Alignment & Warping

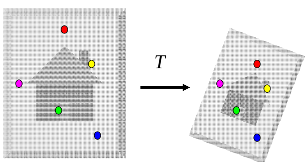
Thursday, Oct 9

Kristen Grauman
UT-Austin

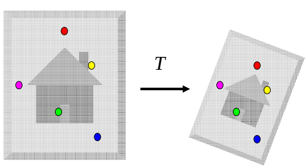
Today

- Alignment & warping
  - 2d transformations
  - Forward and inverse image warping
  - Constructing mosaics
    - Homographies
- Robust fitting with RANSAC

Main questions

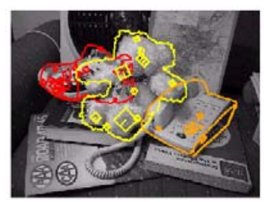


Warping: Given a source image and a transformation, what does the transformed output look like?



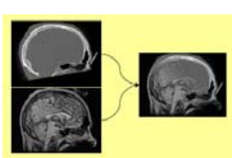
Alignment: Given two images with corresponding features, what is the transformation between them?

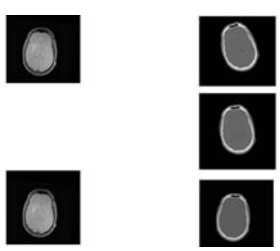
Motivation: Recognition



Figures from David Lowe

Motivation: medical image registration





Motivation: Mosaics

- Getting the whole picture
  - Consumer camera: 50˚ x 35˚

Slide from Brown & Lowe 2003

Motivation: Mosaics

- Getting the whole picture
  - Consumer camera: 50° x 35°
  - Human Vision: 176° x 135°

Slide from Brown & Lowe 2003

Motivation: Mosaics

- Getting the whole picture
  - Consumer camera: 50° x 35°
  - Human Vision: 176° x 135°

- Panoramic Mosaic = up to 360 x 180°

Slide from Brown & Lowe 2003

Warping problem

- Given a set of points and a transformation, generate the warped image

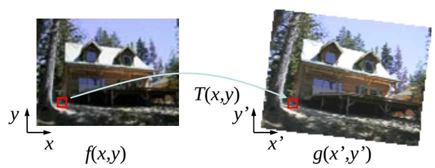
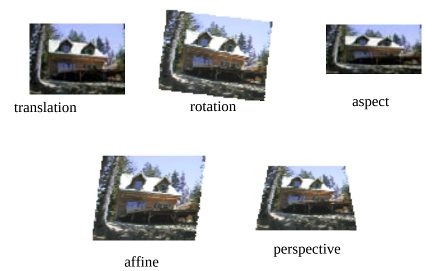


Figure Alyosha Efros

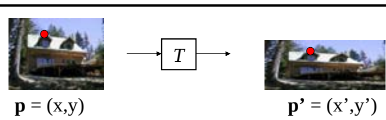
Parametric (global) warping

Examples of parametric warps:



Source: Alyosha Efros

Parametric (global) warping



Transformation T is a coordinate-changing machine:

p' = *T*(p)

What does it mean that *T* is **global**?

- Is the same for any point p
- can be described by just a few numbers (parameters)

Let's represent *T* as a matrix:

p' = **M**p

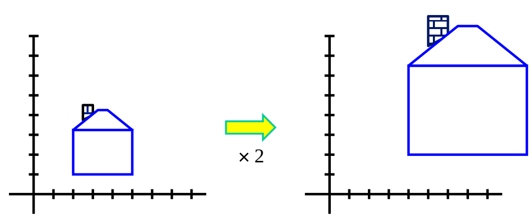
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Source: Alyosha Efros

Scaling

Scaling a coordinate means multiplying each of its components by a scalar

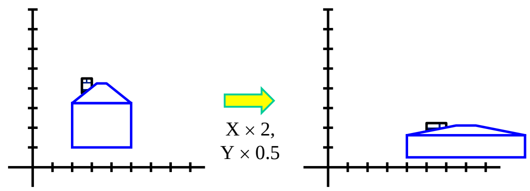
Uniform scaling means this scalar is the same for all components:



Source: Alyosha Efros

Scaling

Non-uniform scaling: different scalars per component:



Source: Alyosha Efros

Scaling

Scaling operation:

$$x' = ax$$
$$y' = by$$

Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix S}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Source: Alyosha Efros

What transformations can be represented with a 2x2 matrix?

2D Scaling?

$$\mathbf{x}' = \mathbf{s}_x * \mathbf{x}$$
$$\mathbf{y}' = \mathbf{s}_y * \mathbf{y}$$

$$\begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \end{bmatrix} = \begin{bmatrix} \mathbf{s}_x & 0 \\ 0 & \mathbf{s}_y \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$$

2D Rotate around (0,0)?

$$x' = \cos\Theta * x - \sin\Theta * y$$
$$y' = \sin\Theta * x + \cos\Theta * y$$

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

2D Shear?

$$\mathbf{x}' = \mathbf{x} + \mathbf{sh}_x * \mathbf{y}$$
$$\mathbf{y}' = \mathbf{sh}_y * \mathbf{x} + \mathbf{y}$$

$$\begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{sh}_x \\ \mathbf{sh}_y & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$$

Source: Alyosha Efros

What transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

$$x' = -x$$
$$y' = y$$

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

2D Mirror over (0,0)?

$$x' = -x$$
$$y' = -y$$

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

2D Translation?

$$\mathbf{x}' = \mathbf{x} + \mathbf{t}_x$$
$$\mathbf{y}' = \mathbf{y} + \mathbf{t}_y$$

NO!

Source: Alyosha Efros

2D Linear Transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Only linear 2D transformations can be represented with a 2x2 matrix.

Linear transformations are combinations of …

- Scale,
- Rotation,
- Shear, and
- Mirror

Source: Alyosha Efros

Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix using homogeneous coordinates?

$$\mathbf{x}' = \mathbf{x} + \mathbf{t}_x$$
$$\mathbf{y}' = \mathbf{y} + \mathbf{t}_y$$

Source: Alyosha Efros

Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix using homogeneous coordinates?

$$x' = x + t_x$$
$$y' = y + t_y$$

A: Using the rightmost column:

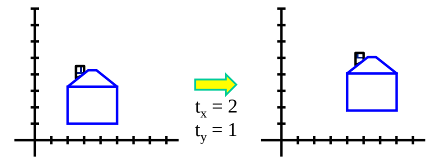
$$\textit{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Source: Alyosha Efros

Translation

Homogeneous Coordinates

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



Source: Alyosha Efros

Basic 2D 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

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

Shear

Source: Alyosha Efros

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

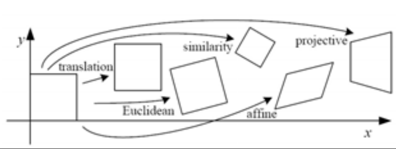
Projective Transformations

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

Projective transformations:

- Affine transformations, and
- Projective warps

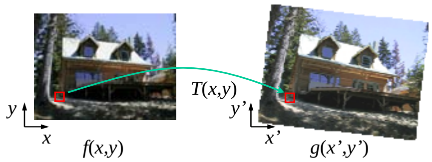
Parallel lines do not necessarily remain parallel



Today

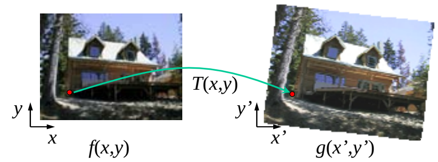
- Alignment & warping
  - 2d transformations
  - Forward and inverse image warping
  - Constructing mosaics
    - Homographies
- Robust fitting with RANSAC

Image warping



Given a coordinate transform and a source image *f*(*x,y*), how do we compute a transformed image *g*(*x',y'*) = *f*(*T*(*x,y*))?

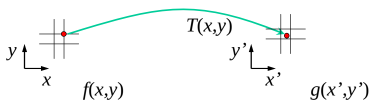
Forward warping



Send each pixel *f*(*x,y*) to its corresponding location (*x',y'*) = *T*(*x,y*) in the second image

Q: what if pixel lands "between" two pixels?

Forward warping



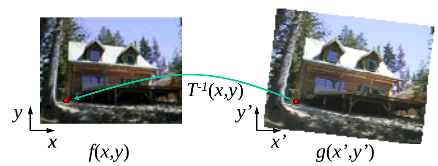
Send each pixel *f*(*x,y*) to its corresponding location (*x',y'*) = *T*(*x,y*) in the second image

Q: what if pixel lands "between" two pixels?

A: distribute color among neighboring pixels (x',y')

- Known as "splatting"

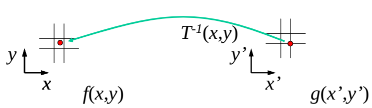
Inverse warping



Get each pixel *g*(*x',y'*) from its corresponding location (*x,y*) = T^{-1}(*x',y'*) in the first image

Q: what if pixel comes from "between" two pixels?

Inverse warping



Get each pixel *g*(*x',y'*) from its corresponding location (*x,y*) = T^{-1}(*x',y'*) in the first image

Q: what if pixel comes from "between" two pixels?

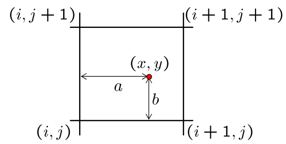
A: *Interpolate* color value from neighbors

- nearest neighbor, bilinear…

`>> help interp2`

Bilinear interpolation

Sampling at *f(x,y):*



$$\begin{aligned} f(x,y) = \; & (1-a)(1-b) \quad f[i,j] \\ & +a(1-b) \quad f[i+1,j] \\ & +ab \quad f[i+1,j+1] \\ & +(1-a)b \quad f[i,j+1] \end{aligned}$$

Alignment problem

- We have previously considered how to fit a model to image evidence
  - e.g., a line to edge points, or a snake to a deforming contour
- In alignment, we will fit the parameters of some **transformation** according to a set of matching feature pairs ("correspondences").

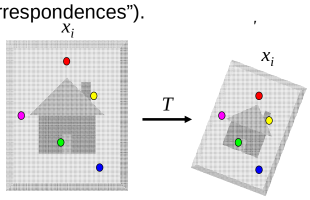
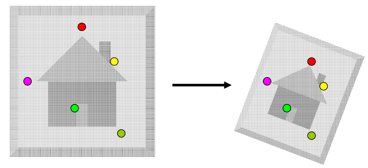


Image alignment



- Two broad approaches:
  - Direct (pixel-based) alignment
    - Search for alignment where most pixels agree
  - Feature-based alignment
    - Search for alignment where *extracted features* agree
    - Can be verified using pixel-based alignment

Source: L. Lazebnik

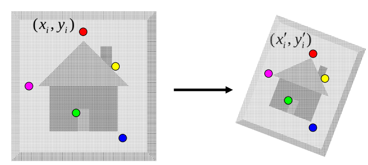
Fitting an affine transformation

Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999

Fitting an affine transformation

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



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

An aside: Least Squares Example

Say we have a set of data points (X1,X1'), (X2,X2'), (X3,X3'), etc. (e.g. person's height vs. weight)

We want a nice compact formula (a line) to predict X's from Xs: Xa + b = X'

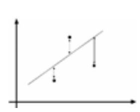
We want to find a and b

How many (X,X') pairs do we need?

$$X_1 a + b = X_1' \qquad X_2 a + b = X_2' \qquad \begin{bmatrix} X_1 & 1 \\ X_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} X_1' \\ X_2' \end{bmatrix} \qquad Ax=B$$

What if the data is noisy?

$$\begin{bmatrix} X_1 & 1 \\ X_2 & 1 \\ X_3 & 1 \\ \dots & \dots \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} X_1' \\ X_2' \\ X_3' \\ \dots \end{bmatrix} \qquad \min \|Ax - B\|^2$$

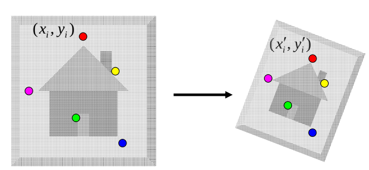


overconstrained

Source: Alyosha Efros

Fitting an affine transformation

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



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

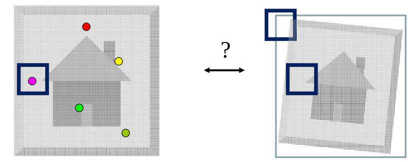
$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

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?
- Once we have solved for the parameters, how do we compute the coordinates of the corresponding point for (x_{new}, y_{new}) ?

What **are** the correspondences?



- Compare content in **local** patches, find best matches.
  e.g., simplest approach: scan with template, and compute SSD or correlation between list of pixel intensities in the patch
- Later in the course: how to select regions according to the geometric changes, and more robust descriptors.

Panoramas

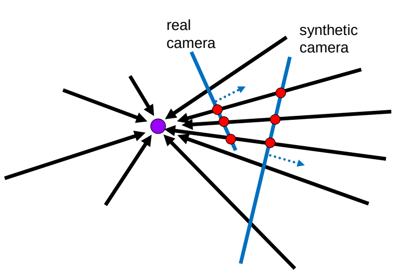
Obtain a wider angle view by combining multiple images.

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?
  - Why aren't we using it?

Source: Steve Seitz

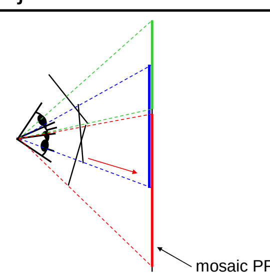
Panoramas: generating synthetic views



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

Source: Alyosha Efros

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

Homography

How to relate two images from the same camera center?

– how to map a pixel from PP1 to PP2?

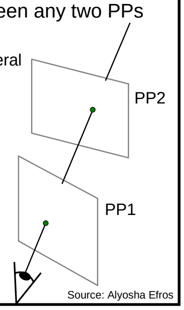
Think of it as a 2D **image warp** from one image to another.

A projective transform is a mapping between any two PPs with the same center of projection

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

called **Homography**

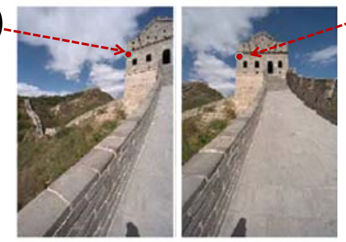
$$\underset{\mathbf{p'}}{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}} = \underset{\mathbf{H}}{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}} \underset{\mathbf{p}}{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$



Source: Alyosha Efros

Homography

$$(x, y)$$



$$\left({}^{wx'}\!/_{w}, {}^{wy'}\!/_{w} \right) = (x', y')$$

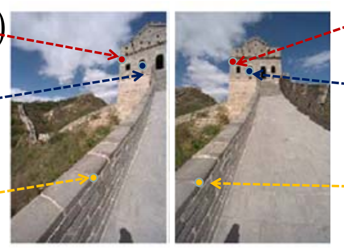
To **apply** a given homography **H**

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

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

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…

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:

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

BOARD

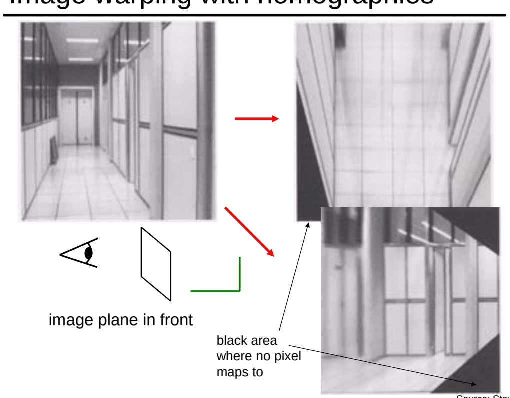
```
>> help lmdivide
```

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)

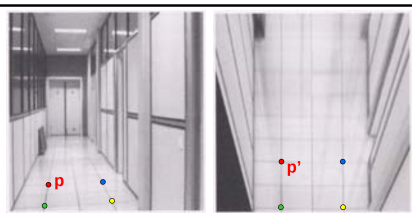
Source: Steve Seitz

Image warping with homographies



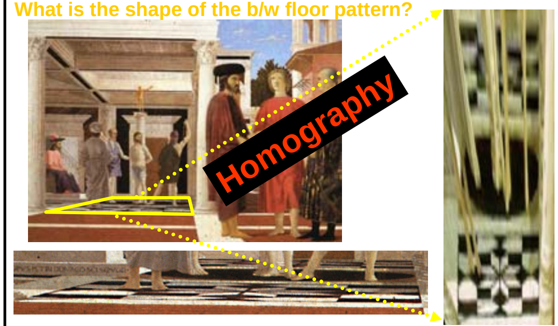
Source: Steve Seitz

Image rectification



Analysing patterns and shapes

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

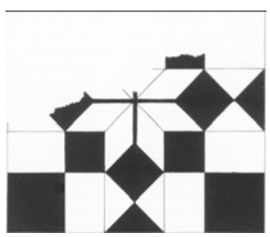


The floor (enlarged)

Automatically rectified floor

Slide from Criminisi

Analysing patterns and shapes



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

Slide from Criminisi

Analysing patterns and shapes

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

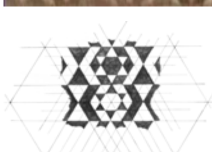
What is the (complicated) shape of the floor pattern?

Automatically rectified floor

Slide from Criminisi

Analysing patterns and shapes

Automatic rectification

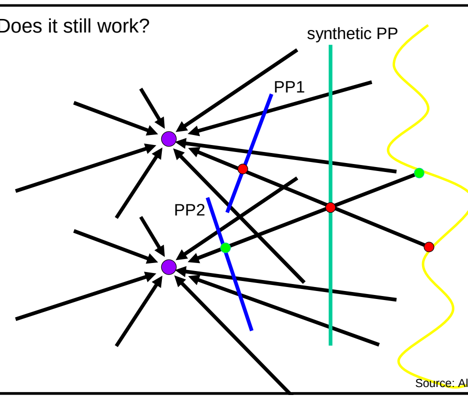


From Martin Kemp, *The Science of Art* *(manual reconstruction)*

Slide from Criminisi

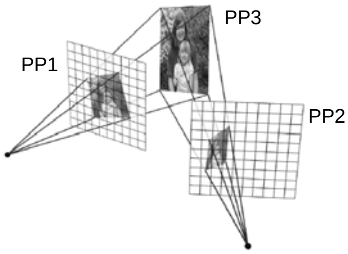
changing camera center

Does it still work?



Source: Alyosha Efros

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

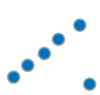
Source: Alyosha Efros

Today

- Alignment & warping
  - 2d transformations
  - Forward and inverse image warping
  - Constructing mosaics
    - Homographies
- Robust fitting with RANSAC

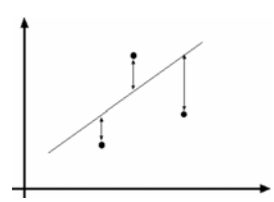
Outliers

- **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.

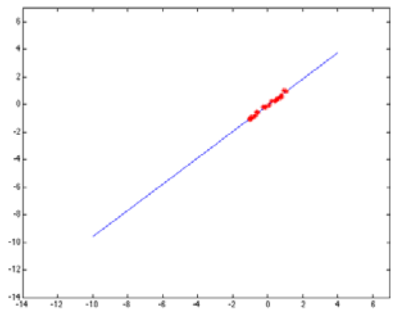


Example: least squares line fitting

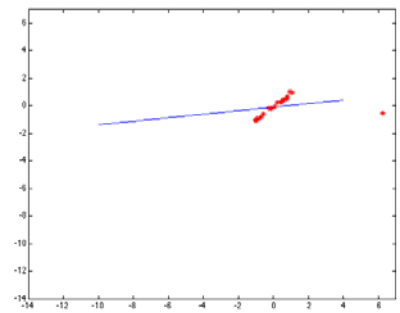
- Assuming all the points that belong to a particular line are known



Outliers affect least squares fit



Outliers affect least squares fit



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.

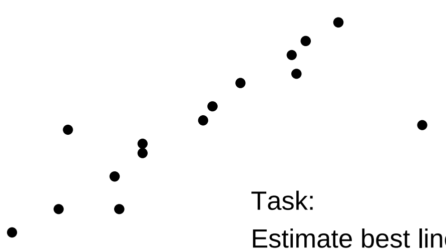
RANSAC

- 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

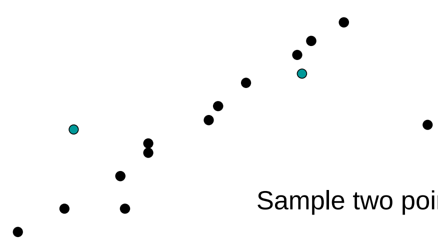
RANSAC Line Fitting Example



Task:
Estimate best line

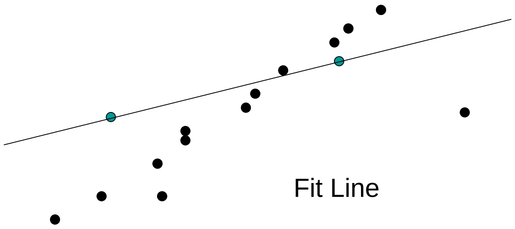
Slide credit: Jinxiang Chai, CMU

RANSAC Line Fitting Example



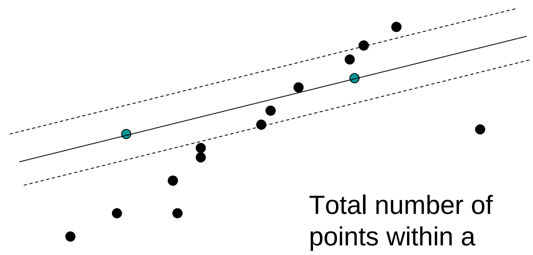
Sample two points

RANSAC Line Fitting Example



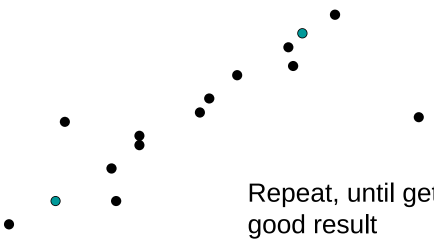
Fit Line

RANSAC Line Fitting Example



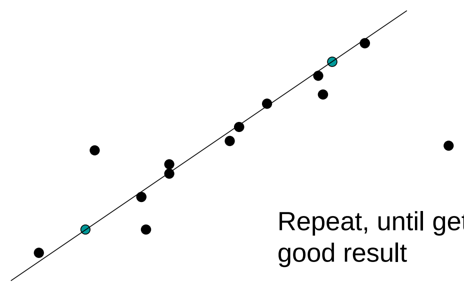
Total number of points within a threshold of line.

RANSAC Line Fitting Example



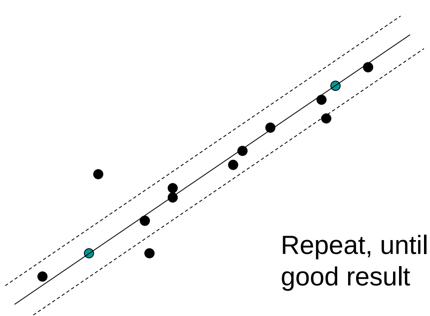
Repeat, until get a good result

RANSAC Line Fitting Example



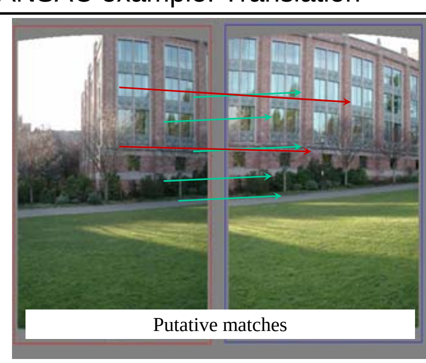
Repeat, until get a good result

RANSAC Line Fitting Example



Repeat, until get a good result

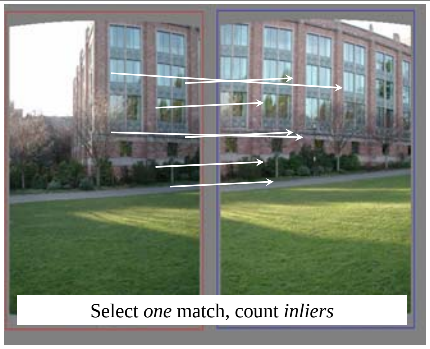
RANSAC example: Translation



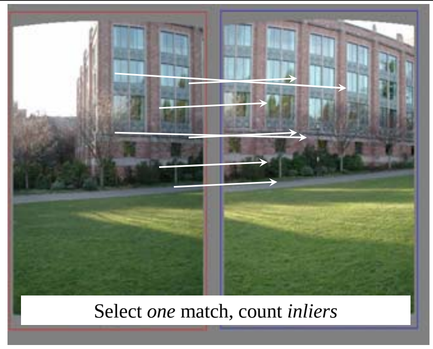
Putative matches

Source: Rick Szeliski

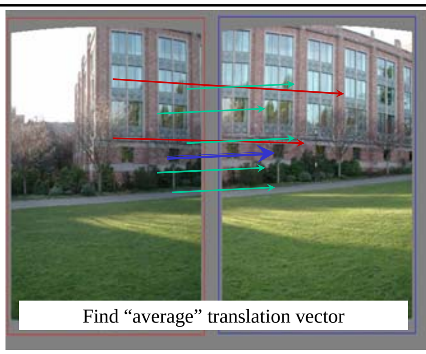
RANSAC example: Translation



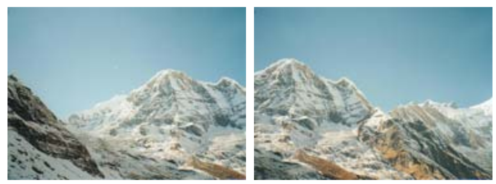
RANSAC example: Translation



RANSAC example: Translation

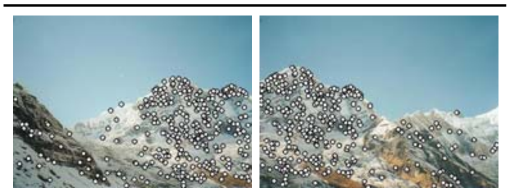


Feature-based alignment outline



Source: L. Lazebnik

Feature-based alignment outline



- Extract features

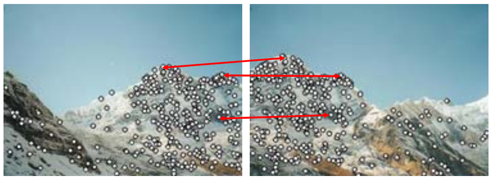
Source: L. Lazebnik

Feature-based alignment outline

- Extract features
- Compute *putative matches*

Source: L. Lazebnik

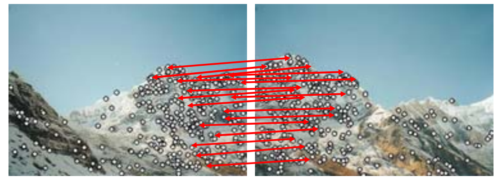
Feature-based alignment outline



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

Source: L. Lazebnik

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

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

Today

- Alignment & warping
  – 2d transformations
  – Forward and inverse image warping
  – Constructing mosaics
    - Homographies
- Robust fitting with RANSAC
- Midterm on Tuesday in class