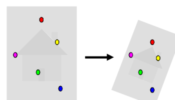
Fitting a transformation: feature-based alignment

Kristen Grauman
UT Austin
Tues Mar 5



Previously

- Interest point detection
  - Harris corner detector
  - Laplacian of Gaussian, automatic scale selection
- Invariant descriptors
  - Rotation according to dominant gradient direction
  - Histograms for robustness to small shifts and translations (SIFT descriptor)

Review questions

- What is the purpose of the “ratio test” for local feature matching?
- What aspects of the SIFT descriptor design promote robustness to lighting changes? Robustness to rotation and translation?
- Does extracting multiple keypoints for multiple local maxima in scale space help recall or precision during feature matching?
- How far in the image plane can an object rotate before the SIFT descriptors will not match?

Multi-view: what's next

Additional questions we need to address to achieve these applications:

- Fitting a parametric transformation given putative matches
- Dealing with outlier correspondences
- Exploiting geometry to restrict locations of possible matches
- Triangulation, reconstruction
- Efficiency when indexing so many keypoints

Motivation: Recognition

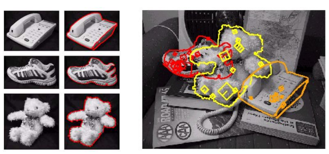
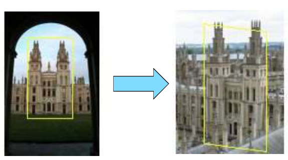
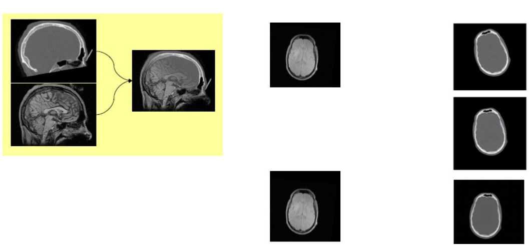


Figure from David Lowe



Motivation: medical image registration



Motivation: mosaics

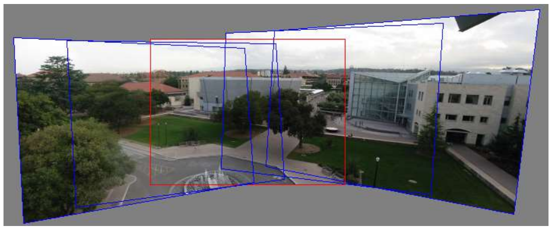
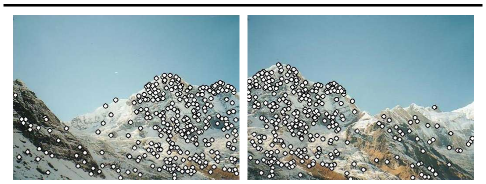


Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fa

Robust feature-based alignment

Source: L. Lazebnik

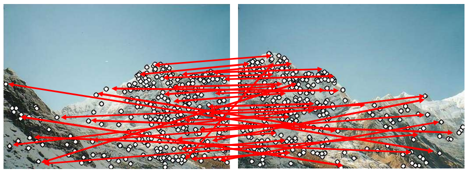
Coming up: robust feature-based alignment



- Extract features

Source: L. Lazebnik

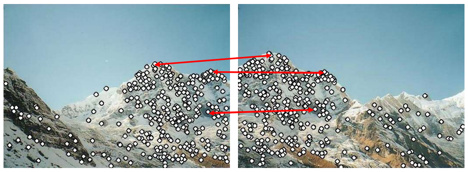
Coming up: robust feature-based alignment



- Extract features
- Compute *putative matches*

Source: L. Lazebnik

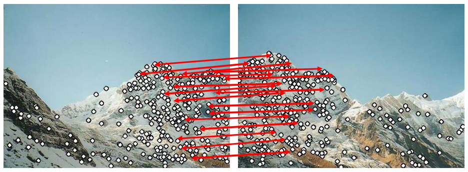
Coming up: robust feature-based alignment



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

Source: L. Lazebnik

Coming up: robust feature-based alignment



- 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

Coming up: robust feature-based alignment

- 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*)

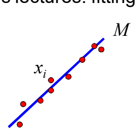
Source: L. Lazebnik

Now

- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC for robust fitting

Alignment as fitting

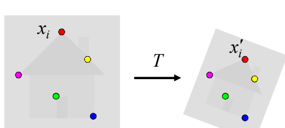
- Previous lectures: fitting a model to features in one image



Find model M that minimizes

$$\sum_i \text{residual}(x_i, M)$$

- Alignment: fitting a transformation between pairs of features (*matches*) in two images



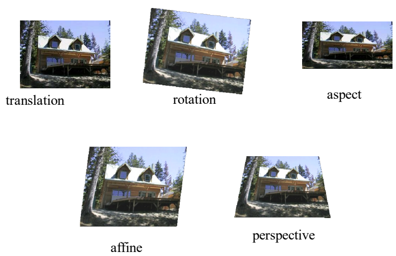
Find transformation T that minimizes

$$\sum_i \text{residual}(T(x_i), x_i')$$

Adapted from: Lana Lazebnik

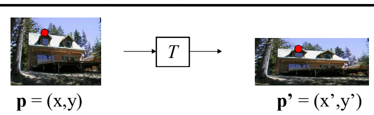
Parametric (global) warping

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

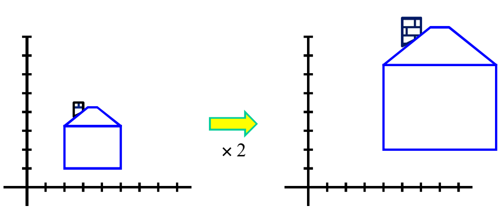
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Source: Alyosha Efros

Scaling

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

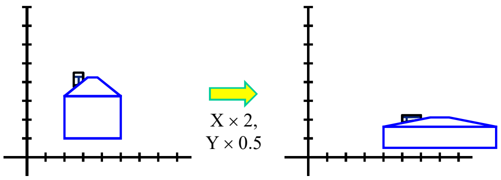
Uniform scaling means this scalar is the same for all components:



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?

$$x' = s_x * x \qquad y' = s_y * y$$

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

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?

$$x' = x + sh_x * y \qquad y' = sh_y * x + y$$

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

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?

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

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

To convert to homogeneous coordinates:

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

homogeneous image coordinates

Converting *from* homogeneous coordinates

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

Homogeneous Coordinates

Q: How can we represent 2d translation as a 3x3 matrix using homogeneous coordinates?

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

A: Using the rightmost column:

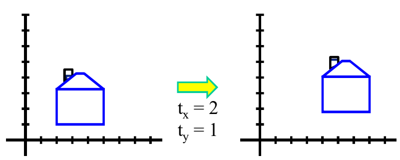
$$\textbf{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

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

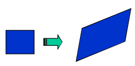
2D Affine Transformations

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

Affine transformations are combinations of …

- Linear transformations, and
- Translations

Parallel lines remain parallel



Today

- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC for robust fitting

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").

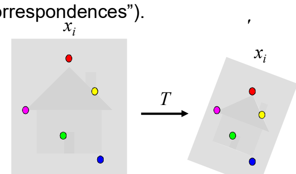
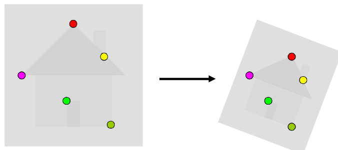


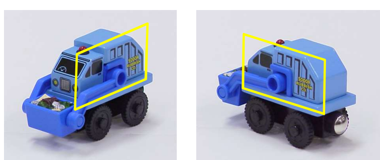
Image alignment



- 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

Let's start with affine transformations

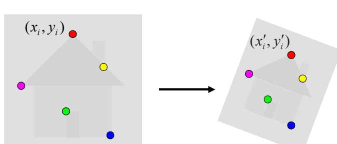
- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras



Slide credit: Lana Lazebnik

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

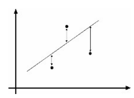
How many (X,X’) pairs do we need?

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

Ax=B

What if the data is noisy?

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

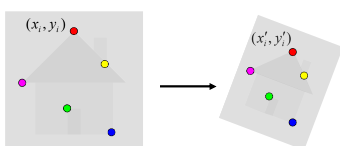


overconstrained

Source: Alyosha Efros

Fitting an affine transformation

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



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

Fitting an affine transformation

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

- 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}) ?
- Where do the matches come from?

Recall: Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]

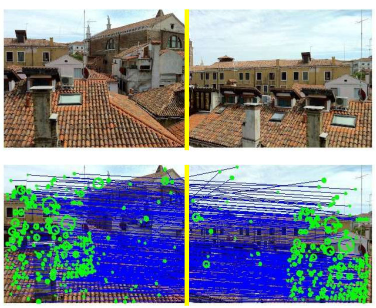
Interest points and their scales and orientations (random subset of 50)

SIFT descriptors

http://www.vlfeat.org/overview/sift.html

Kristen Grauman

Recall: SIFT (preliminary) matches



http://www.vlfeat.org/overview/sift.html

Kristen Grauman

Fitting an affine transformation

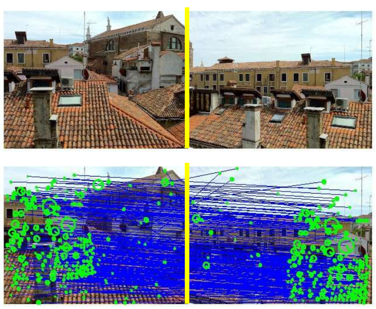
Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999

Today

- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC for robust fitting

Recall: SIFT (preliminary) matches



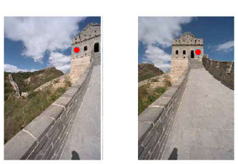
Not all of these are valid matches!

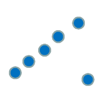
http://www.vlfeat.org/overview/sift.html

Kristen Grauma

Outliers

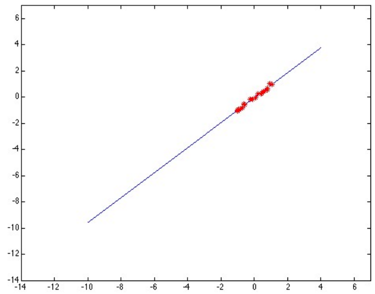
- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn't belong to the line we are fitting.



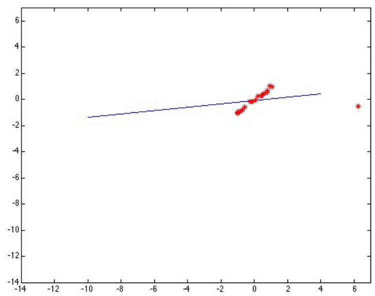


Kristen Grauman

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 (transformation) won't have much support from rest of the points (matches).

Kristen Grauman

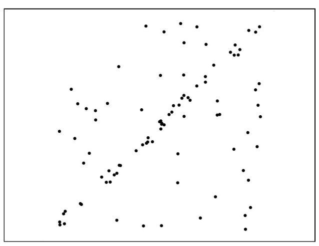
RANSAC for line fitting

Repeat ***N*** times:

- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Lana Lazebnik

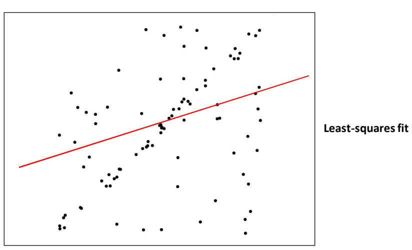
RANSAC for line fitting example



Source: R. Raguram

Lana Lazebnik

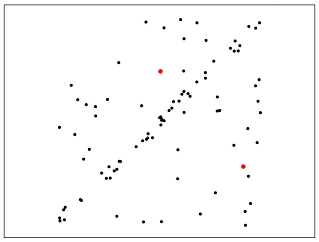
RANSAC for line fitting example



Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

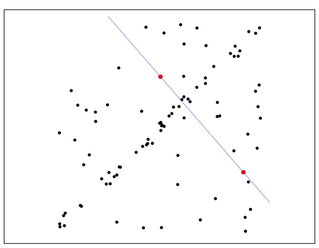


1. Randomly select minimal subset of points

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

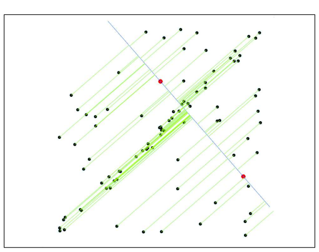


1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

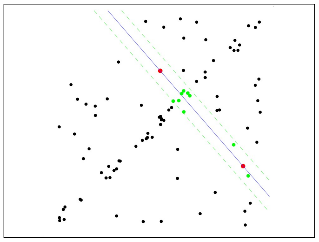


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

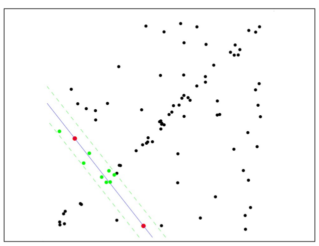


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

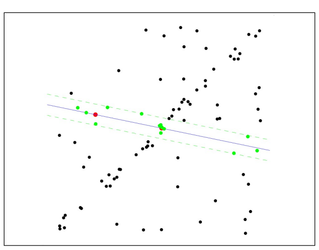


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

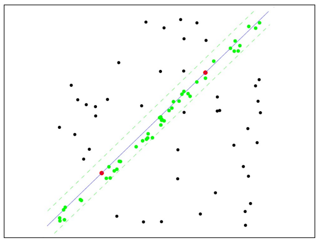
55

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

Uncontaminated sample



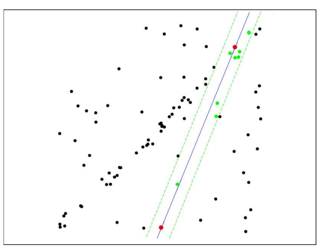
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

56

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Lana Lazebnik

How many trials for RANSAC?

To ensure good chance of finding true inliers, need sufficient number of trials, S.

Let p be probability that any given match is valid

Let P be to the total prob of success after S trials.

Likelihood in one trial that all k random samples are inliers is p^k

Likelihood that all S trials will fa

$$1-P = (1-p^k)^S$$

Required minimum number of t

$$S = \log(1-P) \ / \ \log(1-p^k)$$

k	p	S
3	0.5	35
6	0.6	97
6	0.5	293

Kristen Grauman

RANSAC song

When you have outliers you may face much frustration
if you include them in a model fitting operation.
But if your model's fit to a sample set of minimal size,
the probability of the set being outlier-free will rise.
Brute force tests of all sets will cause computational constipation.

N random samples
will provide an example
of a fitted model uninfluenced by outliers. No need to test all combinations!

Each random trial should have its own unique sample set
and make sure that the sets you choose are not degenerate.
N, the number of sets, to choose is based on the probability
of a point being an outlier, and of finding a set that's outlier free.
Updating *N* as you go will minimise the time spent.

So if you gamble
that *N* samples are ample
to fit a model to your set of points, it's likely that you will win the bet.

Select the set that boasts
that its number of inliers is the most (you're almost there).
Fit a new model just to those inliers and discard the rest,
an estimated model for your data is now possessed!
This marks the end point of your model fitting quest

That is an example fitting a model (line)…

What about fitting a transformation (translation)?

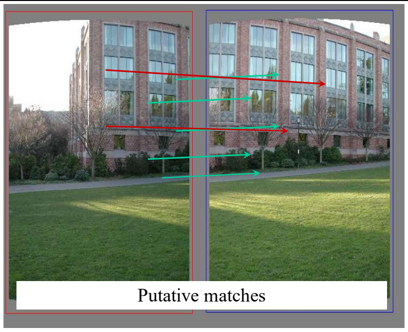
RANSAC: General form

- RANSAC loop:

1. Randomly select a *seed group* 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 estimate of transformation on all of the inliers

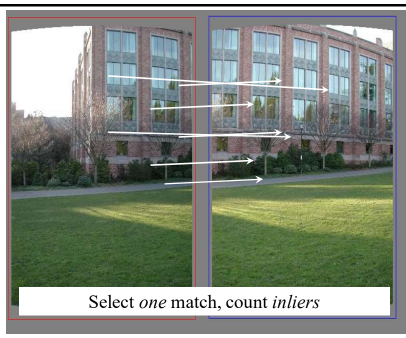
- Keep the transformation with the largest number of inliers

RANSAC example: Translation

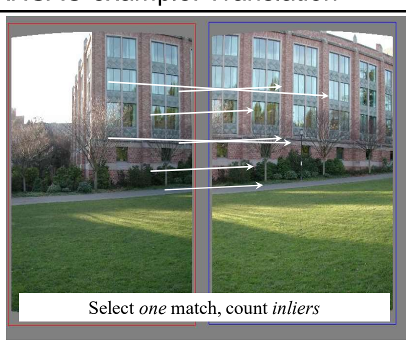


Source: Rick Szeliski

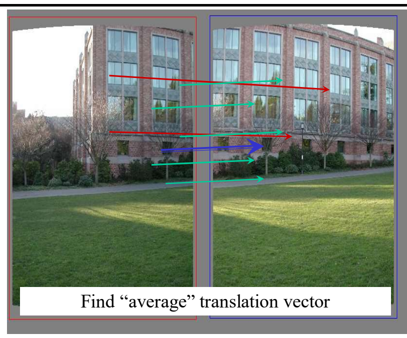
RANSAC example: Translation



RANSAC example: Translation

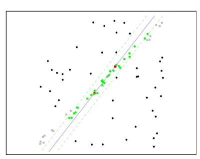


RANSAC example: Translation



RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Parameters to tune
  - Doesn’t work well for low inlier ratios (too many iterations, or can fail completely)
  - Can’t always get a good initialization of the model based on the minimum number of samples



Slide credit: Lana Lazebnik

Another example

Automatic scanned document rotater using Hough lines and RANSAC

https://www.youtube.com/watch?v=O0v9FAk43kY

Recap

- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC

Coming up: alignment and image stitching