CSE 252B: Computer Vision II

Lecturer: Serge Belongie
Scribes: Jeremy Pollock and Neil Alldrin

LECTURE 14
Robust Feature Matching

14.1. Introduction

Last lecture we learned how to find interest points using the Förstner corner detector. The goal of this lecture is to find correspondences between these points automatically. This is known as the *correspondence problem*. The correspondence problem is arguably the hardest problem in structure from motion (SFM). For example, one must deal with pointsets of different length, with spurious coordinates and positional noise. All of SFM relies on the success of finding correspondence between points in the two images. If you have two views that are too widely separated, it can become difficult to match interest points for purposes of estimating a homography or fundamental matrix. Correspondence is an absolutely critical part of the vision problem.

14.2. The Correspondence Problem

There are two categories of correspondence problems: **wide baseline** and **small baseline**. Wide baseline stereo pairs are taken by cameras separated

[1]Department of Computer Science and Engineering, University of California, San Diego.

May 19, 2004

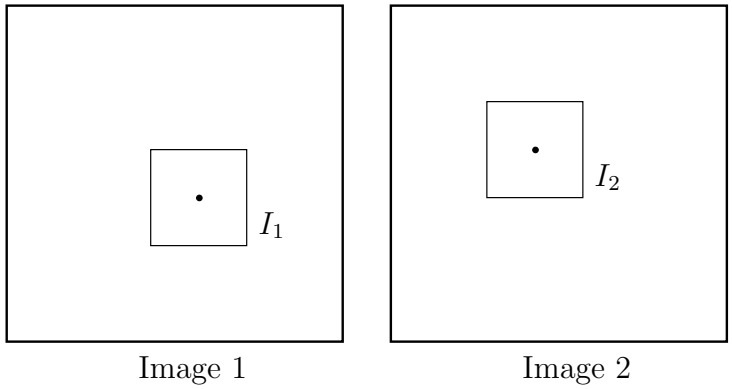

Figure 1. I_1 and I_2 are neighborhoods surrounding interest points in the two images.

by a large translation vector, and the images typically exhibit significant parallax. Small baseline stereo correspondence often arises in video sequences, where the observed change between each successive frame is very small.

The correspondence problem also arises in object recognition. In object recognition, the correspondence problem is much harder because the input is an image of any of a number of different instances of objects that must be classified as belonging to the same category. For instance, one might want to recognize an object as being (or not being) a chalkboard eraser from multiple viewing angles. The correspondence problem then is to find matching points between two instances that are not physically identical. The physical structure among instances of chalkboard erasers is similar (box-like, with reasonable corners), but is different in appearance (consider a new eraser in comparison to a worn, used one). In object recognition, there are so many ways an object can be deformed or distorted that the problem becomes exceedingly difficult.

SFM, on the other hand, provides a more constrained setting. In SFM we can intertwine the process of recovering epipolar geometry and feature extraction by using our knowledge of the relationships between two views of a rigid scene.

Before we can proceed, we need a point descriptor and a measure of similarity. The simplest descriptor of some point in an image is a small window, or neighborhood, surrounding the point of interest.

Example: Neighborhood as a simple point descriptor.

- See figure 1.
- First, run an interest point detector. Consider one interest point in each image.
- Now that we have two candidate features, we need a characterization of the interest points.
- Consider the patch of pixels, I_1, surrounding the interest point in image 1 and the corresponding patch I_2 in image 2.
- I_1 and I_2 can be used to describe each point.

Now that we have a descriptor for each interest point, we need a way of measuring the similarity between two interest points. Methods for measuring similarity between two neighborhoods include:

- Cross Correlation - Dot product of the vectorized neighborhoods.
- Sum of Squared Differences (SSD). A problem with SSD and cross correlation is that they are not invariant to illumination transformations.
- Normalized Cross Correlation (NCC)[1] - This measure is invariant to illumination transformations of the form $\alpha I + \beta$. The scalars α and β represent contrast and brightness modifications, respectively.

Normalized cross correlation is defined as

$$NCC(I_1, I_2) = \frac{\sum_{\boldsymbol{x}}(I_1(\boldsymbol{x}) - \bar{I}_1)(I_2(\boldsymbol{x}) - \bar{I}_2)}{\sqrt{\sum_{\boldsymbol{x}}(I_1(\boldsymbol{x}) - \bar{I}_1)^2 \sum_{\boldsymbol{x}}(I_2(\boldsymbol{x}) - \bar{I}_2)^2}}$$

where

$$\bar{I}_1 = \frac{1}{N}\sum_{\boldsymbol{x}} I_1(\boldsymbol{x}), \quad \bar{I}_2 = \frac{1}{N}\sum_{\boldsymbol{x}} I_2(\boldsymbol{x})$$

are the means of windows I_1 and I_2 respectively.

NCC takes on values in $[-1, 1]$ (1 being most similar, -1 being least similar) and is invariant to brightness shifts and/or contrast scaling. By precomputing the mean of the intensity of the patch of interest, we can eliminate the effect of the brightness. NCC effectively takes the square patches of interest and:

- vectorizes them (using the stack operation),
- subtracts their respective means,
- sums over the inner product of the two vectorized patches of interest,
- and normalizes the quantity with the product of their standard deviations.

This is a convenient method for measuring similarity. Unfortunately, NCC fails when the either window contains a constant image intensity. The

[1]NCC in Matlab is implemented as `normxcorr2.m`.

result is a divide by zero and "NCC blows up". However, in our setting, this is not a problem because the only points of interest that NCC will consider should be windows containing a corner (from the feature detector); the feature detector will only return neighborhoods of rank 2.

There's a certain spatial model of transformation that is implicit here; that is the translational model, with 2 degrees of freedom:

$$\boldsymbol{x}' \longrightarrow \boldsymbol{x} + \boldsymbol{t}$$

where $\boldsymbol{x} = (x, y)^\top$ and $\boldsymbol{t} = (t_x, t_y)^\top$. The translational model is implicit in what we're doing with square patches being moved around the image. Another possibility is the affine model, with 6 degrees of freedom:

$$\boldsymbol{x}' \longrightarrow A\boldsymbol{x} + \boldsymbol{t}$$

where $A \in GL(2)$. The affine model allows you to search over rotations, scalings, and shears of a patch, but involves a six parameter search, which is more costly.

14.3. RANSAC (RANdom SAmple Consensus)

In general there are all kinds of photometric and geometric transformations that can occur between two views of a scene. This means normalized cross correlation will sometimes generate spurious correspondences. To robustly fit a model to the correspondences, we need to overcome the effect of these outliers.

Random sample consensus (RANSAC) provides a general technique for model fitting in the presence of outliers. The following example demonstrates the general methodology of RANSAC.

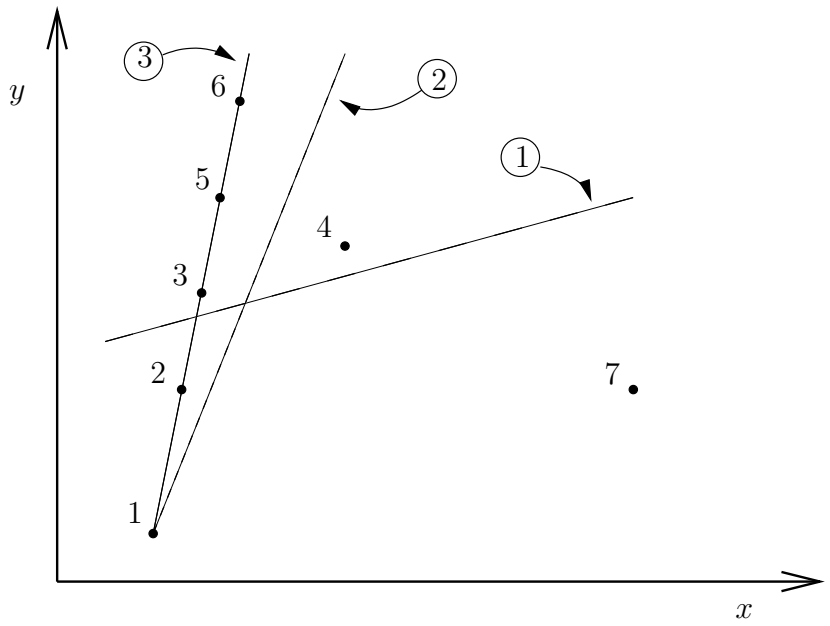

Figure 2. Three possible line fits from data points $\{1, ..., 7\}$. (1) Least squares solution. (2) Least squares with one outlier, 7, removed. (3) RANSAC based on groups of two points. [Torr & Murray]

Example: Line fitting.
See figure 2. Here we illustrate three different types of fit.

(a) The first is classical least squares. Suppose we have reason to believe that these points in the plane should fall on some line. The problem is that there are some outliers in the dataset and least squares is not very tolerant of outliers. If we perform least squares regression, we get line 1 which is obviously corrupted by the outliers.

(b) One reasonable alternative is to try least squares while "allowing" (ignoring) one outlier. This approach completes a fit to the data and then throws out the single most offensive datapoint to the fit. This approach nets us line 2.

(c) RANSAC takes this notion of robust fitting to the next level. Line 3 is the line generated by RANSAC running on permutations of two points. Start by choosing the model that will describe the data. In our case, we need two points to define a line. RANSAC iterates over all pairs of points and performs a fit. Then, for each of these lines generated by the fit, count the inliers that fall inside some ϵ band around the line. A line with a number of inliers above some threshold is considered a good fit. In our example, examining line 3, generated by say points 2 and 5, five points in total fall inside the ϵ band. For the pair of points 4 and 7, only three points fall inside the ϵ band. For points 4 and 6, only those two fall inside the ϵ band. Therefore line 3 is the optimal fit because it is supported by the most points.

In general, applications of RANSAC consist of the following steps:

(a) Choose a model.
(b) Determine the *minimal* number of points needed to specify the model.
(c) Define a threshold on the inlier count.
(d) Fit the model to a randomly selected minimal subset.
(e) Apply the transformation to the complete set of points and count inliers.
(f) If the number of inliers exceeds the threshold, flag the fit as good and stop.
(g) Otherwise repeat steps d to f.

RANSAC does model parameter estimation and outlier detection simultaneously and is also very tolerant of a lot of error in the dataset. The motion models where RANSAC is most commonly used are homography estimation and fundamental matrix estimation.

See figure 4. For homographies, the minimal number of required correspondences is 4. To measure the error, apply the estimated homography and see if each mapped point from view 1 is within some radius (i.e., within an "epsilon ball") of a point in view 2. For fundamental matrix estimation, the minimal number of points is 8 using the eight point algorithm; a 7 point alternative is also possible. Given a fundamental matrix, the error measure between two correspondences can be determined by the distance between a point and the epipolar line defined by its correspondence. Alternatively, the Sampson distance can be used.

Figure 5 outlines homography estimation using NCC and RANSAC. MaSKS Algorithm 11.4 apples RANSAC to Fundamental matrix estimation.

Example: Two images of window and a tree.

- See figure 3.
- Applying a corner detector to the two images; some points of interest will lie on the corners of the windows, for example.
- The result of applying NCC to the neighborhoods of interest in the two images is to pair the points of interest between them (with the caveat that they might be paired incorrectly).

14.4. Automated Image Mosaicing

Image mosaicing is the process of taking two or more images and stitching them together to form a panorama. In the case of a purely rotating camera, a homography defines a mapping between two views. This homography can be determined by using the techniques described earlier in this lecture; namely:

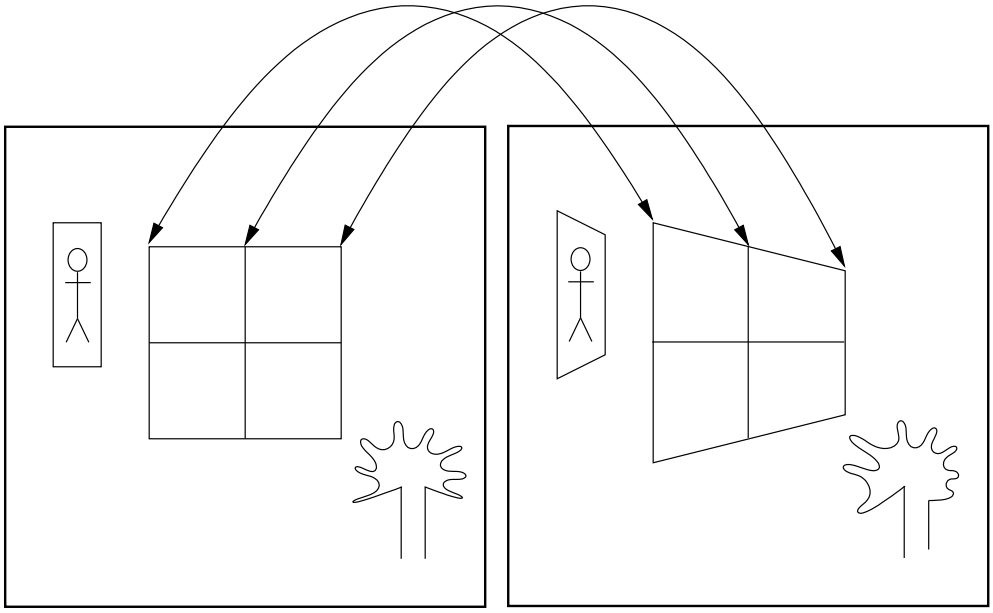

Figure 3. Points mapped by a planar homography.

	Correspondences	Error Measure
H	4 pts	"epsilon ball"
F	7 or 8 pts	Sampson Distance

Figure 4. Summary of the minimal number of points and error measure for homography and fundamental matrix estimation.

(a) Find a set of interest points in each image
(b) Use NCC to determine correspondences
(c) Use RANSAC to fit H and label outlier correspondences
(d) Estimate the homography using the four point algorithm on the complete set of inlier correspondences

Once the homography has been estimated, map all points in the second image onto the first image (called the reference image).

See figure 6. One potential problem with mapping images onto each other is that errors in the mapping can lead to artifacts in the overlap areas of the composite image. This can be ameliorated by a variety of blending techniques. J. Davis [CVPR 1998] proposes an interesting method that finds a minimal cost boundary between the reference image and the mapped image.

For mosaicing multiple images, a single reference view can be selected and images can be mapped onto the reference image pairwise. For cases

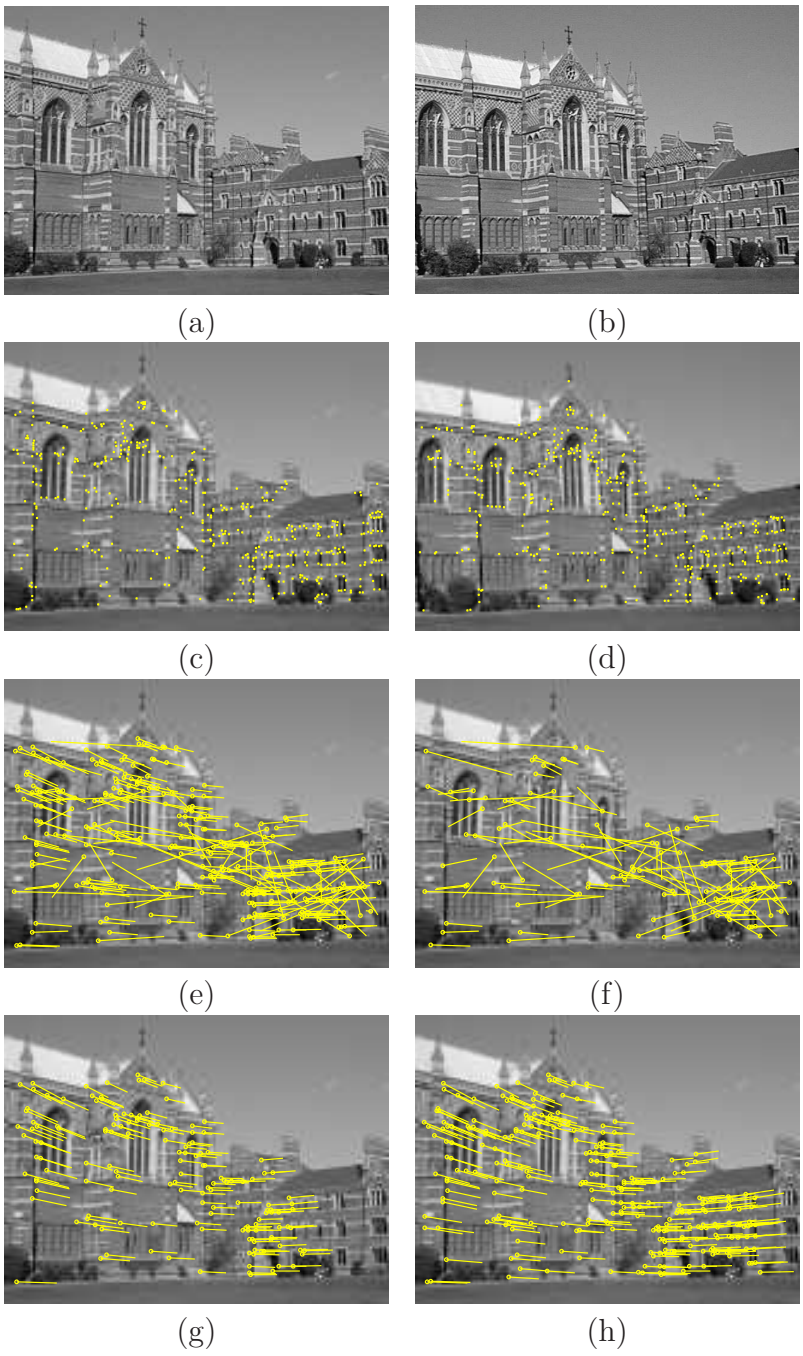

Figure 5. Homography estimation using NCC and RANSAC (from H&Z). (a) Image 1. (b) Image 2. (c) and (d) Interest points returned by a corner detector. (e) Set of correspondences found by NCC. (f) Set of outliers determined by RANSAC. (g) Set of inliers determined by RANSAC. (h) Refined correspondences based on inliers found by RANSAC.

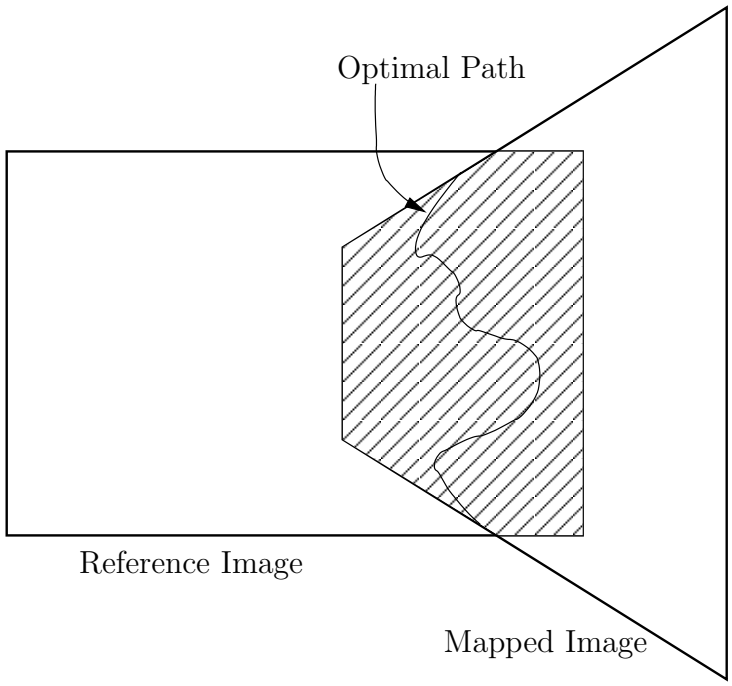

Figure 6. Two images stitched together. The optimal path is the decision boundary indicating which image to use in the overlapping region in order to minimize the visibility of the seam. It can be chosen to minimize the integrated squared difference between the overlapping mapped images within a strip around the cut.

where more than two images overlap, a non-linear technique called *bundle adjustment* can be used to fit all the views optimally.