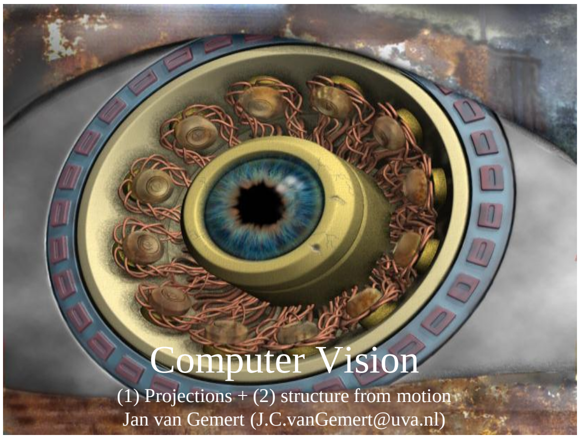


Computer Vision

(1) Projections + (2) structure from motion
Jan van Gemert (J.C.vanGemert@uva.nl)

2

Projection

– Szeliski chapter 2

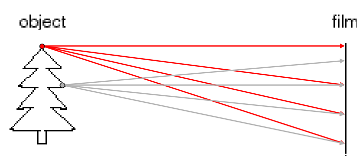
3

Projection

– Szeliski chapter 2

4

Let's design a camera

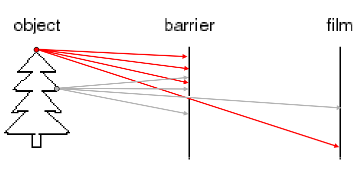


- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?

Slide by Steve Seitz

5

Pinhole camera

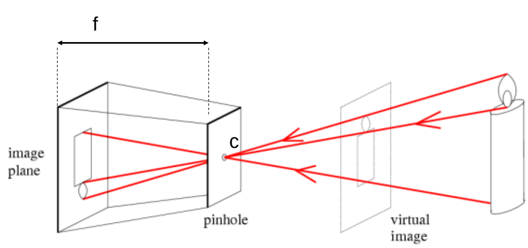


- Add a barrier to block off most of the rays
  - This reduces blurring
  - The opening is known as the **aperture**

Slide by Steve Seitz

6

Pinhole camera



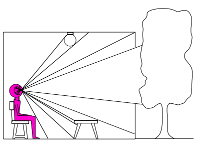
f = focal length
c = center of the camera

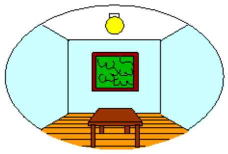
Figure from Forsyth

7

Dimensionality Reduction Machine (3D to 2D)

3D world *2D image*





Point of observation

What have we lost?

- Angles
- Distances (lengths)

Slide by A. Efros
Figures © Stephen E. Palmer, 2002

8

Projection properties

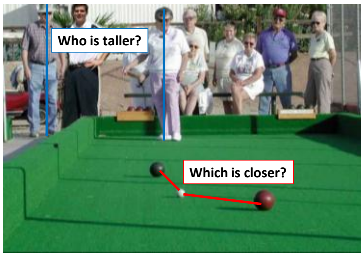
- Many-to-one: any points along same *visual ray* map to same point in image
- Points → points
  - But projection of points on focal plane is undefined
- Lines → lines (collinearity is preserved)
  - But line through focal point (visual ray) projects to a point
- Planes → planes (or half-planes)
  - But plane through focal point projects to line

9

Projective Geometry

What is lost?

- Length



10

Length is not preserved

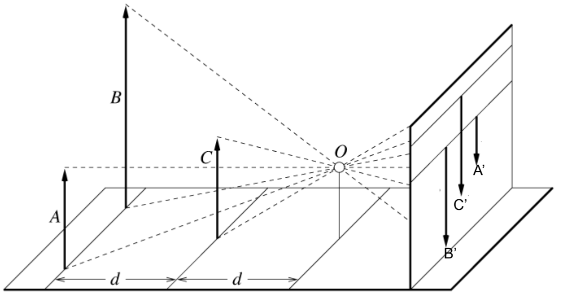


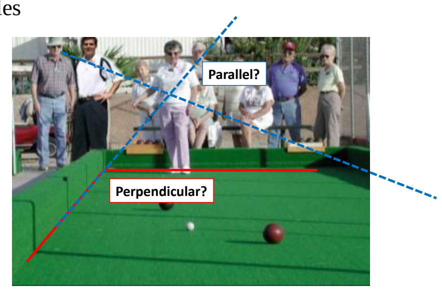
Figure by David Forsyth

11

Projective Geometry

What is lost?

- Length
- Angles

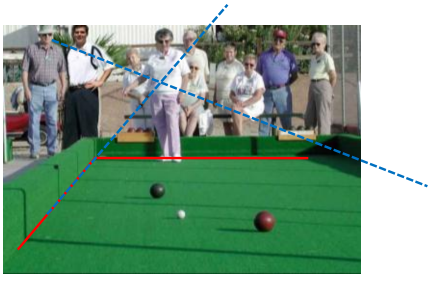


12

Projective Geometry

What is preserved?

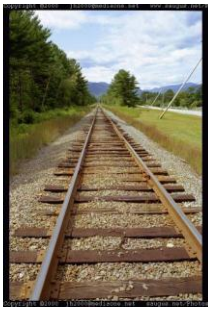
- Straight lines are still straight



13

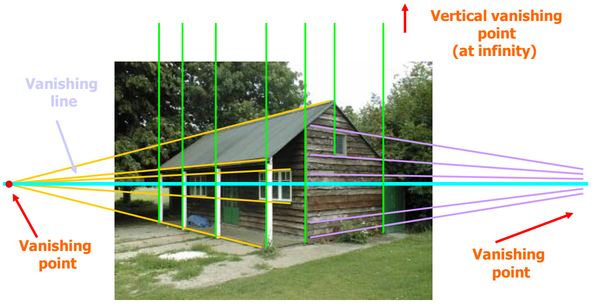
Vanishing points and lines

Parallel lines in the world intersect in the image at a "vanishing point"



14

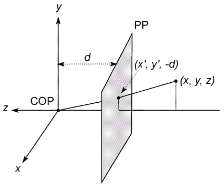
Vanishing points and lines



Slide from Efros, Photo from Criminisi

15

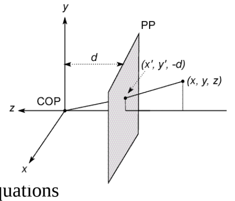
Modeling projection



- The coordinate system
  - We will use the pin-hole model as an approximation
  - Put the optical center (**C**enter **O**f **P**rojection) at the origin
  - Put the image plane (**P**rojection **P**lane) *in front* of the COP
    - Why?
  - The camera looks down the *negative* z axis

16

Modeling projection



- Projection equations
  - Compute intersection with PP of ray from (x,y,z) to COP
  - Derived using similar triangles

$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z},\ -d)$$

- We get the projection by throwing out the z coordinate:

$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z})$$

17

Homogeneous coordinates

- Is this a linear transformation?

no—division by z is nonlinear

Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous image coordinates

homogeneous scene coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

18

Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates:

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

divide by the third coordinate

19

Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates:

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

divide by the third coordinate

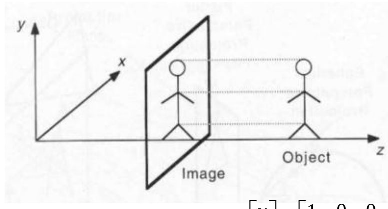
In practice: split into lots of different coordinate transformations…

$$\begin{pmatrix} \text{2D} \\ \text{point} \\ \text{(3x1)} \end{pmatrix} = \begin{pmatrix} \text{Camera to} \\ \text{pixel coord.} \\ \text{trans. matrix} \\ \text{(3x3)} \end{pmatrix} \begin{pmatrix} \text{Perspective} \\ \text{projection matrix} \\ \text{(3x4)} \end{pmatrix} \begin{pmatrix} \text{World to} \\ \text{camera coord.} \\ \text{trans. matrix} \\ \text{(4x4)} \end{pmatrix} \begin{pmatrix} \text{3D} \\ \text{point} \\ \text{(4x1)} \end{pmatrix}$$

20

Orthographic Projection

- Special case of perspective projection
  - Distance from the COP to the image plane is ‘infinite’



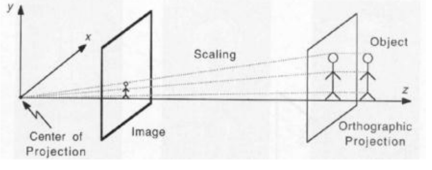
  - Also called “parallel projection”
  - What’s the projection matrix?

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

21

Scaled Orthographic Projection

- Special case of perspective projection
  - Object dimensions are small compared to distance to camera

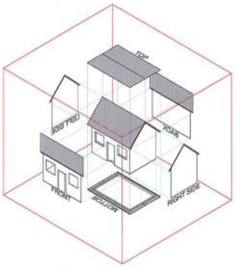


  - Also called “weak perspective”
  - What’s the projection matrix?

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

22

Example of a scaled orthographic projection



23

Questions?

24

Camera Obscura

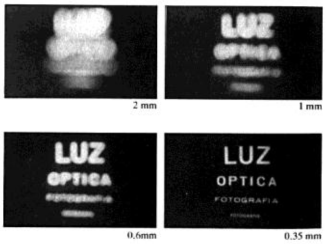
Gemma Frisius, 1558

- Basic principle known to Mozi (470-390 BCE), Aristotle (384-322 BCE)
- Drawing aid for artists: described by Leonardo da Vinci (1452-1519)

Source: A. Efros

25

Shrinking the aperture

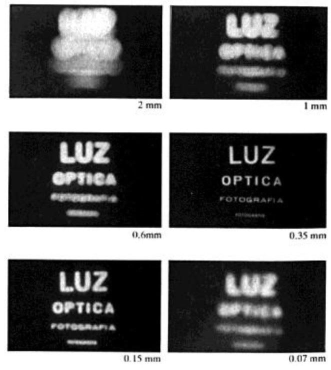


- Why not make the aperture as small as possible?
  - Less light gets through
  - Diffraction effects…

Slide by Steve Seitz

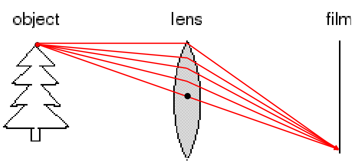
26

Shrinking the aperture



27

Adding a lens

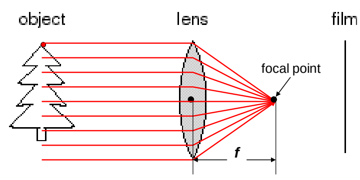


- A lens focuses light onto the film
  - Rays passing through the center are not deviated

Slide by Steve Seitz

28

Adding a lens

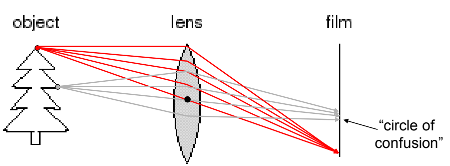


- A lens focuses light onto the film
  - Rays passing through the center are not deviated
  - All parallel rays converge to one point on a plane located at the *focal length f*

Slide by Steve Seitz

29

Adding a lens



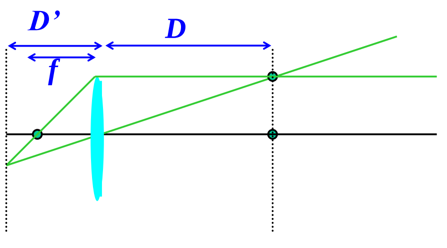
- A lens focuses light onto the film
  - There is a specific distance at which objects are "in focus"
    - other points project to a "circle of confusion" in the image

Slide by Steve Seitz

30

Thin lens formula

Relation between the focal length (f), the distance of the object from the camera (D), and the distance at which the object will be in focus (D')

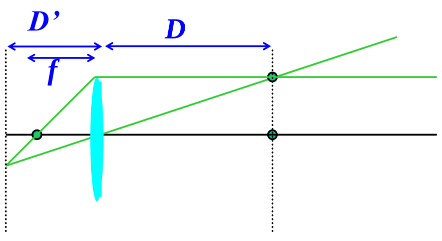


Frédo Durand's slide

31

Thin lens formula

Similar triangles everywhere!

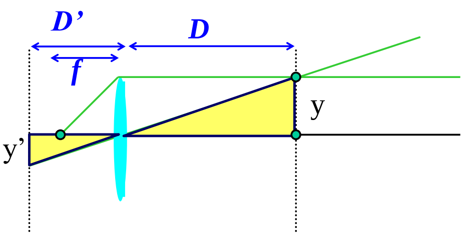


Frédo Durand's slide

32

Thin lens formula

$$y'/y = D'/D$$



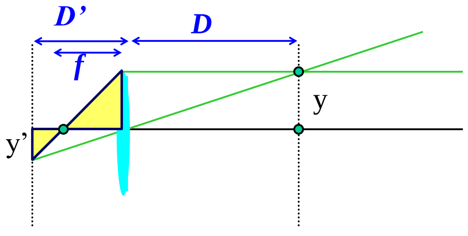
Frédo Durand's slide

33

Thin lens formula

$$y'/y = D'/D$$

$$y'/y = (D'-f)/f$$



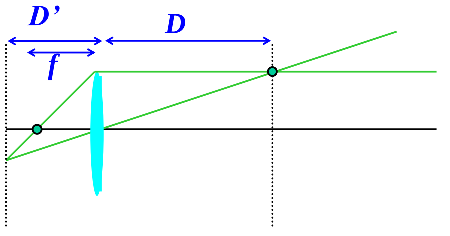
Frédo Durand's slide

34

Thin lens formula

$$\boxed{\frac{1}{D'} + \frac{1}{D} = \frac{1}{f}}$$

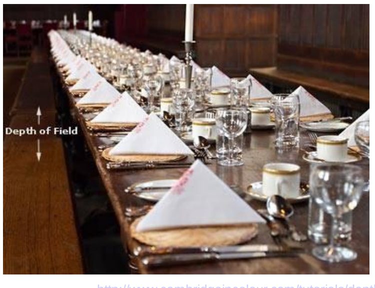
Any point satisfying the thin lens equation is in focus.



Frédo Durand's slide

35

Depth of Field



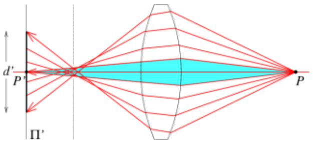
http://www.cambridgeincolour.com/tutorials/depth-of-field.htm

Slide by A. Efros

36

Lens flaws: Spherical aberration

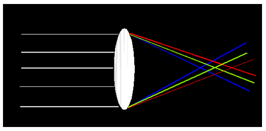
- Spherical lenses don't focus light perfectly
- Rays farther from the optical axis focus closer



37

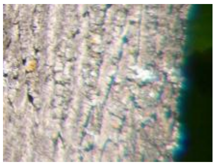
Lens Flaws: Chromatic Aberration

- Lens has different refractive indices for different wavelengths: causes color fringing



Near Lens Center

Near Lens Outer Edge



38

Questions?

39

Basic 2D Transformations

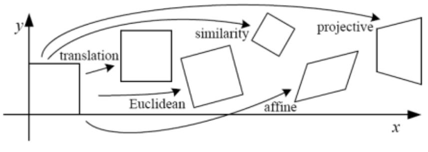


Figure 2.4: *Basic set of 2D planar transformations*

40

Basic 2D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times 3}$	8	straight lines	

41

2D Affine Transformations

Basic 2D transformations as 3x3 matrices

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

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh_x} & 0 \\ \boldsymbol{sh_y} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

Source: Alyosha Efros

42

2D Affine Transformations

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

Affine transformations are combinations of …

- Linear transformations, and
- Translations

Parallel lines remain parallel

Grauman

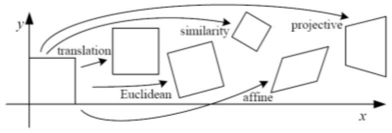
43

Projective Transformations

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

Projective transformations:
- Affine transformations, and
- Projective warps

Parallel lines do not necessarily remain parallel

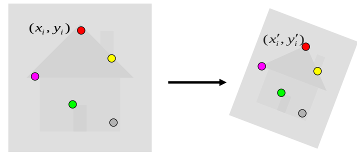


Grauman

45

Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?



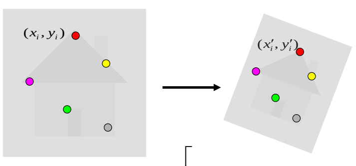
$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Grauman

46

Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \qquad \begin{bmatrix} \quad \\ \\ \\ \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \quad \\ \\ \\ \end{bmatrix}$$

47

Fitting an affine transformation

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?
- Linear system with six unknowns. Each match gives us 2 linearly independent equations: need at least 3 to solve

Grauman

48

Panoramas

Obtain a wider angle view by combining multiple images.

Grauman

49

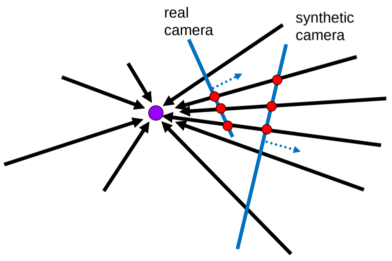
How to stitch together a panorama?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - (If there are more images, repeat)
- …but **wait**, why should this work at all?
  - What about the 3D geometry of the scene?

Source: Steve Seitz

50

Panoramas: generating synthetic views

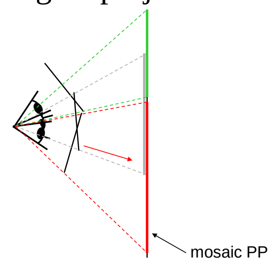


Can generate any synthetic camera view
as long as it has **the same center of projection**!

Source: Alyosha Efros

51

Image reprojection



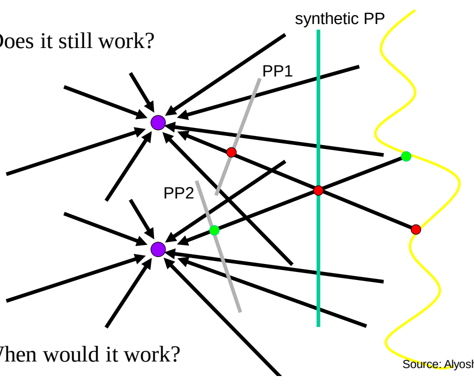
The mosaic has a natural interpretation in 3D
- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*

Source: Steve Seitz

52

changing camera center

Does it still work?

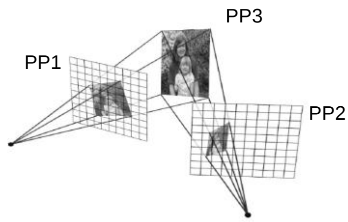


When would it work?

Source: Alyosha Efros

53

Planar scene (or far away)



PP3 is a projection plane of both centers of projection, so we are OK!

This is how big aerial photographs are made

Source: Alyosha Efros

54

Grauman

56

Homography (projective transform)

How to relate two images from the same camera center?

- how to map a pixel from PP1 to PP2?

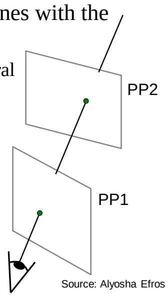
A 2D **image warp** from one image to another.

A mapping between any two Projective Planes with the same center of projection

- rectangle should map to arbitrary quadrilateral
- parallel lines aren't preserved
- but must preserve straight lines

called **Homography**

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

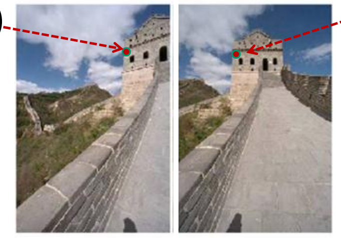


Source: Alyosha Efros

57

Homography

(x, y)



$\left(\frac{wx'}{w}, \frac{wy'}{w}\right) = (x', y')$

To **apply** a given homography **H**

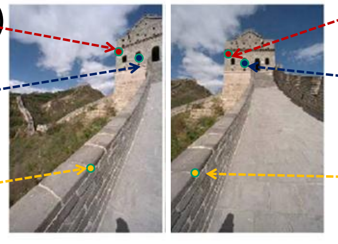
- Compute **p'** = **Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates

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

58

Homography

(x_1, y_1) (x_2, y_2) $\vdots$ (x_n, y_n)



(x'_1, y'_1) (x'_2, y'_2) $\vdots$ (x'_n, y'_n)

To **compute** the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of **H** are the unknowns…

Grauman

59

Solving for homographies

p' = Hp

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

Can set scale factor i=1. So, there are 8 unknowns.

Set up a system of linear equations:

Ah = b

where vector of unknowns h = $[a,b,c,d,e,f,g,h]^T$

Need at least 8 eqs, but the more the better…

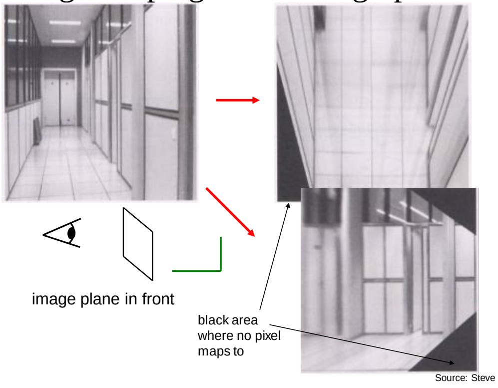
Solve for h. If overconstrained, solve using least-squares:

$$\min \|Ah - b\|^2$$

Grauman

60

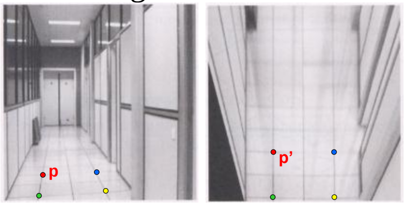
Image warping with homographies



Source: Steve Seitz

61

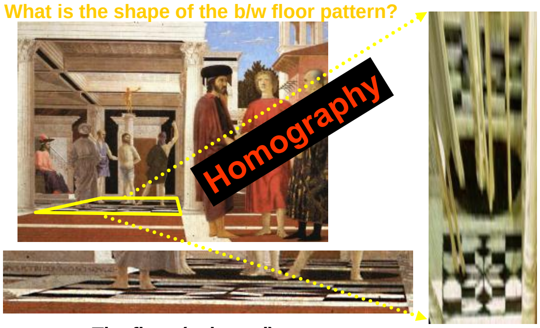
Image rectification



62

Analysing patterns and shapes

What is the shape of the b/w floor pattern?



The floor (enlarged)

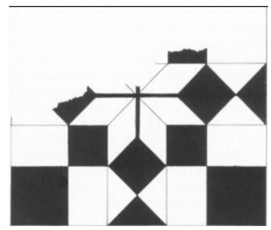
Automatically rectified floor

Slide: Criminisi

63

Analysing patterns and shapes

Automatic rectification



From Martin Kemp *The Science of Art*
(manual reconstruction)

Slide: Criminisi

64

Analysing patterns and shapes

St. Lucy Altarpiece, **D. Veneziano**

What is the (complicated) shape of the floor pattern?

Automatically rectified floor

Slide: Criminisi

65

Analysing patterns and shapes

Automatic rectification

From Martin Kemp, *The Science of Art*
(manual reconstruction)

Slide from Criminisi

66

Questions?

67

Recap: How to stitch together a panorama?

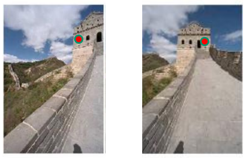
- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - (If there are more images, repeat)

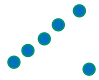
Source: Steve Seitz

68

Outliers

- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn't belong to the line we are fitting.



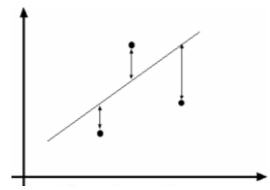


Grauman

69

Example: least squares line fitting

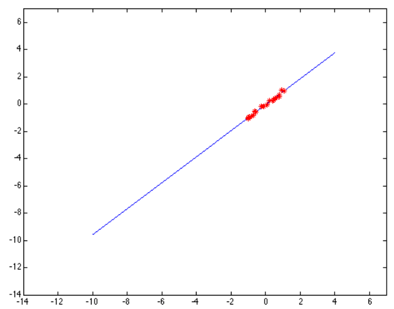
- Assuming all the points that belong to a particular line are known



Grauman

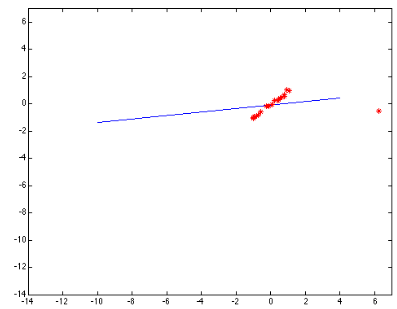
70

Outliers affect least squares fit



71

Outliers affect least squares fit



72

RANSAC

- RANdom Sample Consensus

- Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use those only.

- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

Fischler & Bolles in '81.

73

RANSAC

- RANSAC loop:

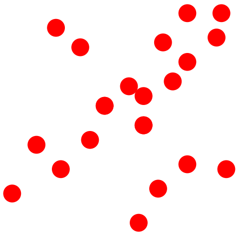
1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

74
RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



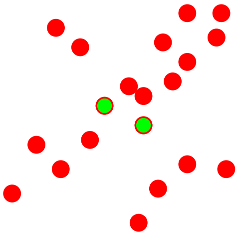
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

75
RANSAC

Line fitting example



Algorithm:

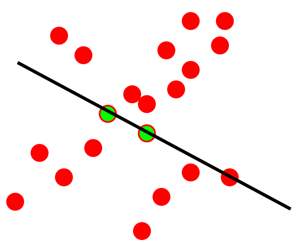
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

76
RANSAC

Line fitting example



Algorithm:

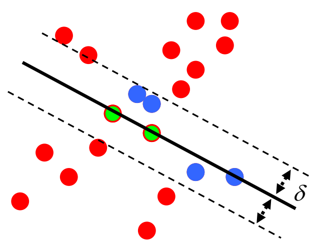
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

77
RANSAC

Line fitting example

$N_I = 6$

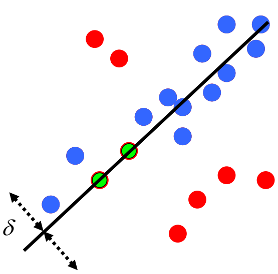


Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

78
RANSAC



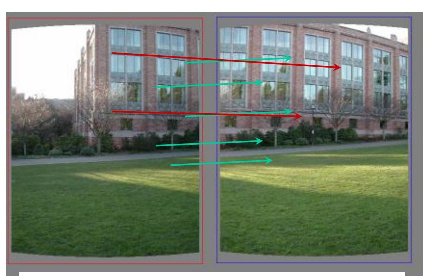
$N_I = 14$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

79
RANSAC example: Translation

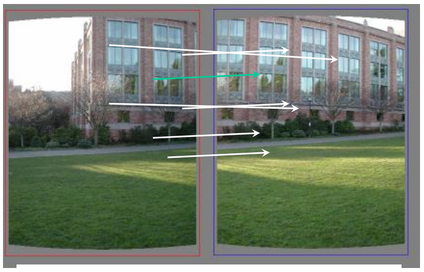


Putative matches

Source: Rick Szeliski

80

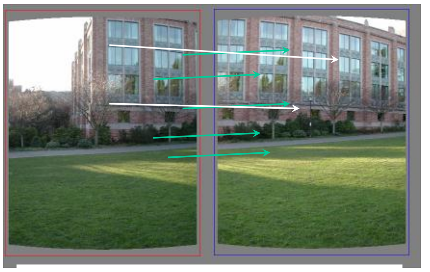
RANSAC example: Translation



Select *one* match, count *inliers*

81

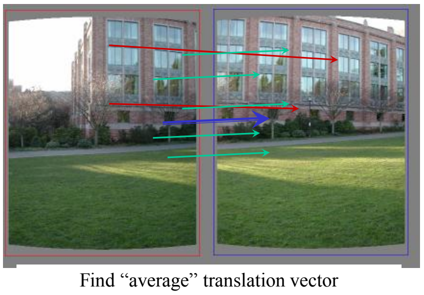
RANSAC example: Translation



Select *one* match, count *inliers*

82

RANSAC example: Translation



Find “average” translation vector

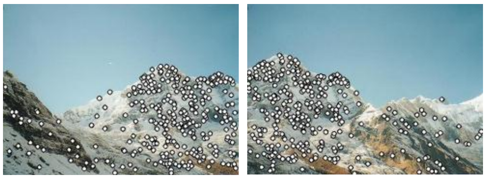
83

Feature-based alignment outline

Source: L. Lazebnik

84

Feature-based alignment outline



- Extract features

Source: L. Lazebnik

85

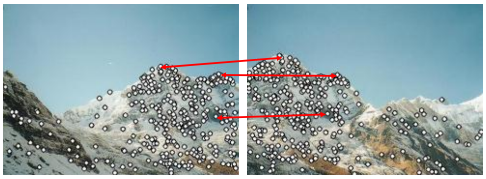
Feature-based alignment outline

- Extract features
- Compute *putative matches*

Source: L. Lazebnik

86

Feature-based alignment outline

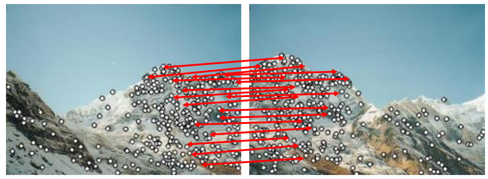


- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)

Source: L. Lazebnik

87

Feature-based alignment outline



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

88

Feature-based alignment outline

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

How well does this work?

Test on 100s of examples…

How well does this work?

Test on 100s of examples…

…still too many failures (5-10%)
for consumer application

91

Matching Mistakes: False Positive

92

Matching Mistakes: False Positive

93

Matching Mistake: False Negative

- Moving objects: large areas of disagreement (ghosting)

94

Matching Mistakes

- Accidental alignment
  - repeated / similar regions
- Failed alignments
  - moving objects / parallax
  - low overlap
  - “feature-less” regions (more variety?)
- No 100% reliable algorithm?

95

How can we fix these?

- Tune the feature detector
- Tune the feature matcher (cost metric)
- Tune the RANSAC stage (motion model)
- Tune the verification stage
- Use “higher-level” knowledge
  - e.g., typical camera motions

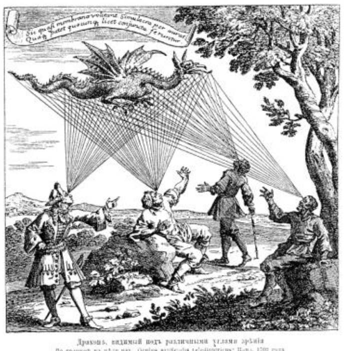
- → Sounds like a big “learning” problem
  - Need a large training/test data set (panoramas)

96

Questions?

97

Structure from motion



98

Multiple-view geometry questions

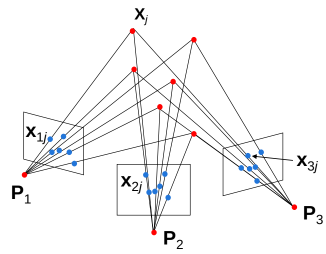
- **Scene geometry (structure):** Given 2D point matches in two or more images, where are the corresponding points in 3D?
- **Correspondence (stereo matching):** Given a point in just one image, how does it constrain the position of the corresponding point in another image?
- **Camera geometry (motion):** Given a set of corresponding points in two or more images, what are the camera matrices for these views?

Slide: S. Lazebnik

99

Structure from motion

- Given: m images of n fixed 3D points
  - $\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j,\ i = 1, \dots, m, \quad j = 1, \dots, n$
- Problem: estimate m projection matrices $\mathbf{P}_i$ and n 3D points $\mathbf{X}_j$ from the mn correspondences $\mathbf{x}_{ij}$



Slide: S. Lazebnik

100

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:

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

It is impossible to recover the absolute scale of the scene!

Slide: S. Lazebnik

101

Structure from motion ambiguity

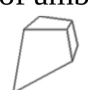
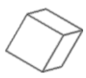
- If we scale the entire scene by some factor k and, at the same time, scale the camera matrices by the factor of $1/k$, the projections of the scene points in the image remain exactly the same
- More generally: if we transform the scene using a transformation $\mathbf{Q}$ and apply the inverse transformation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

$$\mathbf{x} = \mathbf{PX} = \left(\mathbf{PQ}^{-1}\right)\left(\mathbf{QX}\right)$$

Slide: S. Lazebnik

102

Types of ambiguity

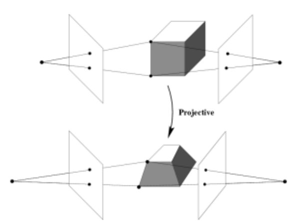
Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, ratios of length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

- With no constraints on the camera 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

103

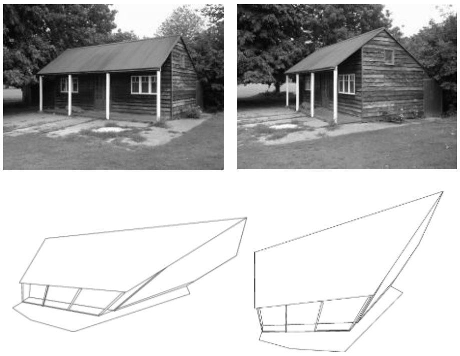
Projective ambiguity



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

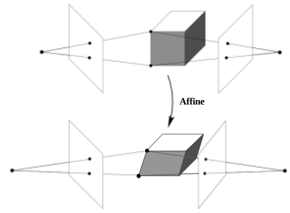
104

Projective ambiguity



105

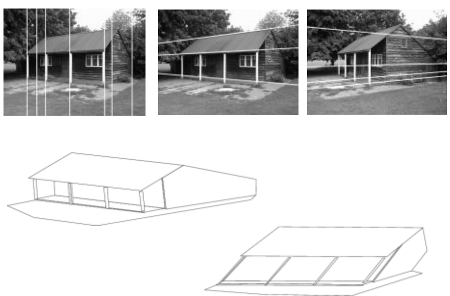
Affine ambiguity



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

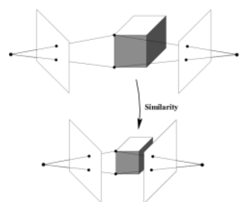
106

Affine ambiguity



107

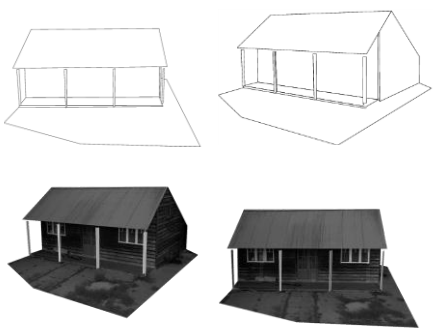
Similarity ambiguity



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

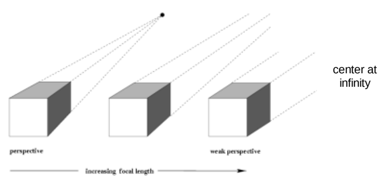
108

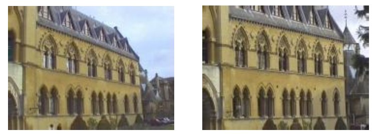
Similarity ambiguity



109

Affine Structure from motion

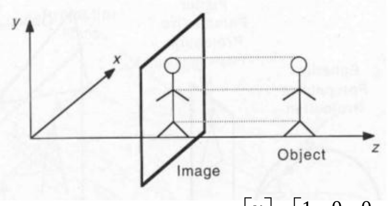




110

Recall: Orthographic Projection

- Special case of perspective projection
  - Distance from the COP to the image plane is 'infinite'



  - Also called "parallel projection"
  - Projection matrix:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

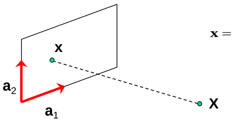
111

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:

$$\mathbf{P} = [3\times3\,\text{affine}] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [4\times4\,\text{affine}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

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

112

Affine structure from motion

- Given: m images of n fixed 3D points:
  - $\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i$, $i = 1, \dots, m, \; j = 1, \dots, n$
- 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 $\mathbf{Q}$ (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 $2mn$ knowns and $8m + 3n$ unknowns (minus 12 dof for affine ambiguity)
- Thus, we must have $2mn >= 8m + 3n - 12$
- For two views, we need four point correspondences

113

Affine structure from motion

- Centering: subtract the centroid of the image points (removes translation)

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

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

114

Affine structure from motion

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

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix}$$

cameras ($2m$)

points (n)

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

115

Affine structure from motion

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

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

points ($3 \times n$)

cameras ($2m \times 3$)

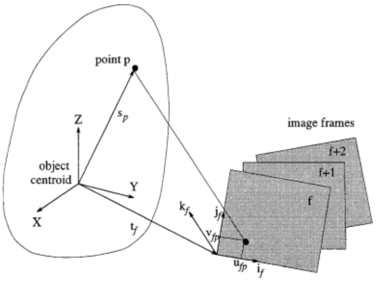
The measurement matrix $\mathbf{D} = \mathbf{MS}$ must have rank 3!

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

116

Affine structure from motion

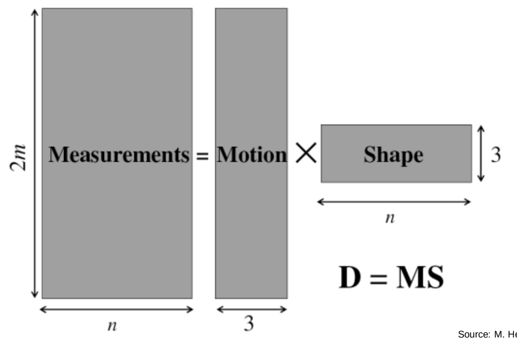
Main idea: You only need M camera orientations and N points



Because Affine transformation is linear, use a linear factorization

117

Factorizing the measurement matrix

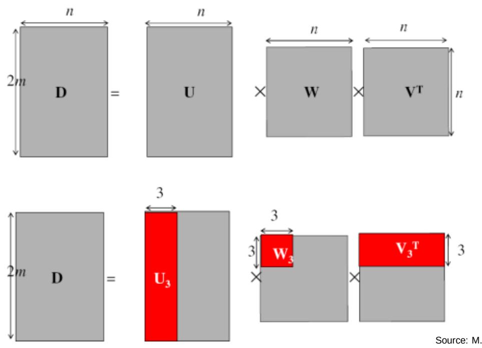


Source: M. Hebert

118

Factorizing the measurement matrix

- Singular value decomposition of D:

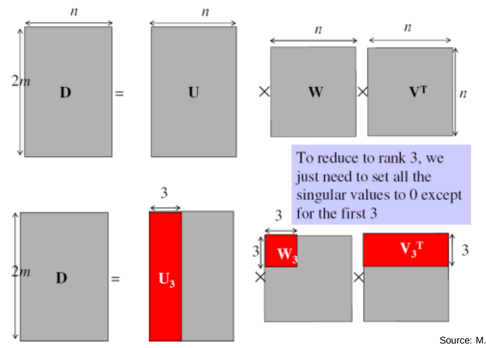


Source: M. Hebert

119

Factorizing the measurement matrix

- Singular value decomposition of D:

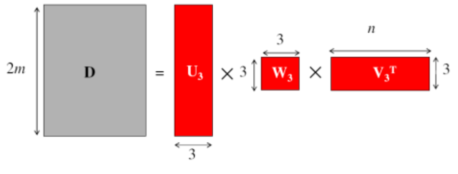


Source: M. Hebert

120

Factorizing the measurement matrix

- Obtaining a factorization from SVD:

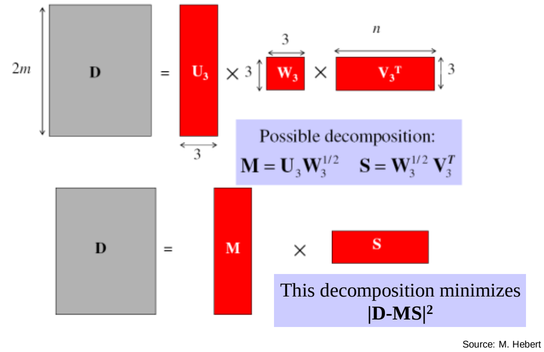


Source: M. Hebert

121

Factorizing the measurement matrix

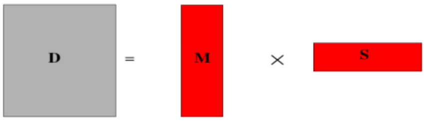
- Obtaining a factorization from SVD:



Source: M. Hebert

122

Affine ambiguity



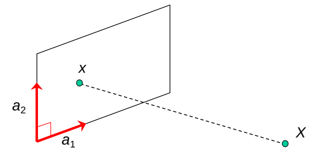
- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations **M** → **MC, S** →**C**$^{-1}$**S**
  (The true M and S are a linear transformation of M, S)
- That is because we have only an affine transformation and we have not enforced any Euclidean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

Source: M. Hebert

123

Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and scale is 1



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

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

$\tilde{\mathbf{a}}_{i1}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i1}^T = 1$

$\tilde{\mathbf{a}}_{i2}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i2}^T = 1$

$\tilde{\mathbf{a}}_{i1}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i2}^T = 0$

- This translates into 3m equations in $\mathbf{L} = \mathbf{CC}^T$:

  $\mathbf{A}_i \mathbf{L} \mathbf{A}_i^T = \mathbf{Id}$, $\quad i = 1, \ldots, m$

  - Solve for **L**
  - Recover **C** from **L** by Cholesky decomposition: $\mathbf{L} = \mathbf{CC}^T$
  - Update **M** and **S**: $\mathbf{M} = \mathbf{MC}$, $\mathbf{S} = \mathbf{C}^{-1}\mathbf{S}$

Lazebnik

Source: M. Hebert

124

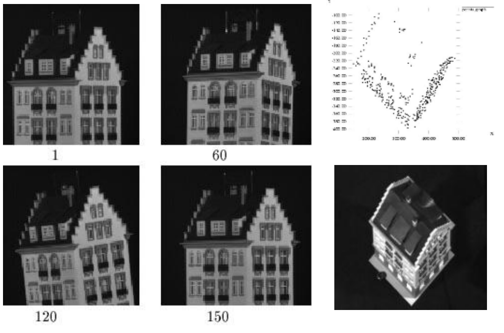
Algorithm summary

- Given: m images and n features $\mathbf{x}_{ij}$
- For each image i, center the feature coordinates
- Construct a 2m × n measurement matrix **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 **D**:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^T$
  - Create $\mathbf{U}_3$ by taking the first 3 columns of **U**
  - Create $\mathbf{V}_3$ by taking the first 3 columns of **V**
  - Create $\mathbf{W}_3$ by taking the upper left 3 × 3 block of **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

Source: M. Hebert

125

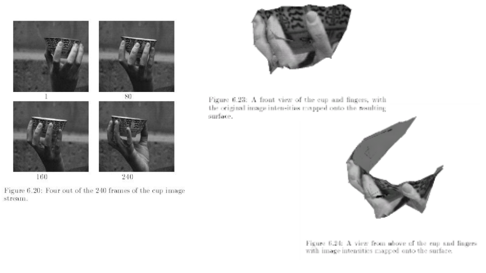
Reconstruction results



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

126

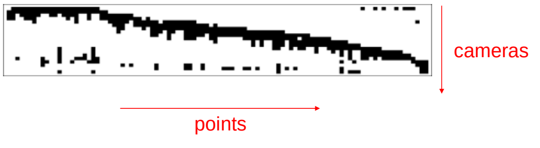
Results



127

Dealing with missing data

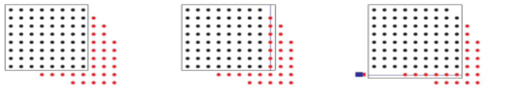
- So far, we have assumed that all points are visible in all views
- In reality, the measurement matrix typically looks something like this:



128

Dealing with missing data

- Possible solution: decompose matrix into dense sub-blocks, factorize each sub-block, and fuse the results



(1) Perform factorization on a dense sub-block

(2) Solve for a new 3D point visible by at least two known cameras (linear least squares)

(3) Solve for a new camera that sees at least three known 3D points (linear least squares)

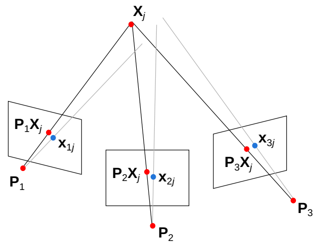
F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, Modeling, and Matching Video Clips Containing Multiple Moving Objects. PAMI 2007.

129

Bundle adjustment

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

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



130

Further Factorization work

Factorization with uncertainty
(Irani & Anandan, IJCV'02)

Factorization for indep. moving objects
(Costeira and Kanade '94)

Factorization for articulated objects
(Yan and Pollefeys '05)

Factorization for dynamic objects
(Bregler et al. 2000, Brand 2001)

Perspective factorization
(Sturm & Triggs 1996, …)

Factorization with outliers and missing pts.
(Jacobs '97 (affine), Martinek & Pajdla'01 Aanaes'02 (perspective))

Pollefeys

131

Questions?

132

Summary

- Projections
- Pin-hole camera
- Transformation
- Homography
- Ransac
- Structure from Motion