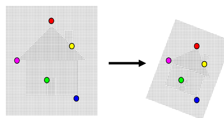
Fitting a transformation: feature-based alignment

Wed, Feb 23

Prof. Kristen Grauman

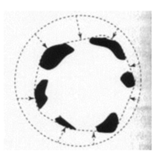
UT-Austin



Announcements

- Reminder: Pset 2 due Wed March 2
- Midterm exam is Wed March 9
  (2 weeks from now)

Last time: Deformable contours



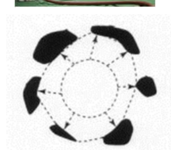


Image from http://www.healthline.com/blogs/exercise_fitness/uploaded_images/HandBand2-795868.JPG

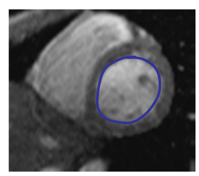
Kristen Grauman

Last time: Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object

Goal: evolve the contour to fit exact object boundary



Main idea: elastic band is iteratively adjusted so as to
- be near image positions with high gradients, **and**
- satisfy shape "preferences" or contour priors

[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Figure credit: Yuri Boykov

Last time: Deformable contours

<u>Pros:</u>
- Useful to track and fit non-rigid shapes
- Contour remains connected
- Possible to fill in "subjective" contours
- Flexibility in how energy function is defined, weighted.

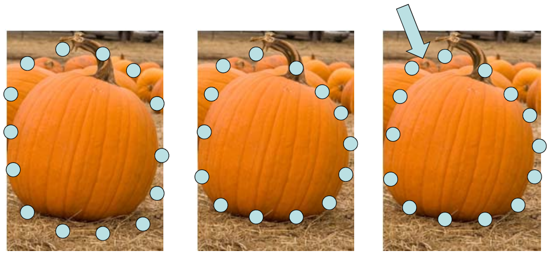
<u>Cons</u>:
- Must have decent initialization near true boundary, may get stuck in local minimum
- Parameters of energy function must be set well based on prior information

Kristen Grauman

Today

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

Interactive forces

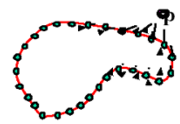


How can we implement such an *interactive* force with deformable contours?

Kristen Grauman

Interactive forces

- An energy function can be altered online based on user input – use the cursor to push or pull the initial snake away from a point.
- Modify external energy term to include:



$$E_{push} = \sum_{i=0}^{n-1} \frac{r^2}{|v_i - p|^2}$$

Nearby points get pushed hardest

Kristen Grauman

Intelligent scissors

Another form of interactive segmentation:

Compute optimal paths **from every point to the seed** based on edge-related costs.

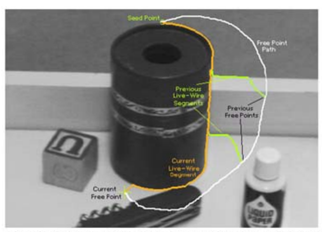
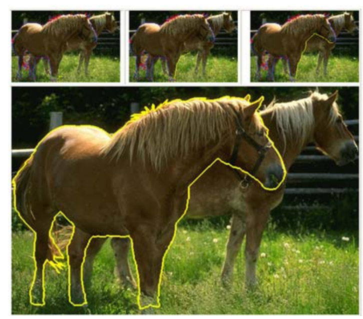


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

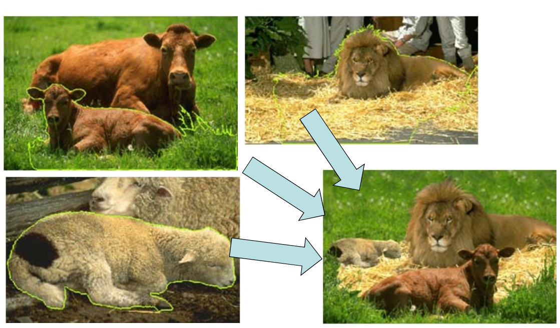
[Mortensen & Barrett, SIGGRAPH 1995, CVPR 1999]

Intelligent scissors



- http://rivit.cs.byu.edu/Eric/Eric.html

Intelligent scissors



- http://rivit.cs.byu.edu/Eric/Eric.html

Beyond boundary snapping…

- Another form of interactive guidance: specify regions
- Usually taken to suggest foreground/background color distributions

User Input

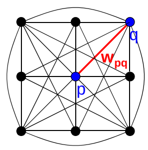
Result

Boykov and Jolly (2001)

How to use this information?

Kristen Grauman

Recall: Images as graphs

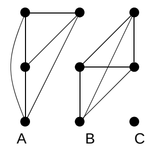


Fully-connected graph

- node for every pixel
- link between *every* pair of pixels, **p**,**q**
- similarity w_{pq} for each link
  - » similarity is *inversely proportional* to difference in color and position

Steve Seitz

Recall: Segmentation by Graph Cuts

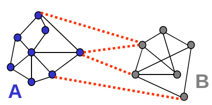


Break graph into segments

- Delete links that cross between segments
- Easiest to break links that have low similarity
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

Steve Seitz

Recall: Segmentation by Graph Cuts



Link Cut

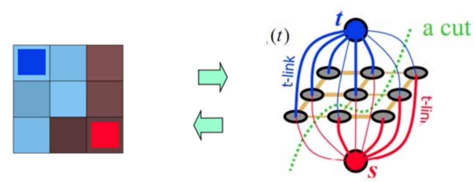
- set of links whose removal makes a graph disconnected
- cost of a cut: $$cut(A,B) = \sum_{p\in A, q\in B} w_{p,q}$$

Find minimum cut

- gives you a segmentation
- fast algorithms exist for doing this

Source: Steve Seitz

Graph cuts for interactive segmentation



Adding hard constraints:

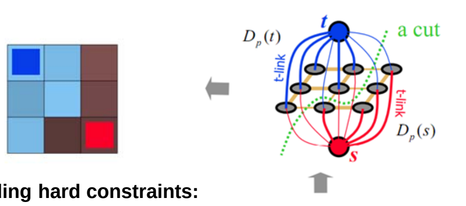
Add two additional nodes, object and background "terminals"

Link each pixel

- To both terminals
- To its neighboring pixels

Yuri Boykov

Graph cuts for interactive segmentation



Adding hard constraints:

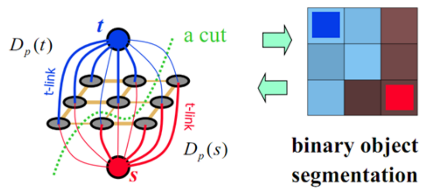
Let the edge weight to object or background terminal reflect similarity to the respective seed pixels.

$$D_p(s) = const - |I_p - I^s|$$
$$D_p(t) = const - |I_p - I^t|$$

Yuri Boykov

Graph cuts for interactive segmentation

$$E(L) = \underbrace{\sum_p -D_p(L_p)}_{\text{t-links}}^{\text{Regional term}} + \underbrace{\sum_{pq\in N} w_{pq}\cdot\delta(L_p \neq L_q)}_{\text{n-links}}^{\text{Boundary term}} \qquad L_p \in \{s,t\}$$

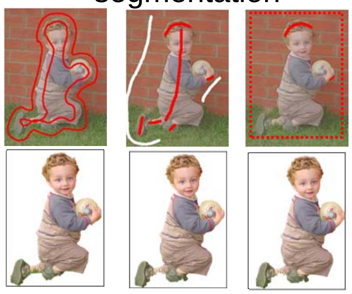


binary object segmentation

Boykov and Jolly (2001)

Yuri Boykov

Graph cuts for interactive segmentation



Another interaction modality: specify bounding box

Intelligent Scissors Mortensen and Barrett (1995)

Graph Cuts Boykov and Jolly (2001)

GrabCut Rother et al. (2004)

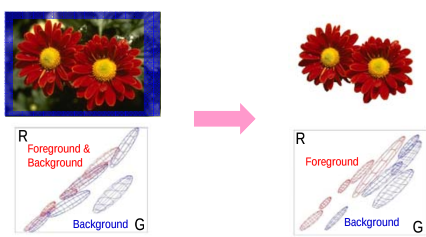
"Grab Cut"

- Loosely specify foreground region
- Iterated graph cut

Rother et al (2004)

"Grab Cut"

- Loosely specify foreground region
- Iterated graph cut



Gaussian Mixture Model (typically 5-8 components)

Rother et al (2004)

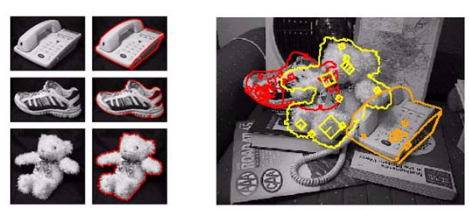
"Grab Cut"

Rother et al (2004)

Today

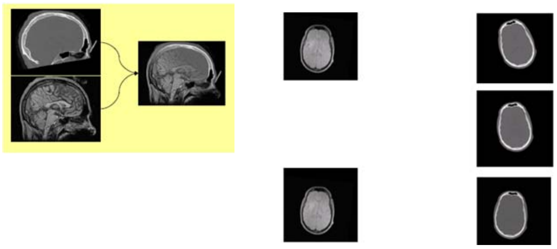
- Interactive segmentation
- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC

Motivation: Recognition



Figures from David Lowe

Motivation: medical image registration



Motivation: mosaics

(In detail next week)

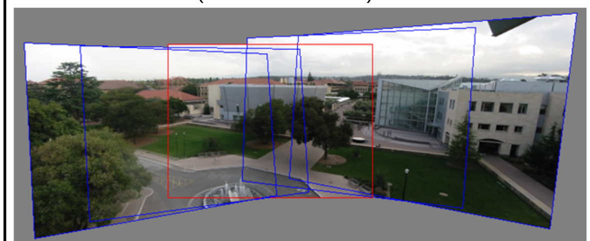
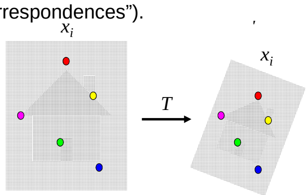


Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fa

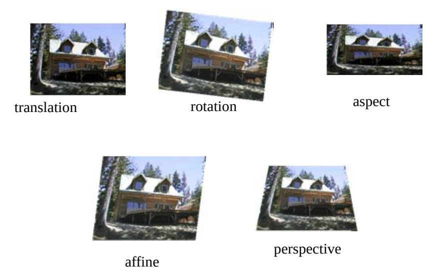
Alignment problem

- 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").



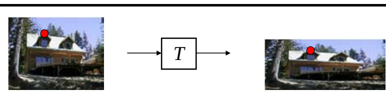
Parametric (global) warping

Examples of parametric warps:



Source: Alyosha Efros

Parametric (global) warping



p = (x,y) **p'** = (x',y')

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

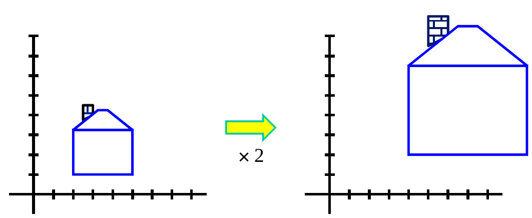
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Source: Alyosha Efros

Scaling

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

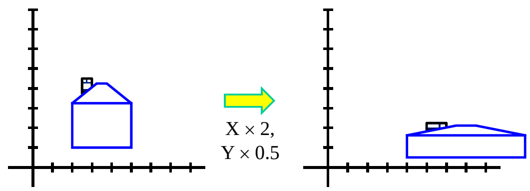
Uniform scaling means this scalar is the same for all components:



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?

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

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

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?

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

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

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?

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

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:

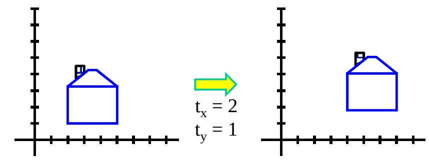
$$\textbf{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

Shear

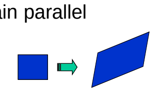
Source: Alyosha Efros

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

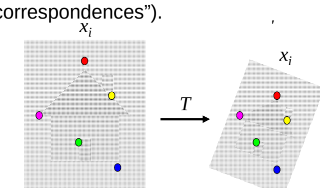


Today

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

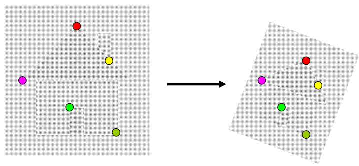
Alignment problem

- 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").



Kristen Grauman

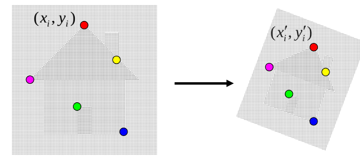
Image alignment



- 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

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'

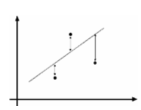
We want to find a and b

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

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

What if the data is noisy?

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

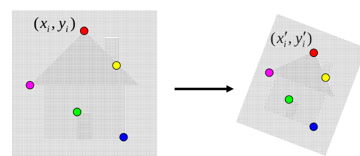


overconstrained

Source: Alyosha Efros

Fitting an affine transformation

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



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

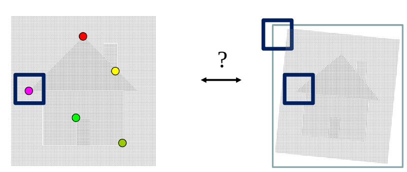
Fitting an affine transformation

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

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

Kristen Grauman

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.

Kristen Grauman

Fitting an affine transformation

Affine model approximates perspective projection of planar objects.

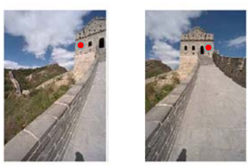
Figures from David Lowe, ICCV 1999

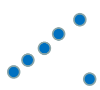
Today

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

Outliers

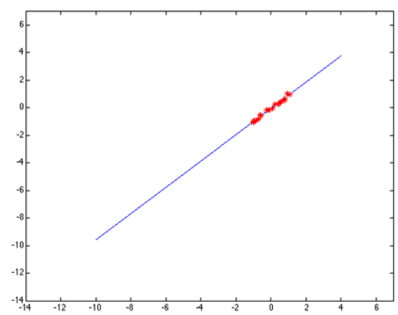
- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn't belong to the line we are fitting.



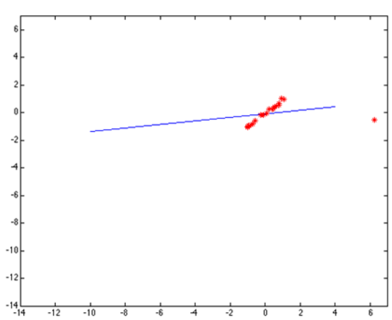


Kristen Grauman

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 won't have much support from rest of the points.

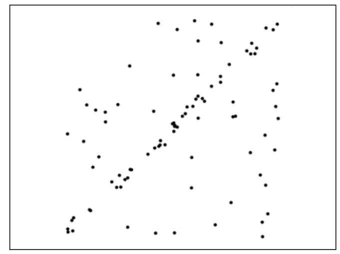
RANSAC: General form

- 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 estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

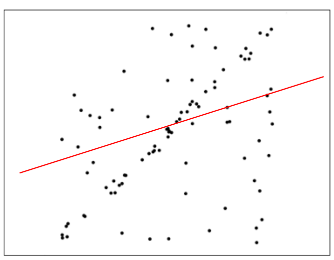
RANSAC for line fitting example



Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

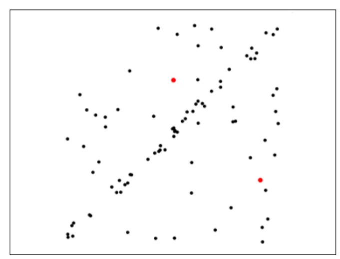


Least-squares fit

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

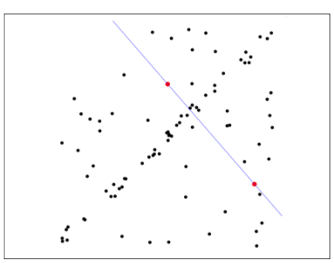


1. Randomly select minimal subset of points

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

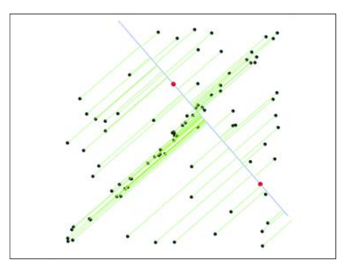


1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

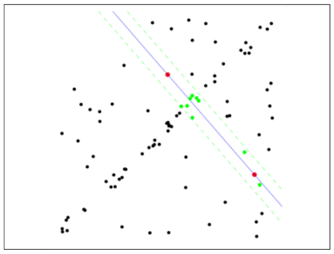


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

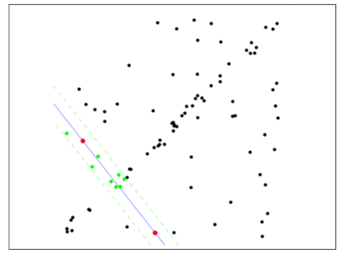


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

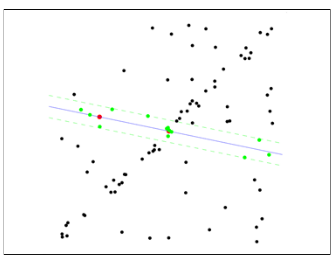


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

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

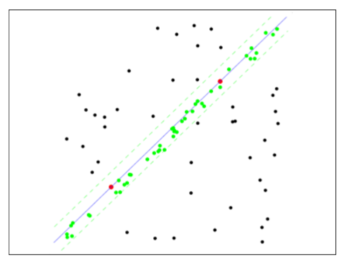
63

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

Uncontaminated sample



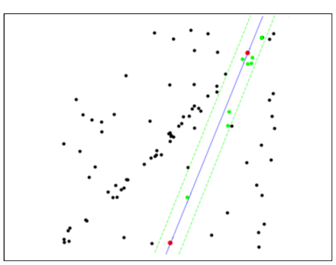
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

64

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

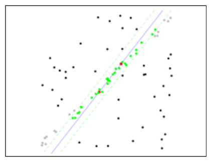
Repeat ***N*** times:

- 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 pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of 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



Lana Lazebnik

Today

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

Coming up: alignment and image stitching