Follow pset instructions

• We will take points off:
  - if you don’t call your methods the right way
  - if you fail to include required images
  - if your readme is not an ascii file
  - if you don’t put things in a ZIP archive
Corners and scale

• Change both initial and tensor blur
• Keep constant factor between the two

\[ \sigma_G = 0.5; \sigma_{\text{tensor}} = 2 \quad \sigma_G = 1; \sigma_{\text{tensor}} = 4 \quad \sigma_G = 2; \sigma_{\text{tensor}} = 8 \]
Fun with homography

Nischay Kumar

Thursday, March 22, 12
Fun with homography

Michael L Puncel
Fun with homography

Anjali Muralidhar
Fun with homography

6.815 is mildly awesome.

Adam Leonard
More fun with homographies

- http://vimeo.com/38940289

- http://www.youtube.com/watch?v=TONif6BC0kc&feature=youtu.be
Corners & tracking

- Corners are also the easy points to track across time

- See e.g. [http://en.wikipedia.org/wiki/Kanade%E2%80%93Lucas%E2%80%93Tomasi_feature_tracker](http://en.wikipedia.org/wiki/Kanade%E2%80%93Lucas%E2%80%93Tomasi_feature_tracker)

Feature Matching and RANSAC

© Krister Parmstrand

Nikon D70. Stitched Panorama. The sky has been retouched. No other image manipulation.

with a lot of slides stolen from
Alyosha Efros, Steve Seitz and Rick Szeliski

Thursday, March 22, 12
Patch descriptor
Basic idea

- Descriptor = kxk patch
Bells and whistles

- In pset 7, we only use luminance values
- Blur image a little before
  - To reduce sampling/aliasing effects
- Correct for potential brightness/contrast changes
  - e.g. because of exposure, vignetting, clouds
  - Subtract mean
  - Normalize by standard deviation

- At the end of the day, a descriptor is $k \times k$ numbers
Descriptor Visualization

- Green=positive, red=negative
Matching descriptors
Status

• We have extracted N1 corners from image 1, and N2 from image 2
• For each corner, we have a kxk descriptor
• The combination of a corner+descriptor is often called a feature point
• Now we need to match feature points from image 1 to image 2
Brute force search

• For each feature point \( i \) in image 1
  - scan all feature points \( j \) in image 2
    - compute (squared) distance between descriptors \( i \) & \( j \)
      - i.e. sum of square difference for \( k \times k \) numbers
    - Keep descriptor with closest distance
Status

- We have a match for each corner of an image
  - But lots of wrong matches
  - Even scene points that are not on the overlap between the images have a match!
Feature matching

- **Exhaustive search**
  - for each feature in one image, look at *all* the other features in the other image(s)
  - Usually not so bad

- **Hashing**
  - compute a short descriptor from each feature vector, or hash longer descriptors (randomly)

- **Nearest neighbor techniques**
  - $k$-trees and their variants (Best Bin First)
Nearest neighbor techniques

- \( k \)-D tree
- Best Bin First (BBF)

Figure 6: \( kd \)-tree with 8 data points labelled A-H, dimension of space \( k=2 \). On the right is the full tree, the leaf nodes containing the data points. Internal node information consists of the dimension of the cut plane and the value of the cut in that dimension. On the left is the 2D feature space carved into various sizes and shapes of bin, according to the distribution of the data points. The two representations are isomorphic. The situation shown on the left is after initial tree traversal to locate the bin for query point “+” (contains point D). In standard search, the closest nodes in the tree are examined first (starting at C). In BBF search, the closest bins to query point \( q \) are examined first (starting at B). The latter is more likely to maximize the overlap of (i) the hypersphere centered on \( q \) with radius \( D_{\text{cur}} \), and (ii) the hyperrectangle of the bin to be searched. In this case, BBF search reduces the number of leaves to examine, since once point B is discovered, all other branches can be pruned.

Indexing Without Invariants in 3D Object Recognition, Beis and Lowe, PAMI'99
Second-Nearest-Neighbor test
Problem

- How do we tell good matches from bad ones?
Naive idea

• If distance is too big, ignore match
Evaluating naive idea

- Using ground truth
  - e.g. user clicks to tell us which corner matches which

- distribution of good vs. bad matches as function of descriptor difference
  - slightly different descriptor from us though, smarter

- Not a clear threshold
Better idea: Second NN

- Evaluate ratio of distance to best descriptor to that of second best
- If they are too similar, the match is ambiguous

Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

- Much better separation!
Status

• We have feature points
• Matches between two images
  - Most of them are good
  - But not all of them. There can still be error.
  - Next time: RANSAC to remove remaining outliers
Example of outlier
Feature-space outlier rejection

Blue: passed the 2NN test

- Can we now compute $H$ from the blue points?
  - No! Still too many outliers…
  - What can we do?
Still to be learned

• Better detector: across scales
• Better descriptor
  - robust to rotation and scale.
  - e.g. SIFT
• Homography fitting despite the presence of outliers
  - RANSAC
• Correct for optical distortions
• Blending to make seams invisible
• Deal with moving objects
Robustness to outliers
(b) Matching features at the image level--looking for a good homography

Simplified illustration with translation instead of homography

What do we do about the “bad” matches?

Note: at this point we don’t know which ones are good/bad
RAndon SAmple Consensus

Select *one* match, count *inliers*
Random Sample Consensus

Select one match, count inliers

1 inliers


**RAandom SAmple Consensus**

Select *one* match, count *inliers*

5 inliers
Random SAmple Consensus

Select one match, count inliers
Keep match with largest set of inliers
At the end: Least squares fit

Find “average” translation vector, but with only inliers
RANSAC

• [Link](http://portal.acm.org/citation.cfm?id=358692)
RANSAC for estimating homography

RANSAC loop:
- Select four feature pairs (at random)
- Compute homography H (exact)
- Compute inliers where $\|p_i', Hp_i\| < \varepsilon$

Keep largest set of inliers
Re-compute least-squares H estimate on all of the inliers
Simple example: fit a line

- Rather than homography $H$ (8 numbers)
  fit $y=ax+b$ (2 numbers $a$, $b$) to 2D pairs
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers
Simple example: fit a line

• Pick 2 points
• Fit line
• Count inliers

4 inlier
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers
Simple example: fit a line

- Use biggest set of inliers
- Do least-square fit
RANSAC for estimating homography

RANSAC loop:

Select four feature pairs (at random)
Compute homography $H$ (exact)
Compute *inliers* where $\|p_i', H p_i\| < \varepsilon$

Keep largest set of inliers
Re-compute least-squares $H$ estimate on all of the inliers
Correspondences

- I lowered the 2NN threshold to make things more interesting...
RANSAC iterations
Resulting inliers
After reweighted least square

• Took into account near outliers
Reprojection
More results
Probabilistic analysis
Robustness

• Proportion of inliers in our pairs is X
• Our model needs P pairs
– P=4 for homography
• Probability that after N RANSAC iterations we succeed?
Probabilities

• Failure vs. Success
• AND vs. OR

• AND is easy!
Robustness

• Proportion of inliers in our pairs is X

• Our model needs P pairs
  – P=4 for homography

• Probability that we pick P inliers?
  – X^P

• Probability that after N RANSAC iterations we have not picked a set of inliers?
  – (1-X^P)^N
Robustness: example

- Matlab: \( p=4; \ x=0.5; \ n=1000; \ (1-x^p)^n \)
- Proportion of inliers \( X=0.5 \)
- Probability that we pick \( P=4 \) inliers?
  - \( 0.5^4=0.0625 \) (6% chances)
- Probability that we have not picked a set of inliers?
  - \( N=100 \) iterations:
    \( (1-0.5^4)^{100}=0.00157 \) (1 chance in 600)
  - \( N=1000 \) iterations:
    1 chance in \( 1e28 \)
Robustness: example

- Proportion of inliers $X=0.3$
- Probability that we pick $P=4$ inliers?
  - $0.3^4=0.0081$ (0.8% chances)
- Probability that we have not picked a set of inliers?
  - $N=100$ iterations:
    - $(1-0.3^4)^{100}=0.44$ (1 chance in 2)
  - $N=1000$ iterations:
    - 1 chance in 3400
Robustness: example

• Proportion of inliers $X=0.1$

• Probability that we pick $P=4$ inliers?
  $0.1^4=0.0001$ (0.01% chances, 1 in 10,000)

• Probability that we have not picked a set of inliers?
  – $N=100$ iterations: $(1-0.1^4)^{100}=0.99$
  – $N=1000$ iterations: 90%
  – $N=10,000$: 36%
  – $N=100,000$: 1 in 22,000
Graph

Probability to fail

N = 10 iter.

100 iter.

1000 iter.

$10^4$ iter.

$10^6$ iter.

$10^5$ iter.

Percentage of inliers X
Robustness: conclusions

- Effect of number of parameters of model/number of necessary pairs
  - Bad exponential
- Effect of percentage of inliers
  - Base of the exponential
- Effect of number of iterations
  - Good exponential
RANSAC recap

• For fitting a model with low number $P$ of parameters (8 for homographies)

• Loop
  – Select $P$ random data points
  – Fit model
  – Count inliers
    (other data points well fit by this model)

• Keep model with largest number of inliers
Recap: Auto pano

- Extract Interest points (Harris corner)
- Match with descriptors (9x9 patches)
- Second-Nearest-Neighbor test rejects bad matches
- RANSAC fits homography, robust to outliers
Extract Interest points (Harris)
Descriptors
After match & 2NN test
After Ransac
Result with blending
3D version: Photo tourism and
Photo Tourism:
Exploring Photo Collections in 3D

Noah Snavely
Steven M. Seitz
University of Washington
Richard Szeliski
Microsoft Research

© 2006 Noah Snavely
New version with code!

Photo Tourism overview

Input photographs → Scene reconstruction → Photo Explorer

- Relative camera positions and orientations
- Point cloud
- Sparse correspondence
Photo Tourism overview

Input photographs ➔ Scene reconstruction ➔ Photo Explorer
Scene reconstruction

- Automatically estimate
  - position, orientation, and focal length of cameras
  - 3D positions of feature points

Feature detection

Pairwise feature matching

Correspondence estimation

Incremental structure from motion
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]
Feature matching

Match features between each pair of images
Feature matching

Refine matching using RANSAC [Fischler & Bolles 1987] to estimate fundamental matrices between pairs

(See 6.801/6.866 for fundamental matrix, or Hartley and Zisserman, Multi-View Geometry)
Structure from motion

\begin{align*}
\text{minimize} & \quad f(R, T, P) \\
\end{align*}
Incremental structure from motion
Photo Tourism overview

Input photographs → Scene reconstruction → Photo Explorer

- Navigation
- Rendering
- Annotations
Object-based browsing
Object-based browsing

- Visibility
- Resolution
- Head-on view
Relation-based browsing
Relation-based browsing

Image A

Image B

© 2006 Noah Snavely
Relation-based browsing

Image A

Image B
Relation-based browsing

Image A

to the right of

Image B

Image C
Relation-based browsing

Image A

Image B

to the right of

Image C
Relation-based browsing

Image A

Image B

to the right of

is detail of

Image C

Image D

© 2006 Noah Snavely
Relation-based browsing

Image A

to the right of

is detail of

Image C

Image D

Image B
Relation-based browsing

Image A

is zoom-out of Image D

is detail of Image B

to the left of Image C

Image B

is detail of Image A

is zoom-out of Image C

Image C

is detail of Image D

Image D

is detail of Image B

to the right of Image A
Rendering
Rendering
Rendering transitions

Camera A

Camera B
Photo Tourism
Exploring photo collections in 3D

Noah Snavely  Steven M. Seitz  Richard Szeliski
University of Washington  Microsoft Research

SIGGRAPH 2006
Live demo

- see also: http://photosynth.net/
Limitations / Future work

- Not all photos can be reliably matched
  → Better feature detection / matching
  → Integrating GPS & other localization info.
- Structure from motion scalability
  → More efficient (sparse) algorithms
- Plane-based transitions lack parallax
  → Richer transitions
- Photo explorer scalability…
Conclusion

Indexing everyone’s photos provides a new way to share and experience our world

To find out more:
– http://phototour.cs.washington.edu
– http://research.microsoft.com/IVM/PhotoTourism
– http://labs.live.com/photosynth
– Exhibition booth #2619
Links

• Code available:  
  http://phototour.cs.washington.edu/bundler/
• http://phototour.cs.washington.edu/
• http://livelabs.com/photosynth/
• http://www.cs.cornell.edu/~snavely/
• http://da.vidr.cc/projects/pixelstruct/
• http://www.cs.washington.edu/homes/ccwu/vsfm/