathbf{A}_i\hat{\mathbf{X}}_j$$

- For simplicity, assume that the origin of the world coordinate system is at the centroid of the 3D points
- After centering, each normalized point $\mathbf{x}_{ij}$ is related to the 3D point $\mathbf{X}_i$ by

$$\hat{\mathbf{x}}_{ij} = \mathbf{A}_i\mathbf{X}_j$$

Affine structure from motion

- Let's create a $2m \times n$ data (measurement) matrix:

$$
\begin{array}{cc}
\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix} & \Big\downarrow \text{cameras } (2m) \\
\xrightarrow{\text{points } (n)} &
\end{array}
$$

C. Tomasi and T. Kanade. [Shape and motion from image streams under orthography: A factorization method.]() *IJCV*, 9(2):137-154, November 1992.

Affine structure from motion

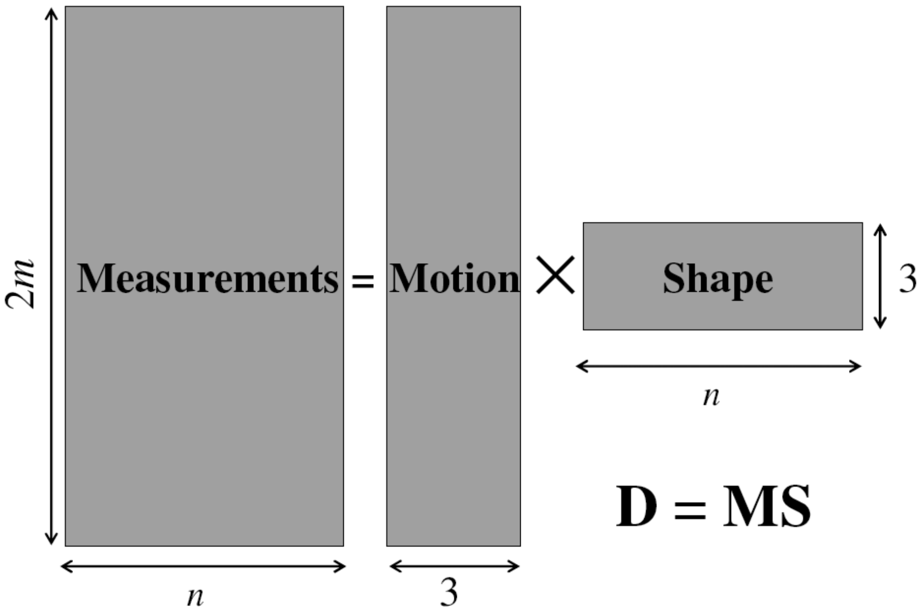
- Let's create a $2m \times n$ data (measurement) matrix:

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}}_{\text{cameras } (2m \times 3)} \underbrace{\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}}_{\text{points } (3 \times n)}$$

The measurement matrix $\mathbf{D} = \mathbf{MS}$ must have rank 3!

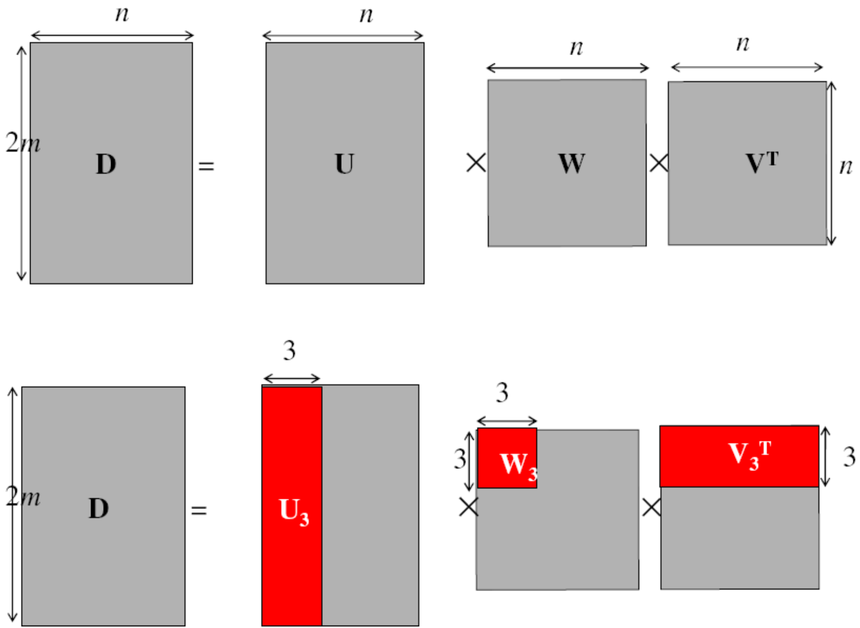
C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137-154, November 1992.

Factorizing the measurement matrix



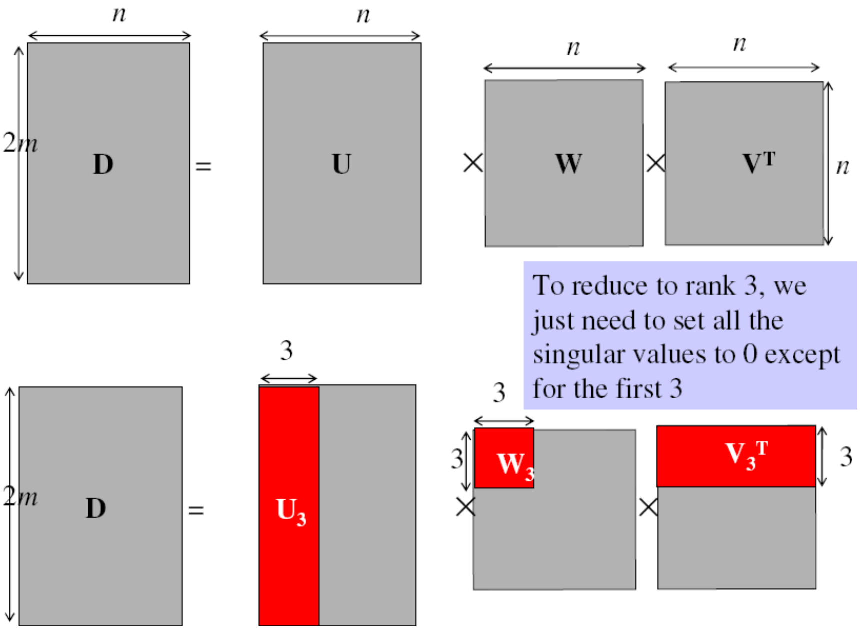
Factorizing the measurement matrix

- Singular value decomposition of D:



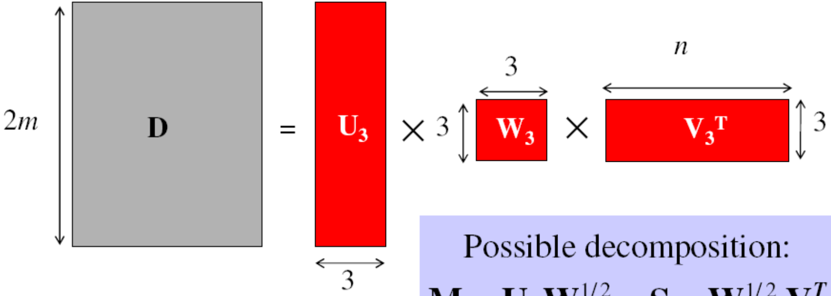
Factorizing the measurement matrix

- Singular value decomposition of D:



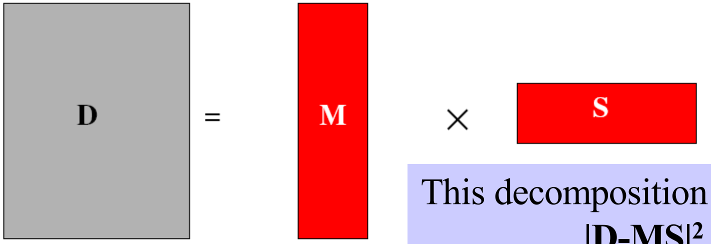
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



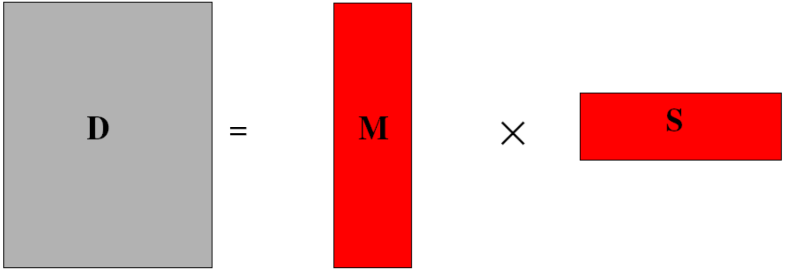
Possible decomposition:

$$\mathbf{M} = \mathbf{U}_3 \mathbf{W}_3^{1/2} \qquad \mathbf{S} = \mathbf{W}_3^{1/2}\, \mathbf{V}_3^T$$



This decomposition minimizes $|\mathbf{D}\text{-}\mathbf{MS}|^2$

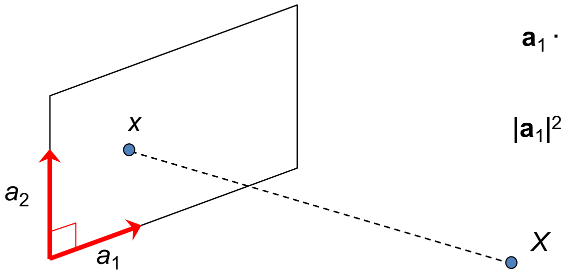
Affine ambiguity



- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations **M** → **MC**, **S** → $\mathbf{C}^{-1}\mathbf{S}$
- That is because we have only an affine transformation and we have not enforced any Euclidean constraints (like forcing the image axes to be perpendicular, for example)

Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and of unit length



$\mathbf{a}_1 \cdot \mathbf{a}_2 = 0$

$|\mathbf{a}_1|^2 = |\mathbf{a}_2|^2 = 1$

Solve for orthographic constraints

Three equations for each image i

$$\tilde{\mathbf{a}}_{i1}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i1}^T = 1$$
$$\tilde{\mathbf{a}}_{i2}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i2}^T = 1 \quad \text{where} \quad \tilde{\mathbf{A}}_i = \begin{bmatrix} \tilde{\mathbf{a}}_{i1}^T \\ \tilde{\mathbf{a}}_{i2}^T \end{bmatrix}$$
$$\tilde{\mathbf{a}}_{i1}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i2}^T = 0$$

- Solve for **L = CC^T**
- Recover **C** from **L** by Cholesky decomposition: **L = CC^T**
- Update **A** and **X**: **A = AC, X = C^{-1}X**

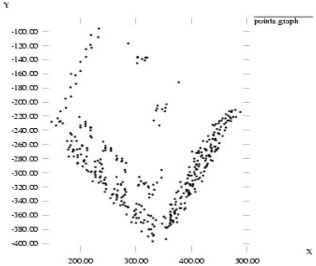
Algorithm summary

- Given: m images and n features $\mathbf{x}_{ij}$
- For each image i, center the feature coordinates
- Construct a $2m \times n$ measurement matrix $\mathbf{D}$:
  - Column j contains the projection of point j in all views
  - Row i contains one coordinate of the projections of all the n points in image i
- Factorize $\mathbf{D}$:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^{\mathbf{T}}$
  - Create $\mathbf{U}_3$ by taking the first 3 columns of $\mathbf{U}$
  - Create $\mathbf{V}_3$ by taking the first 3 columns of $\mathbf{V}$
  - Create $\mathbf{W}_3$ by taking the upper left 3×3 block of $\mathbf{W}$
- Create the motion and shape matrices:
  - $\mathbf{M} = \mathbf{U}_3\mathbf{W}_3^{1/2}$ and $\mathbf{S} = \mathbf{W}_3^{1/2}\,\mathbf{V}_3^{T}$ (**or** $\mathbf{M} = \mathbf{U}_3$ and $\mathbf{S} = \mathbf{W}_3\mathbf{V}_3^{T}$)
- Eliminate affine ambiguity

Reconstruction results

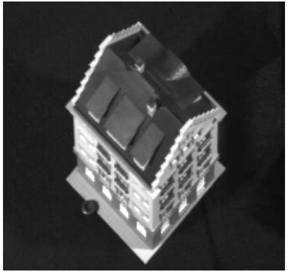
1

60



120

150



C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137-154, November 1992.

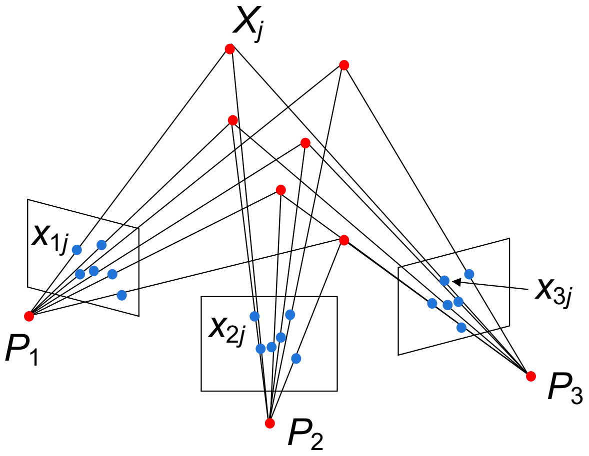
Multi-view projective structure from motion

Projective structure from motion

- Given: *m* images of *n* fixed 3D points

$$z_{ij}\,\mathbf{x}_{ij} = \mathbf{P}_i\,\mathbf{X}_j\,,\quad i = 1,\ldots,m,\quad j = 1,\ldots,n$$

- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* correspondences $\mathbf{x}_{ij}$



Projective structure from motion

- Given: *m* images of *n* fixed 3D points

$$z_{ij}\,\mathbf{x}_{ij} = \mathbf{P}_i\,\mathbf{X}_j\,, \quad i = 1,\dots, m, \quad j = 1, \dots, n$$

- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* correspondences $\mathbf{x}_{ij}$
- With no calibration info, cameras and points can only be recovered up to a 4x4 projective transformation **Q**:

$$\mathbf{X} \rightarrow \mathbf{QX},\ \mathbf{P} \rightarrow \mathbf{PQ}^{-1}$$

- We can solve for structure and motion when

$$2mn \geq 11m + 3n - 15$$

- For two cameras, at least 7 points are needed

Projective SFM: Two-camera case

- Compute fundamental matrix $\mathbf{F}$ between the two views
- First camera matrix: $[\mathbf{I}|\mathbf{0}]$
- Second camera matrix: $[\mathbf{A}|\mathbf{b}]$
- Then $\mathbf{b}$ is the epipole ($\mathbf{F}^T\mathbf{b} = 0$), $\mathbf{A} = -[\mathbf{b}_\times]\mathbf{F}$

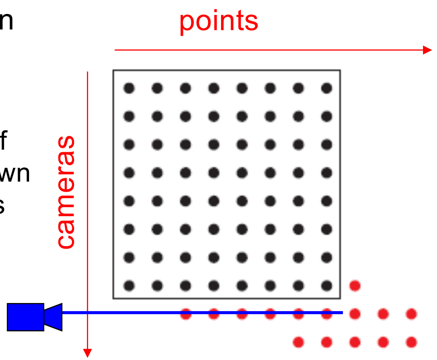
Sequential structure from motion

•Initialize motion from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*



Sequential structure from motion

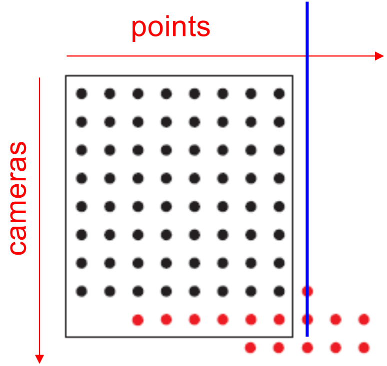
•Initialize motion from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*

- Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*



Sequential structure from motion

•Initialize motion from two images using fundamental matrix

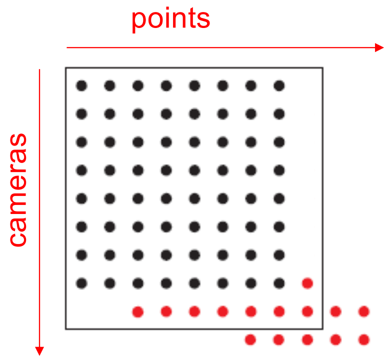
•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*

- Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*

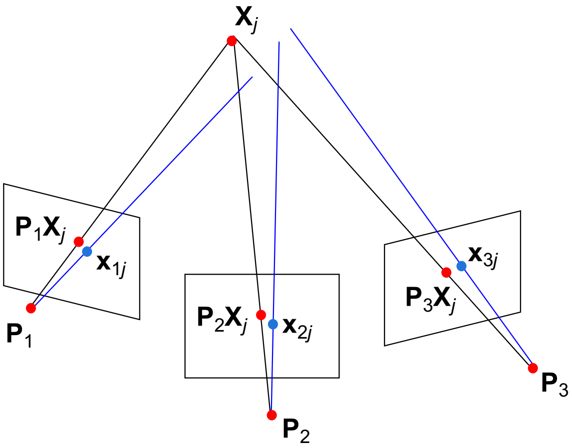
•Refine structure and motion: bundle adjustment



Bundle adjustment

- Non-linear method for refining structure and motion
- Minimizing reprojection error

$$E(\mathbf{P},\mathbf{X}) = \sum_{i=1}^{m}\sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, \mathbf{P}_i\mathbf{X}_j\right)^2$$

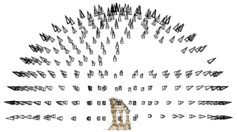


Review: Structure from motion

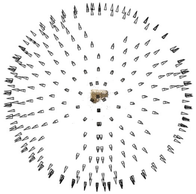
- Ambiguity
- Affine structure from motion
  - Factorization
- Dealing with missing data
  - Incremental structure from motion
- Projective structure from motion
  - Bundle adjustment

Large-scale structure from motion

Structure from motion



Reconstruction (side)

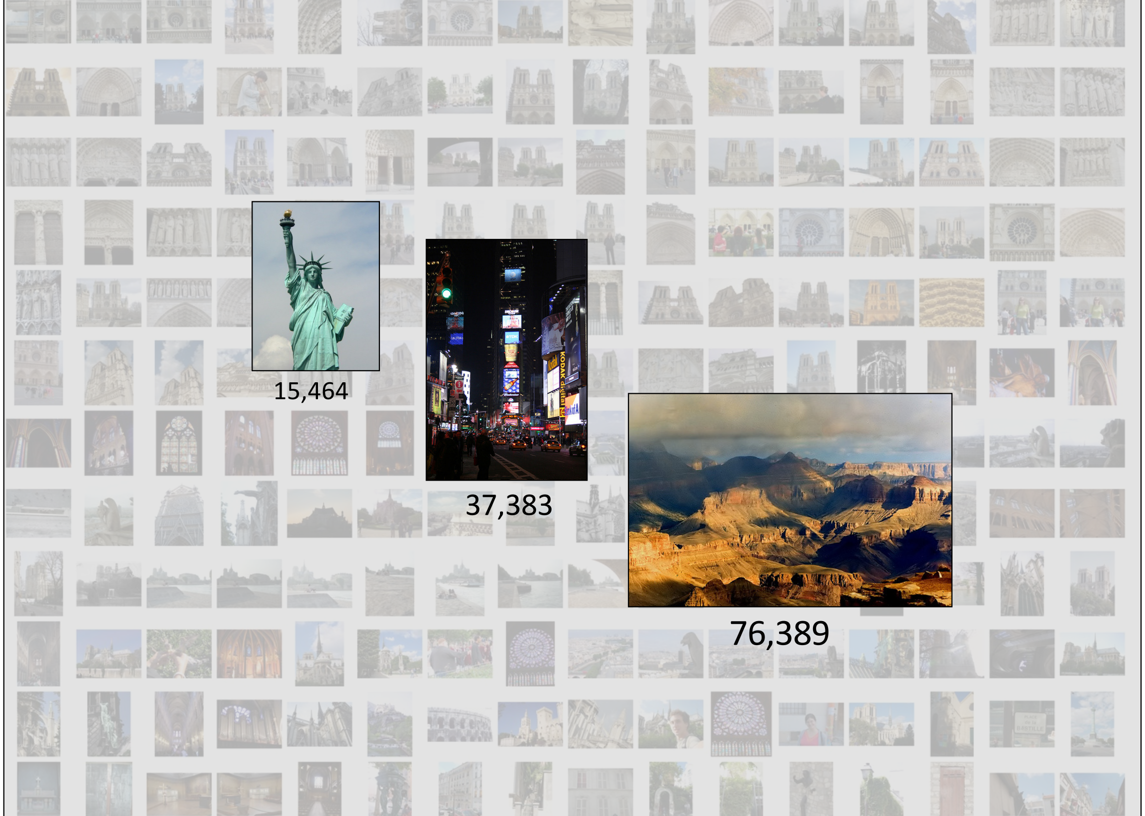


(top)

- Input: images with points in correspondence
  $p_{i,j} = (u_{i,j}, v_{i,j})$

- Output
  - structure: 3D location $\mathbf{x}_i$ for each point p_i
  - motion: camera parameters $\mathbf{R}_j$, $\mathbf{t}_j$ possibly $\mathbf{K}_j$

- Objective function: minimize *reprojection error*


15,464
37,383
76,389

Standard way to view photos

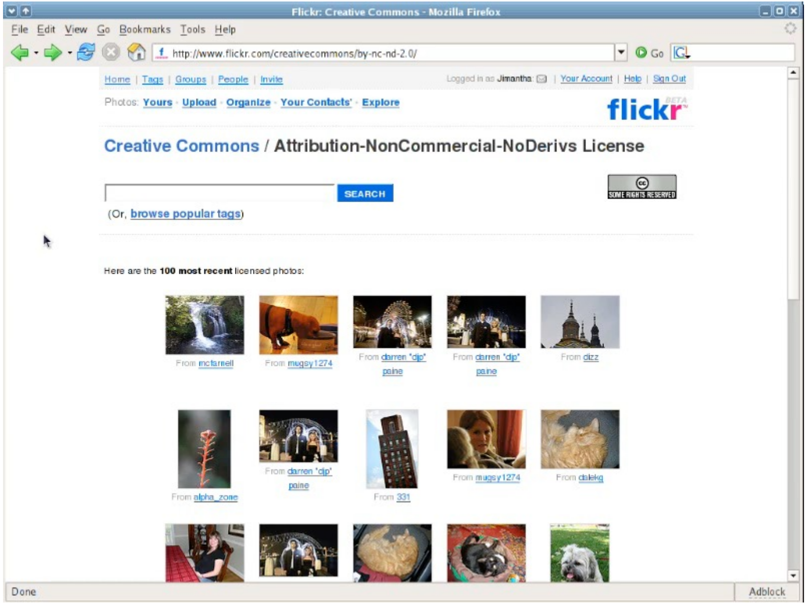
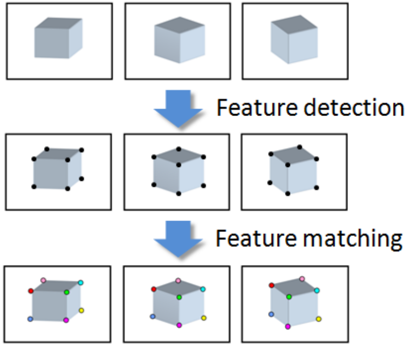


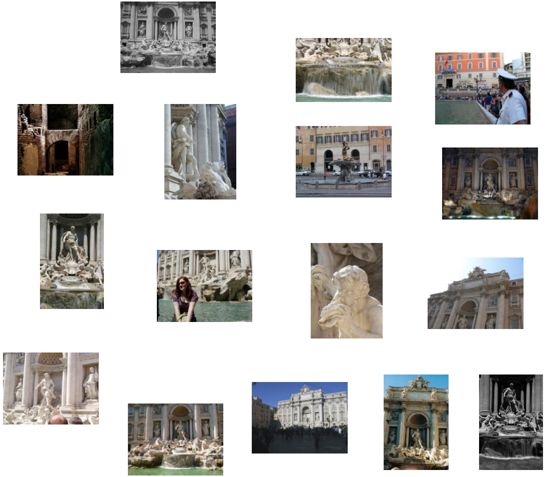
Photo Tourism

Input: Point correspondences



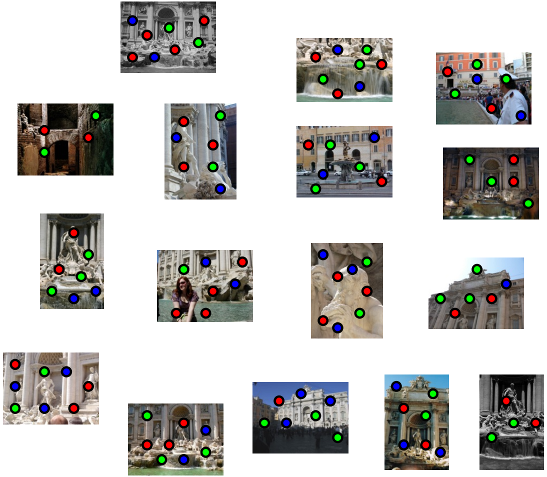
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



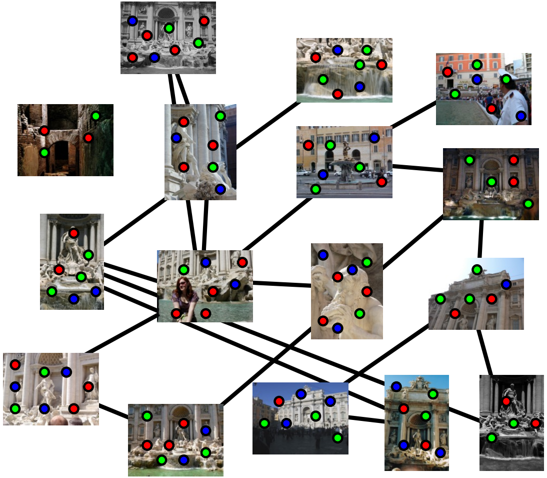
Feature description

Describe features using SIFT [Lowe, IJCV 2004]



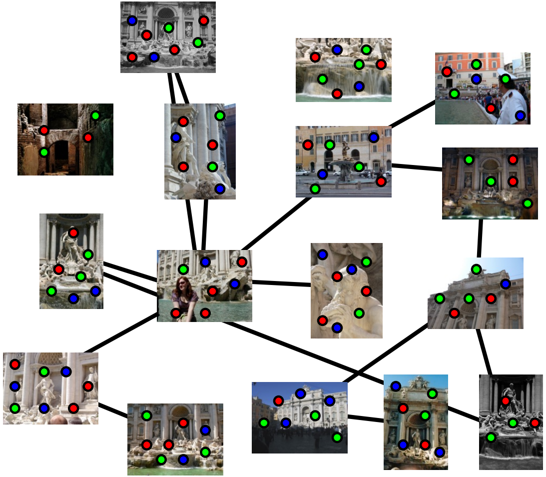
Feature matching

Match features between each pair of images



Feature matching

Refine matching using RANSAC to estimate fundamental matrix between each pair



Correspondence estimation

- Link up pairwise matches to form connected components of matches across several images

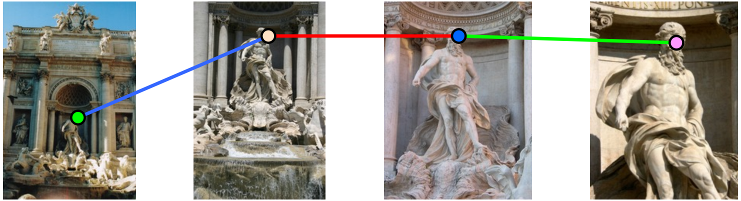
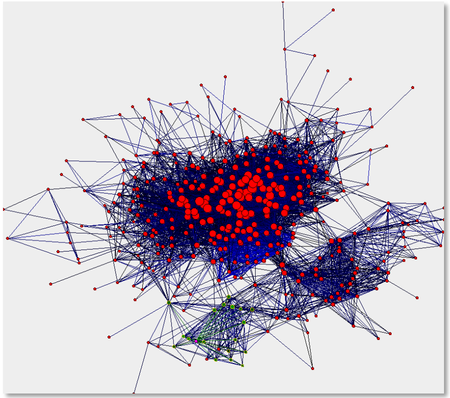


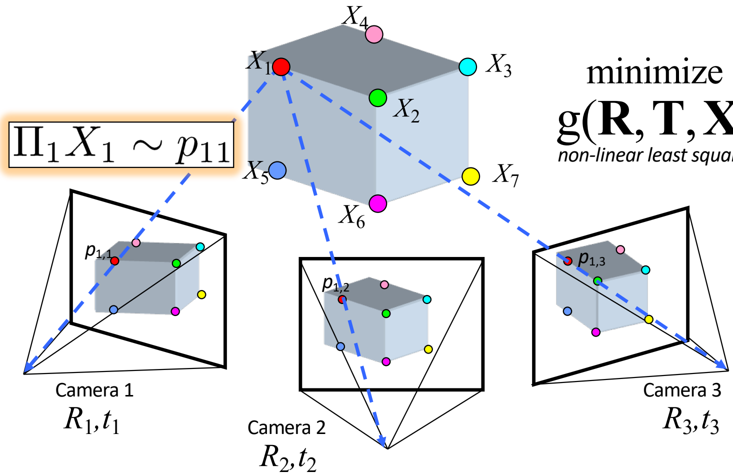
Image 1 Image 2 Image 3 Image 4

Image connectivity graph



(graph layout produced using the Graphviz toolkit: http://www.graphviz.org/)

Structure from motion



minimize

$\mathrm{g}(\mathbf{R}, \mathbf{T}, \mathbf{X})$

non-linear least squares

Global structure from motion

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \underbrace{w_{ij}}_{\substack{\textit{indicator variable}: \\ \text{is point } i \text{ visible in image } j\text{ ?}}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\textit{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\textit{observed} \\ \text{image location}}} \right\|^2$$

- Minimizing this function is called *bundle adjustment*
  - Optimized using non-linear least squares, e.g. Levenberg-Marquardt

Problem size

- What are the variables?
- How many variables per camera?
- How many variables per point?

- Trevi Fountain collection

  466 input photos

  + > 100,000 3D points

  = very large optimization problem

Doing bundle adjustment

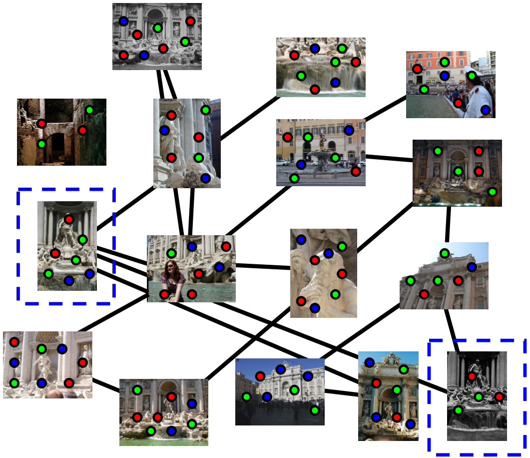
- Minimizing g is difficult
  - g is non-linear due to rotations, perspective division
  - lots of parameters: 3 for each 3D point, 6 for each camera
  - difficult to initialize
  - gauge ambiguity: error is invariant to a similarity transform (translation, rotation, uniform scale)

Doing bundle adjustment

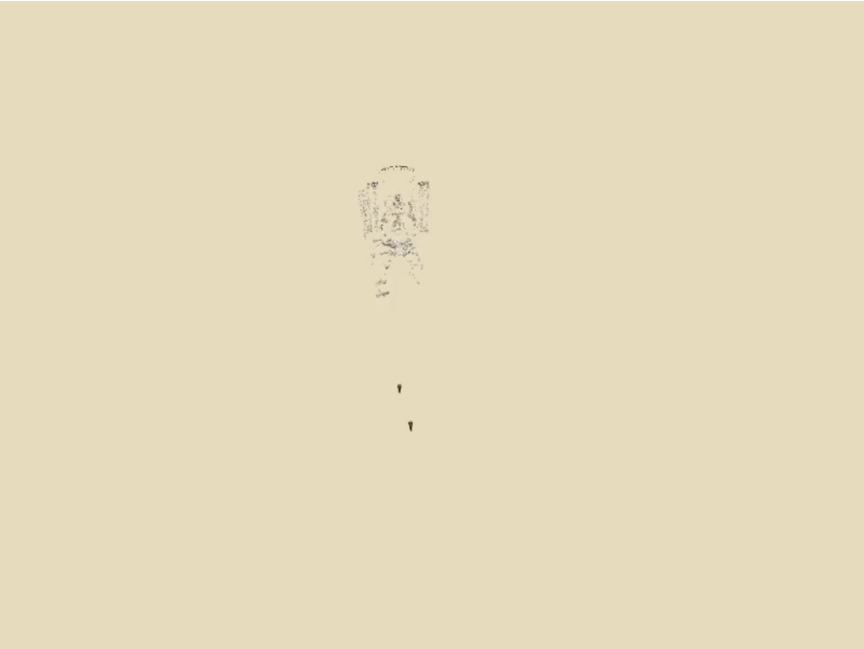
- Minimizing *g* is difficult
  - *g* is non-linear due to rotations, perspective division
  - lots of parameters: 3 for each 3D point, 6 for each camera
  - difficult to initialize
  - gauge ambiguity: error is invariant to a similarity transform (translation, rotation, uniform scale)

- Many techniques use non-linear least-squares (NLLS) optimization (*bundle adjustment*)
  - Levenberg-Marquardt is one common algorithm for NLLS
  - Lourakis, **The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm**, http://www.ics.forth.gr/~lourakis/sba/
  - http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm

Initialization: Incremental structure from motion



Incremental structure from motion



Final reconstruction

More examples

More examples

More examples

Even larger scale SfM

City-scale structure from motion

- “Building Rome in a day”

http://grail.cs.washington.edu/projects/rome/

SfM applications

- 3D modeling
- Surveying
- Robot navigation and mapmaking
- Visual effects (“Match moving”)
  - https://www.youtube.com/watch?v=RdYWp70P_kY

Applications – Photosynth

Applications – Hyperlapse

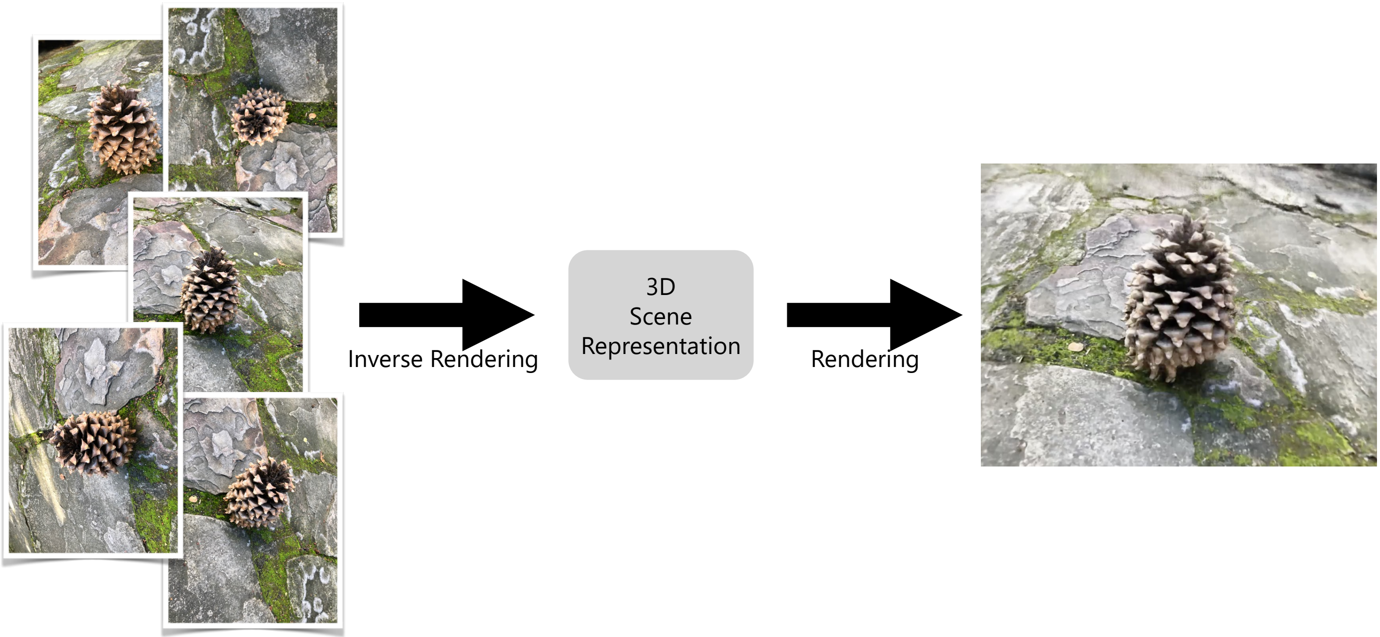
https://www.youtube.com/watch?v=SOpwHaQnRSY

Computer vision as inverse rendering

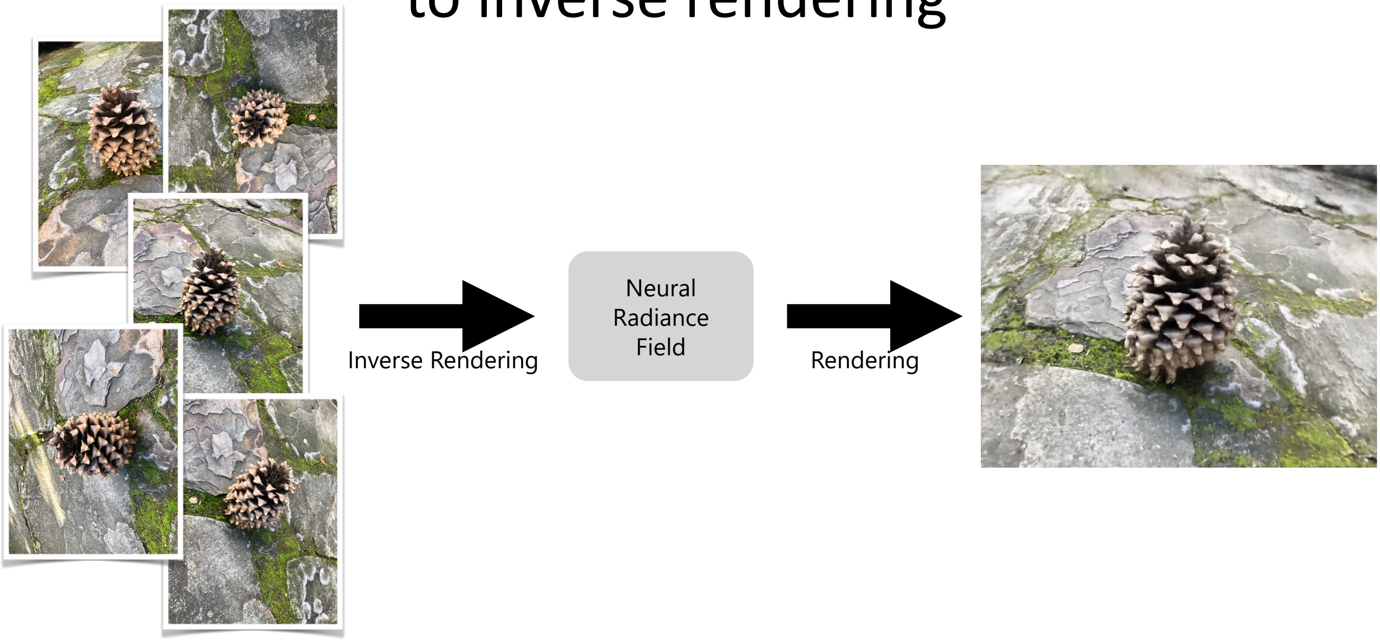
3D
Scene
Representation

Rendering

Computer vision as inverse rendering



Neural Radiance Fields (NeRF) as an approach to inverse rendering



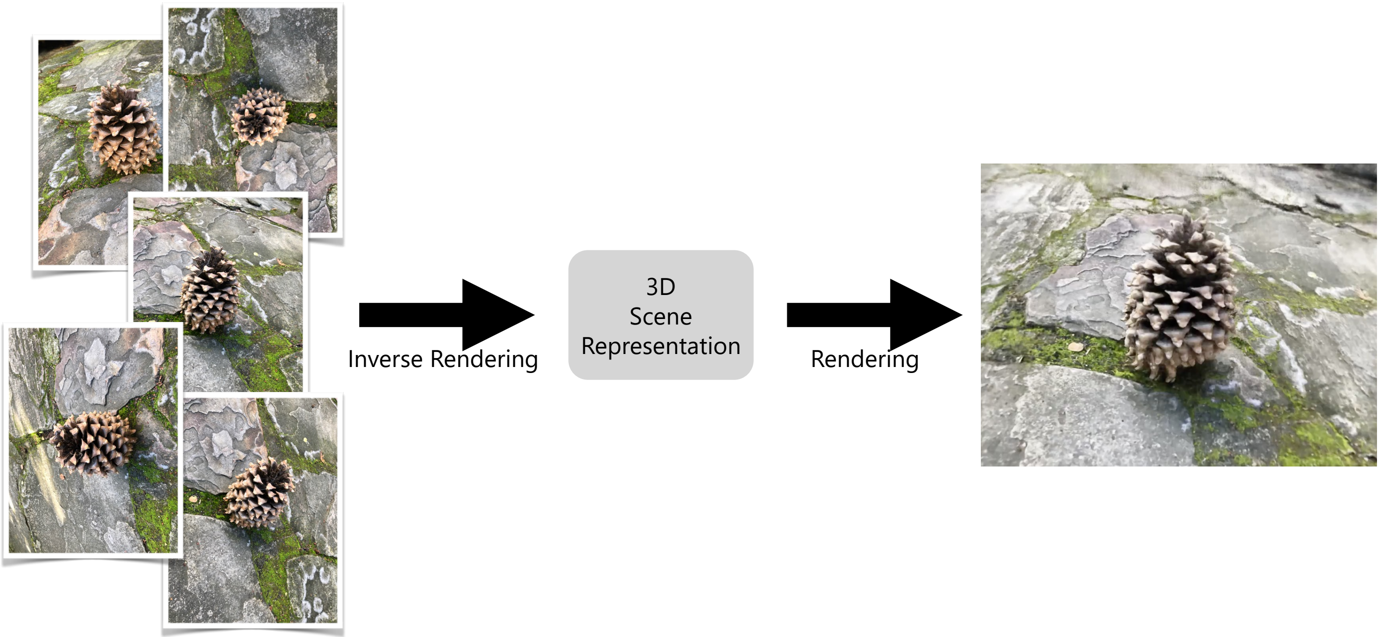
Deep learning for 3D reconstruction

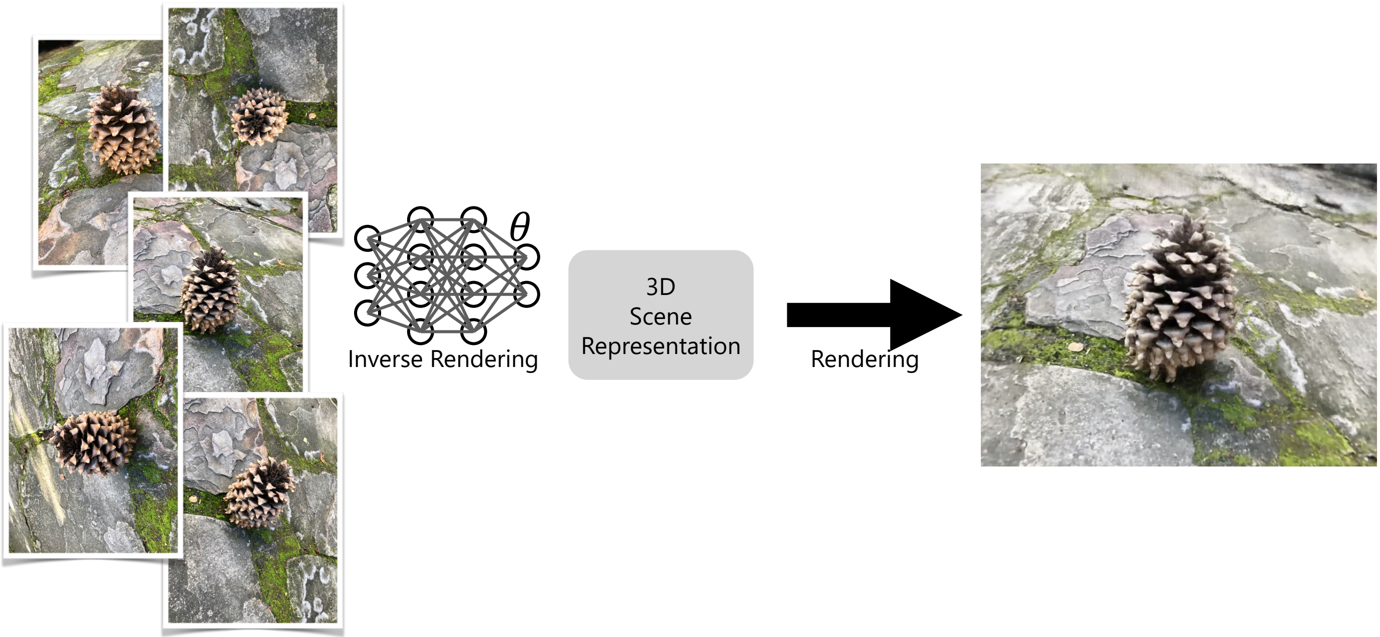
- Previously: we reconstruct geometry by running stereo or multi-view stereo on a set of images
  - "Classical" approach
- How can we leverage powerful tools of deep learning?
  - Deep neural networks
  - GPU-accelerated stochastic gradient descent

NeRF and related methods – Key ideas

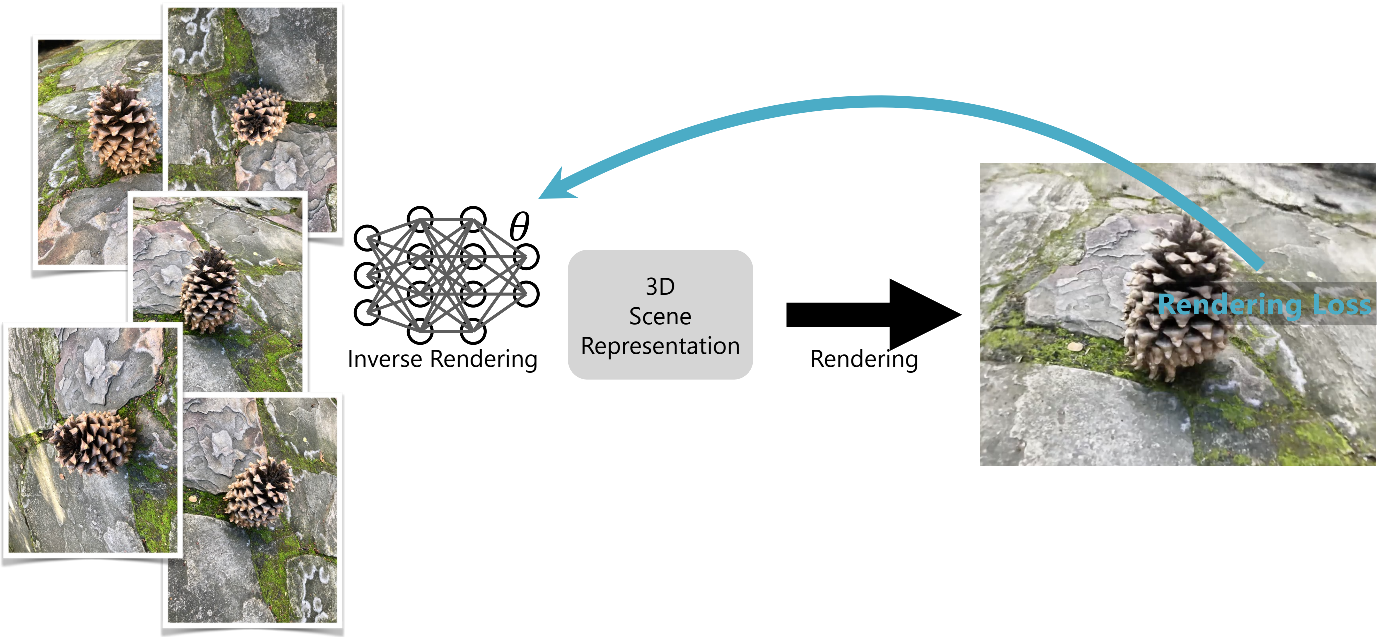
- We need to create a loss function and a scene representation that we can optimize using gradient descent to reconstruct the scene
- *Differentiable rendering*

Computer vision as inverse rendering

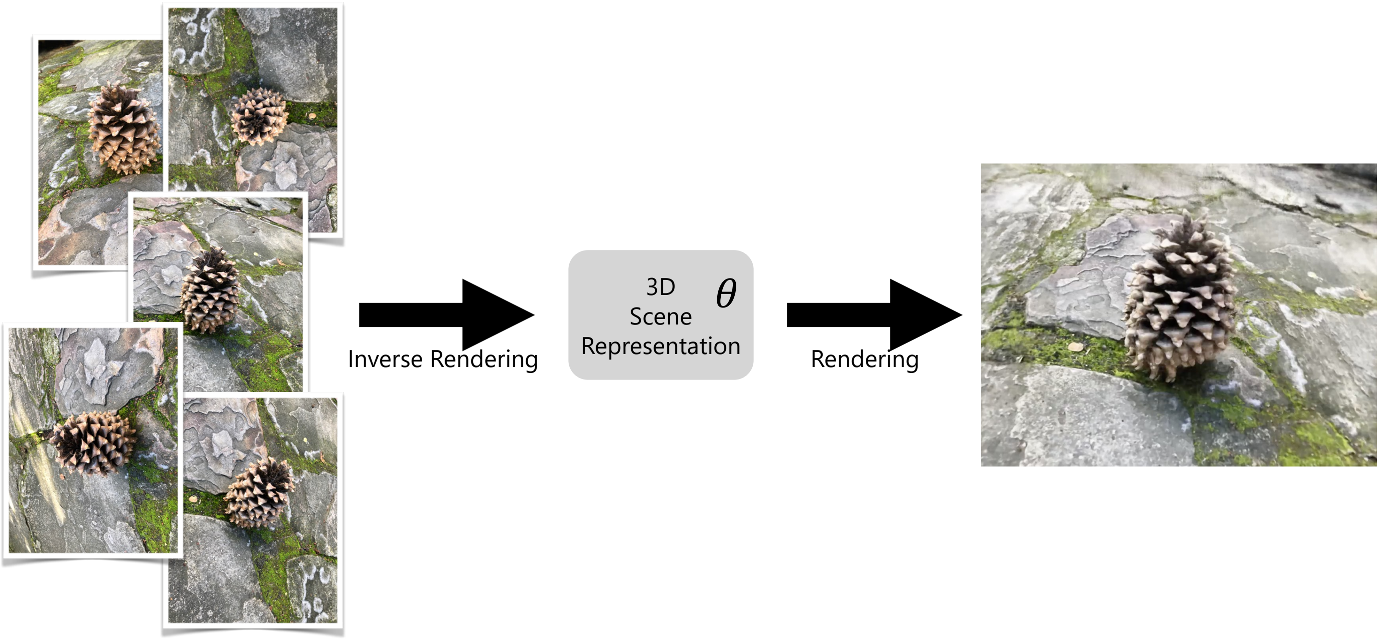


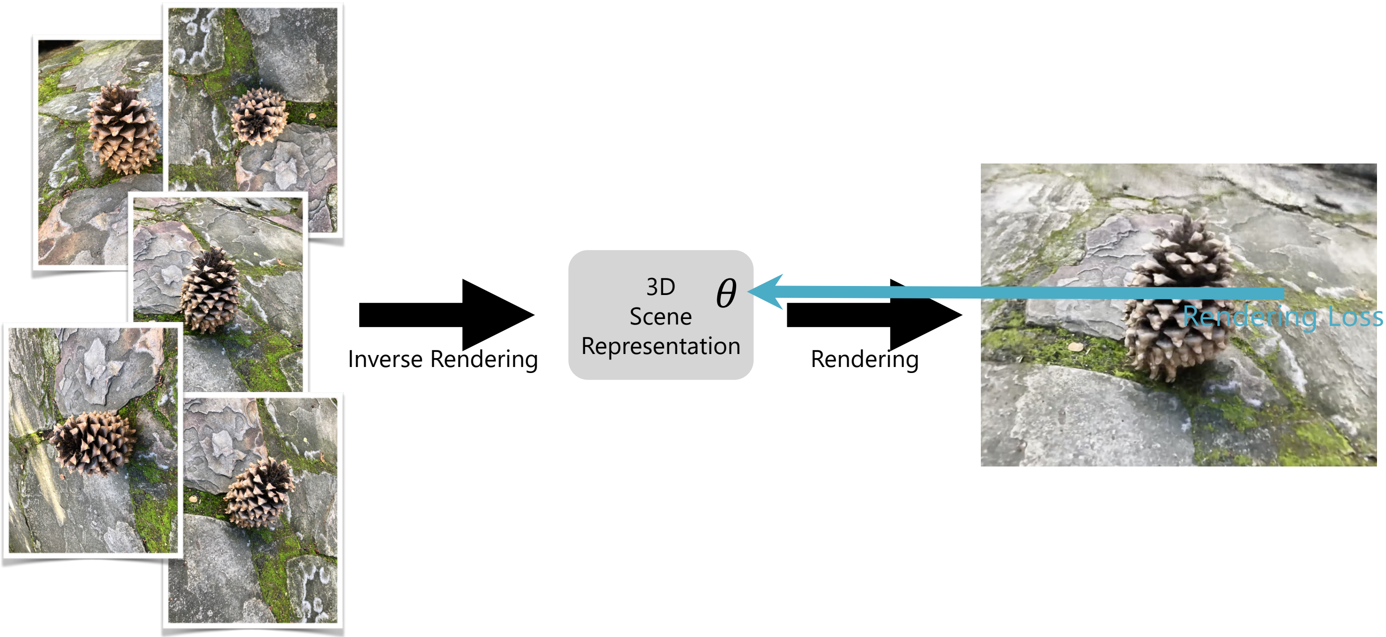


θ
Inverse Rendering
3D
Scene
Representation
Rendering



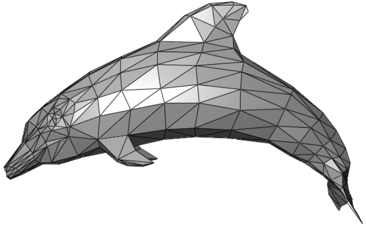
θ
Inverse Rendering
3D
Scene
Representation
Rendering
Rendering Loss


Inverse Rendering
3D
Scene
Representation
θ
Rendering


Inverse Rendering
3D
Scene
Representation
θ
Rendering
Rendering Loss

What representation to use?

- Could use triangle meshes, but hard to differentiate during rendering



NeRF == Differentiable Rendering with a **Neural Volumetric Representation**

Barron et al 2021, Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields

Neural Volumetric Rendering

Neural Volumetric Rendering

querying the radiance value
along rays through 3D space

What color?

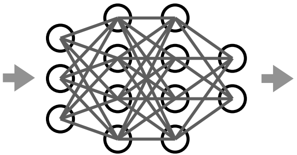
Neural **Volumetric** Rendering

continuous, differentiable
rendering model without
concrete ray/surface intersections

Neural Volumetric Rendering

using a neural network as a scene representation, rather than a voxel grid of data

(x, y, z)



Scene properties

Multi-layer Perceptron (Neural Network)

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020

Ben Mildenhall*

UC Berkeley

Pratul Srinivasan*

UC Berkeley

Matt Tancik*

UC Berkeley

Jon Barron

Google Research

Ravi Ramamoorthi

UC San Diego

Ren Ng

UC Berkeley

Optimize a NeRF model

Given a set of sparse views of an object with known camera poses

3D reconstruction viewable from any angle

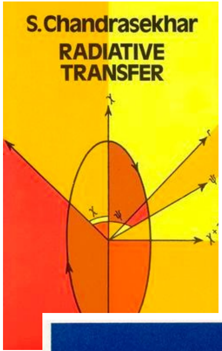
NeRF Overview

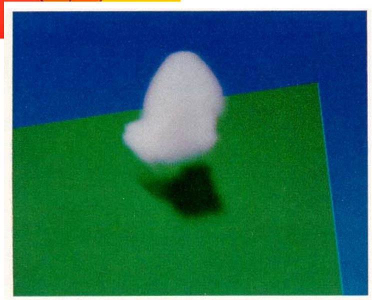
- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

NeRF Overview

- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

Traditional volumetric rendering



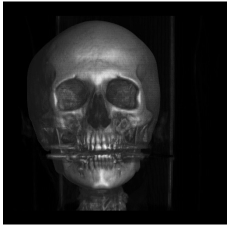


Ray tracing simulated cumulus cloud [Kajiya]

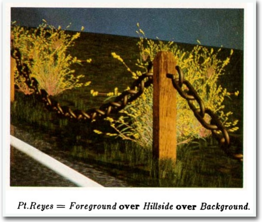
- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering

Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities

Traditional volumetric rendering



Medical data visualisation [Levoy]



Alpha compositing [Porter and Duff]

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing

Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities
Levoy 1988, Display of Surfaces from Volume Data
Max 1995, Optical Models for Direct Volume Rendering
Porter and Duff 1984, Compositing Digital Images

Traditional volumetric rendering


Physically-based Monte Carlo rendering [Novak et al]

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing
- Modern path tracers use sophisticated Monte Carlo methods to render volumetric effects

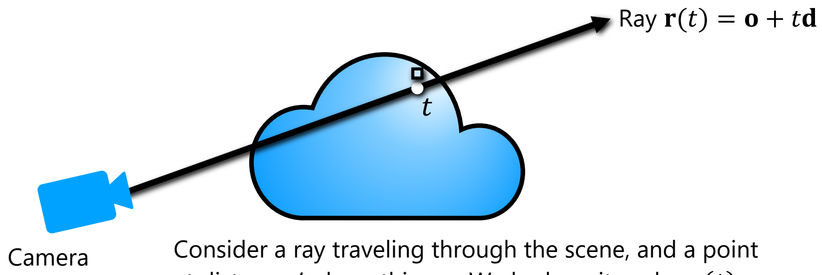
Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities
Levoy 1988, Display of Surfaces from Volume Data
Max 1995, Optical Models for Direct Volume Rendering
Porter and Duff 1984, Compositing Digital Images
Novak et al 2018, Monte Carlo methods for physically based volume rendering

Volumetric formulation for NeRF

Scene is a cloud of colored fog

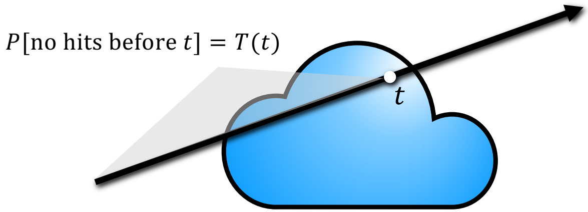
Max and Chen 2010, Local and Global Illumination in the Volume Rendering Integral

Volumetric formulation for NeRF



Consider a ray traveling through the scene, and a point at distance t along this ray. We look up its color $\mathbf{c}(t)$, and its opacity (alpha value) $\alpha(t)$

Volumetric formulation for NeRF



But t may also be blocked by earlier points along the ray. $T(t)$: probability that the ray didn't hit any particles earlier.
$T(t)$ is called "transmittance"

Volume rendering estimation: integrating color along a ray

Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

final rendered color along ray

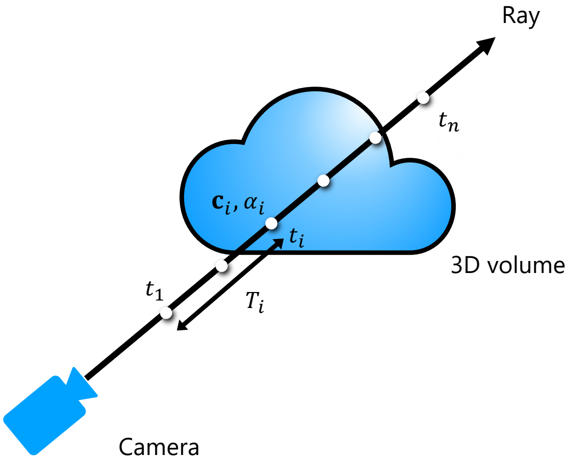
weights

colors

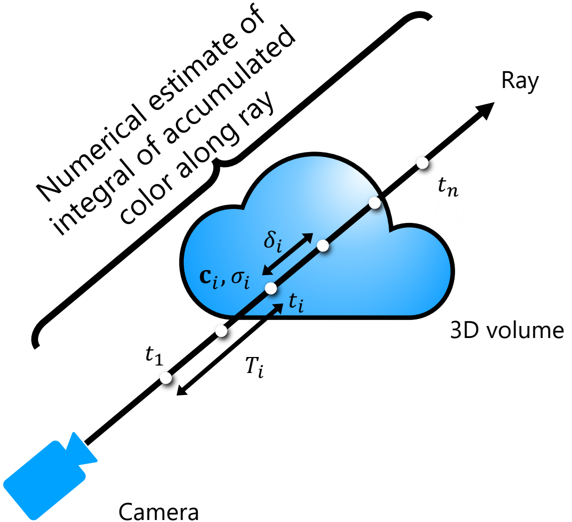
How much light is blocked earlier along ray:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

Computing the color for a set of rays through the pixels of an image yields a rendered image



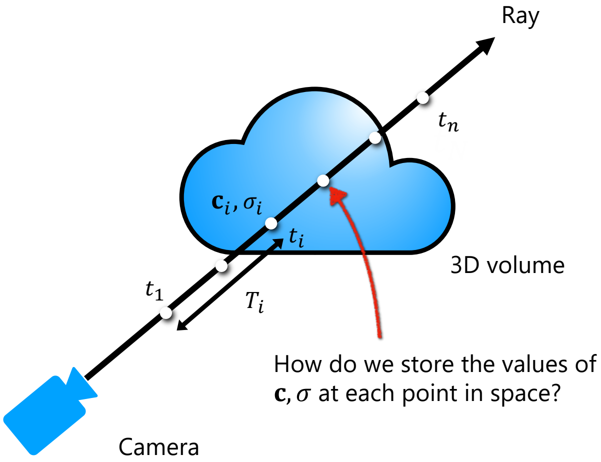
Volume rendering estimation: integrating color along a ray



$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$

Slight modification: α is not directly stored in the volume, but instead is derived from a stored volume density sigma (σ) that is multiplied by the distance between samples delta (δ):

Volume rendering estimation: integrating color along a ray



Computing the color for a set of rays through the pixels of an image yields a rendered image

NeRF Overview

- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

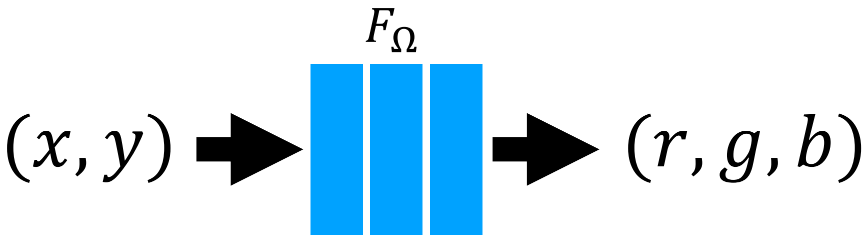
Toy problem: storing 2D image data

(x, y)

(r, g, b)

Usually we store an image as a
2D grid of RGB color values

Toy problem: storing 2D image data



What if we train a simple fully-connected network (MLP) to do this instead?

Naive approach fails!

Ground truth image

Neural network output fit
with gradient descent

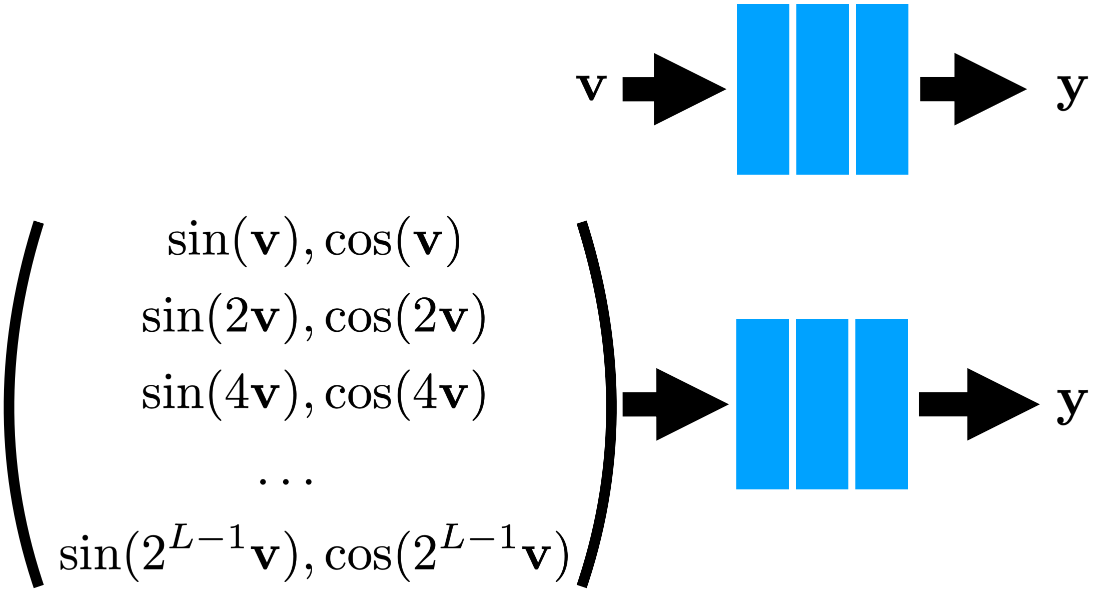
Problem:

"Standard" coordinate-based MLPs cannot represent
high frequency functions

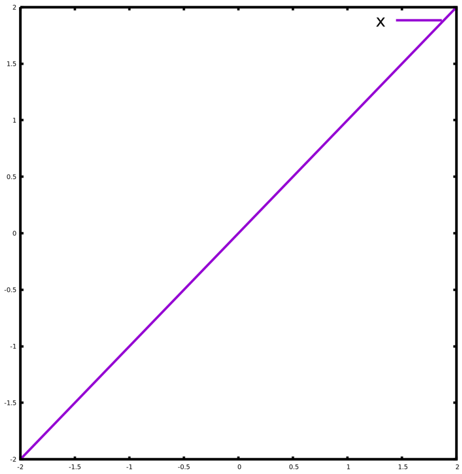
Solution:

Pass input coordinates through a high frequency mapping first

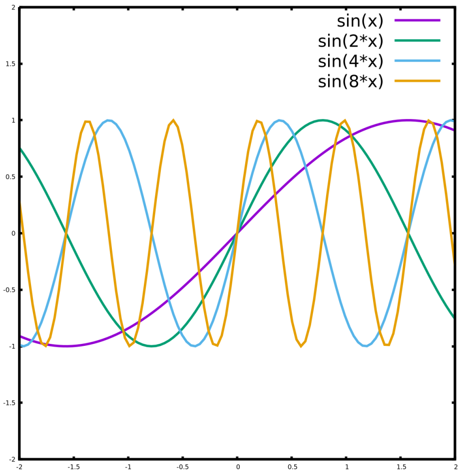
Example mapping: “positional encoding”



Positional encoding



Raw encoding of a number ***x***

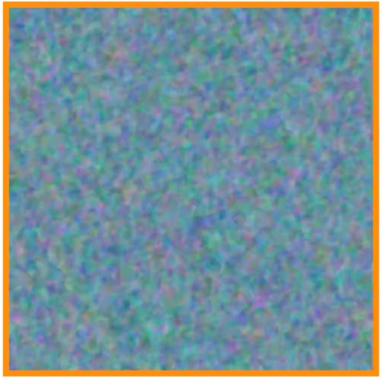


"Positional encoding" of a number ***x***

Problem solved!

Ground truth image

Neural network output without high frequency mapping



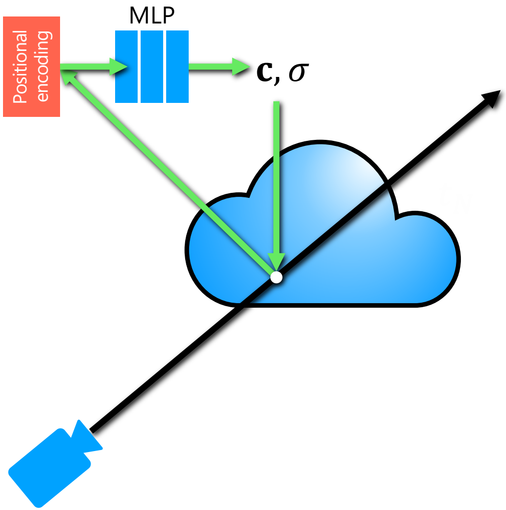
Neural network output with high frequency mapping

NeRF Overview

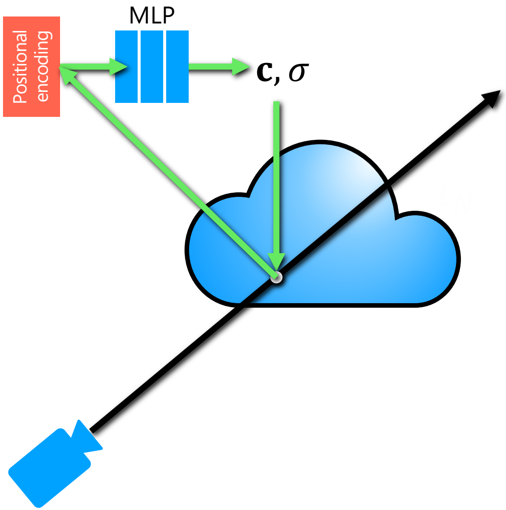
- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

NeRF = volume rendering + coordinate-based network

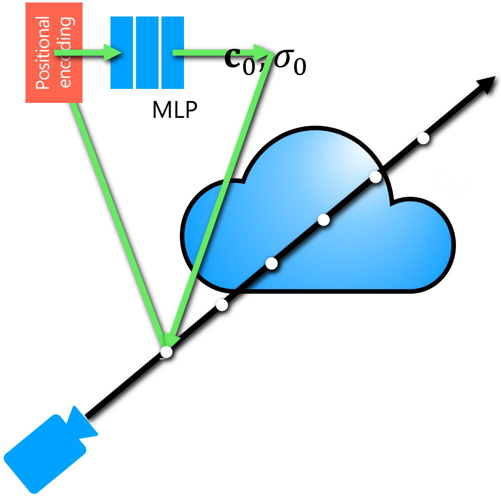
How do we store the values of $\mathbf{c}, \sigma$ at each point in space?



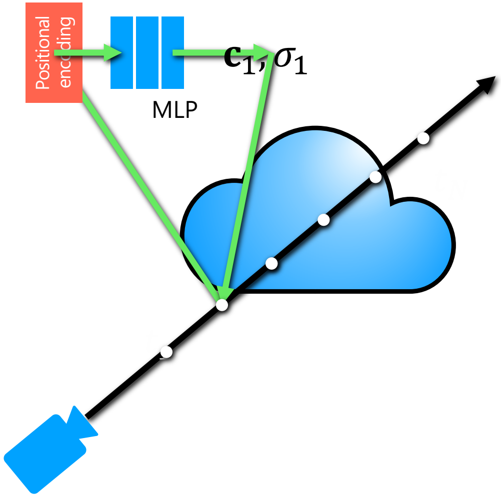
How do we store the values of $\mathbf{c}, \sigma$ at each point in space?



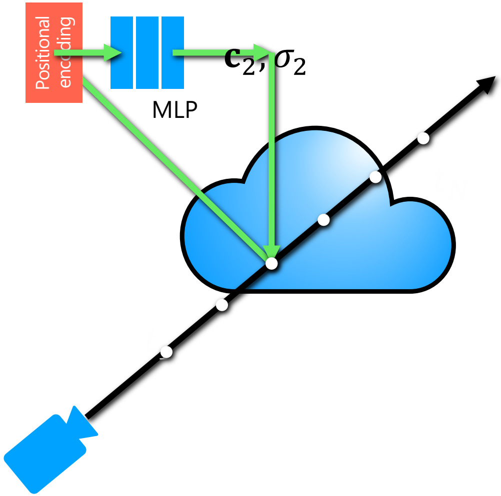
How do we store the values of $\mathbf{c}, \sigma$ at each point in space?



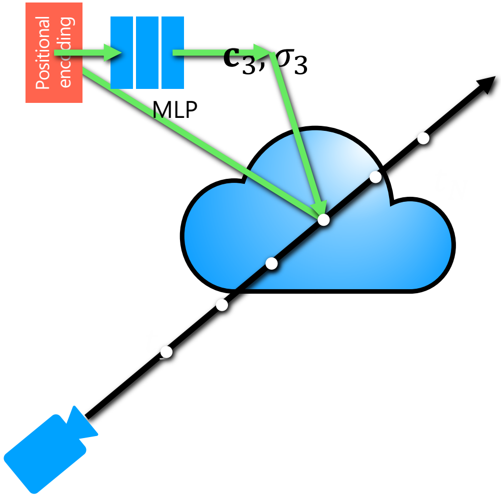
How do we store the values of $\mathbf{c}, \sigma$ at each point in space?



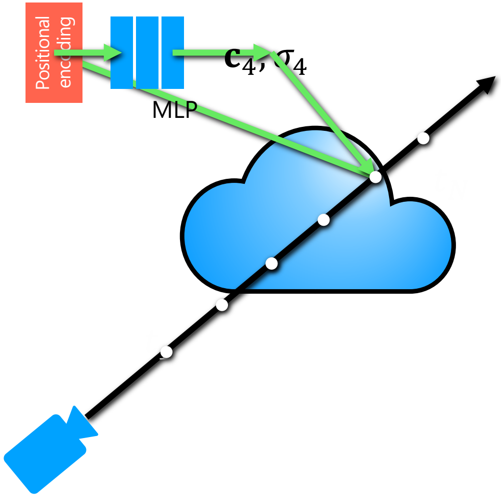
How do we store the values of $\mathbf{c}, \sigma$ at each point in space?



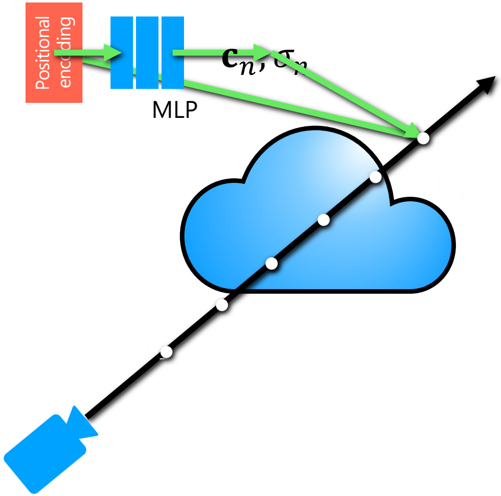
How do we store the values of $\mathbf{c}, \sigma$ at each point in space?



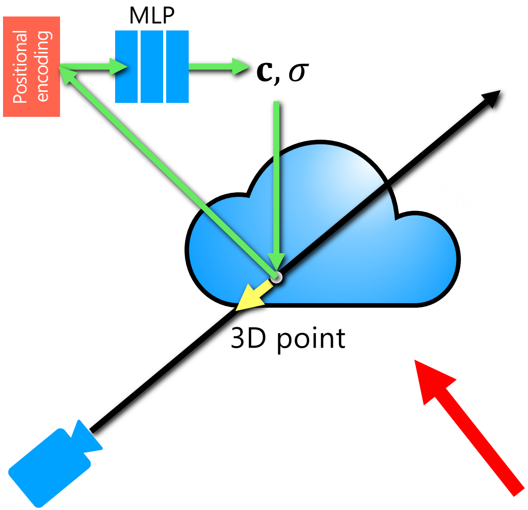
How do we store the values of $\mathbf{c}, \sigma$ at each point in space?



How do we store the values of $\mathbf{c}, \sigma$ at each point in space?

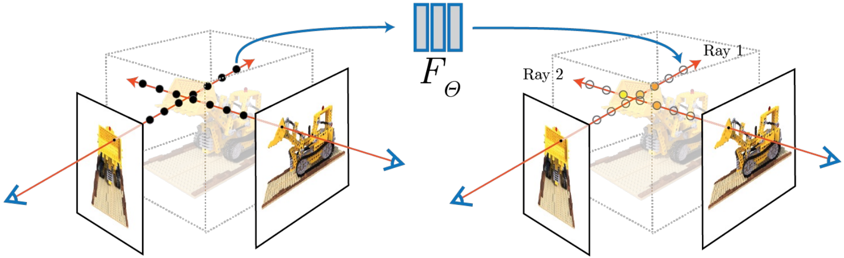


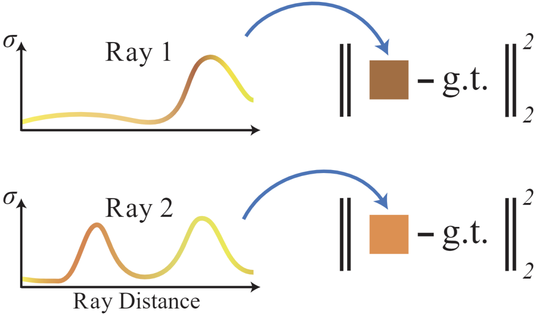
Extension: view-dependent field



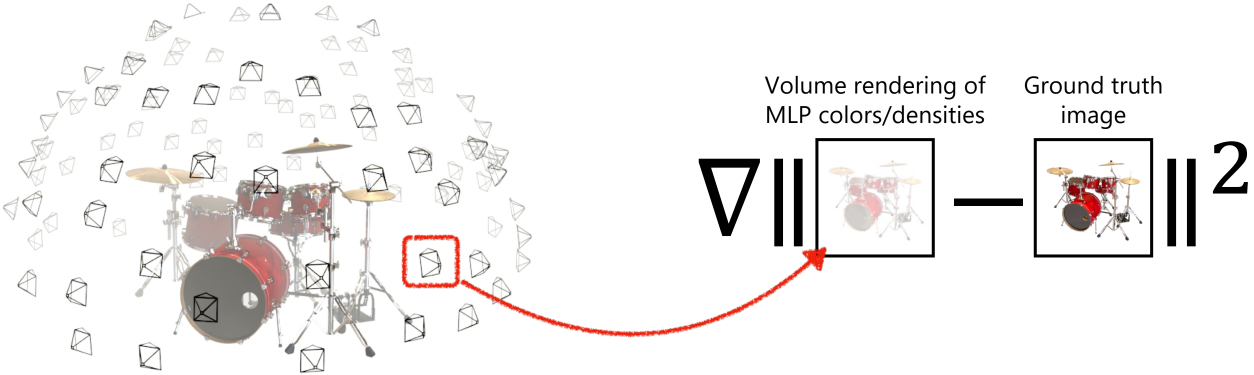
Include the ray direction in the input to the MLP → allows for capturing and rendering view-dependent effects (e.g., shiny surfaces)

Putting it all together





Train network using gradient descent to reproduce all input views of scene



Results

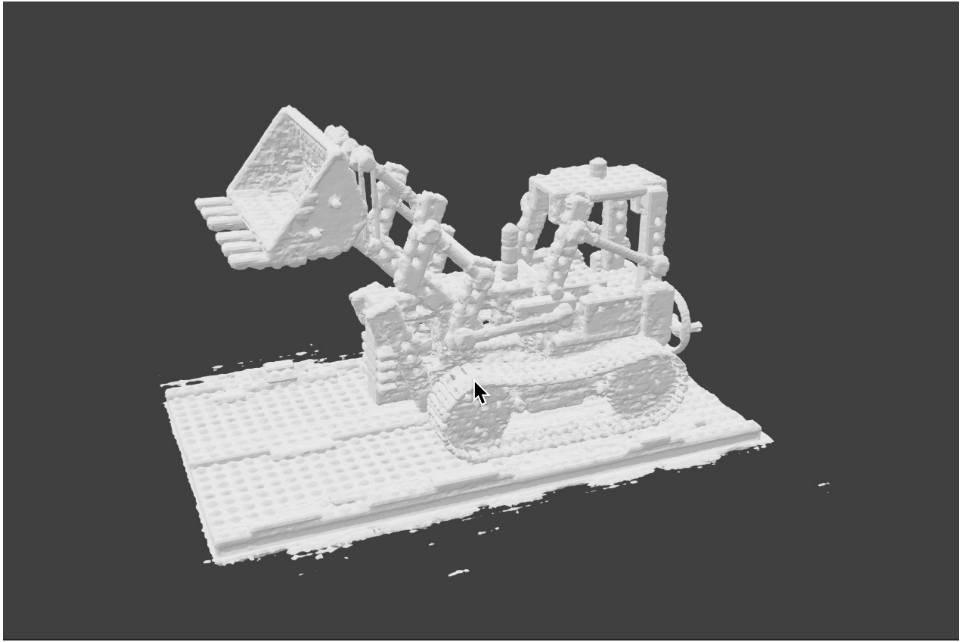
NeRF encodes convincing view-dependent effects using directional dependence

NeRF encodes convincing view-dependent effects using directional dependence

NeRF encodes detailed scene geometry with occlusion effects

NeRF encodes detailed scene geometry with occlusion effects

NeRF encodes detailed scene geometry



Summary

- Represent the scene as volumetric colored “fog”
- Store the fog color and density at each point as an MLP mapping 3D position (x, y, z) to color c and density σ
- Render image by shooting a ray through the fog for each pixel
- Optimize MLP parameters by rendering to a set of known viewpoints and comparing to ground truth images

NeRF in the Wild (NeRF-W)

Brandenburg Gate

Sacre Coeur

Trevi Fountain

Martin-Brualla*, Radwan*, Sajjadi*, Barron, Dosovitskiy, Duckworth.
NeRF in the Wild. CVPR 2021.
https://www.youtube.com/watch?v=mRAKVQj5LRA

Next Time

optical flow & tracking!

References

Basic reading:
- FCV 44-45
- Szeliski textbook, Chapter 7.
- Hartley and Zisserman, Chapter 18.