CSE 252C: Computer Vision III

Lecturer: Serge Belongie
Scribes: Andrew Rabinovich and Vincent Rabaud
Edited by: Catherine Wah

LECTURE 10
The Correspondence Problem

10.1. Introduction

In the last lecture, we assumed an “oracle” gave us the correspondences that we used for estimating planar transformations (affine, TPS). How do we get these, assuming no oracle is available?

This lecture, we’ll look at four popular methods:

- Scott & Longuet-Higgins (SLH)
- Hungarian
- Softassign
- RANSAC (related to geometric hashing, pose clustering, Hough Transform)

These methods are widely used in two main areas: multiview geometry (Structure From Motion) and object recognition. The correspondence problem can be classified as:

- sparse v.s. dense
- narrow baseline v.s. wide baseline

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

September 20, 2009

In object recognition, we are primarily interested in sparse and wide baseline. Compared to SFM, object recognition is different in that the image being compared may not be different views of the same objection; it is more common to have different instances of the same object category. Also, note that the baseline can't be too wide, or self-occlusion takes over. Most approaches use multiple 2D views to cover the viewing sphere adequately.

Often the goal is to use the sparse correspondence to get a foothold on a coordinate transform that will allow dense correspondence (explicitly or implicitly) – we'll see that RANSAC has this spirit.

Several algorithm exist to find correspondences between images. Most approaches for estimating correspondences require a cost function over putative point-to-point matches. In practice, these can come from methods such as SIFT or MSER. To motivate the basic idea of solving a correspondence problem, we'll consider a very simple example that depends only on distances between points.

10.2. Scott & Longuet-Higgins (SLH)

One algorithm that makes very few assumptions is Scott & Longuet-Higgins (1991). It is an instance of a sparse correspondence algorithm. The reason it works is a bit mysterious, but it will become clearer later in the class when we discuss grouping and segmentation.

The input to this algorithm is two sets of points: $\boldsymbol{p}_i, i = 1, 2, \ldots, m$ and $\boldsymbol{p}_j, j = 1, 2, \ldots, n$. Assume $m \leq n$ w.l.o.g. Think of the two point sets as lying in the same place. Now, construct the Gaussian weighted distance matrix G:

$$G_{ij} = e^{-r_{ij}^2/2\sigma^2},\ r_{ij}^2 = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2. \tag{10.1}$$

G represnts the proximities between the two point sets, and σ controls the notion of "close" or "far away."

Now comes the mysterious part. Compute the SVD of G:

$$G = U\Sigma V^\top,$$

and set $\Sigma = I$ to make a new matrix P (Figure **??**):

$$P = UV^\top.$$

SLH showed that P is the orthogonal matrix that "correlates" best with G in the sense of maximizing the inner product:

$$\sum_i \sum_j P_{ij} G_{ij} = \mathrm{tr}(P^\top G). \tag{10.2}$$

In numerical linear algebra, this is known as the "orthogonal Procrustes problem" (Golub & Van Loan, 1996).

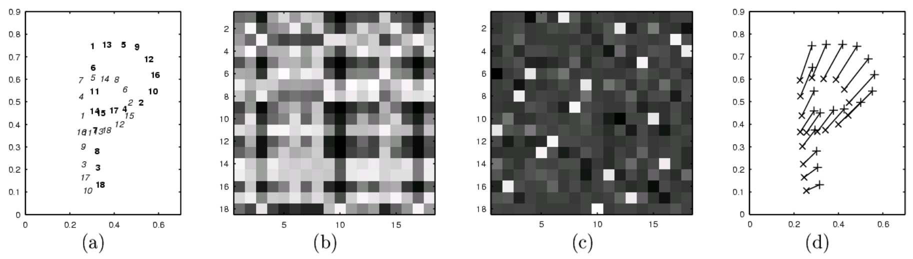

Figure 1. Longuet-Higgins algorithm. **(a)** Original points. **(b)** G. **(c)** P. **(d)** Final matching

Orthogonal matrices are one possible generalization of permutation matrices, and a permutation matrix that takes us from i in $\boldsymbol{p}_i$ to j in $\boldsymbol{p}_j$ is what we seek. In fact, the final step of SLH is to form the matrix Π where:

$$\Pi_{ij} = \begin{cases} 1, & \text{if } p_{ij} \text{ is both maximal in its row and column} \\ 0, & \text{otherwise} \end{cases} \tag{10.3}$$

This enforces the *principle of exclusion* or "uniqueness constraint" (*i.e.* one-to-one matching) while applying the *principle of proximity.* Outliers are gracefully handled in that their corresponding row or column in Π will be empty.

Pilu (*CVPR*'97) has pointed out that G_{ij} can be enhanced by including a term for the similarity of a small grayscale patch around each point; this revived some interest in this algorithm a few years back.

10.3. Hungarian Method (Kuhn-Munkres)

There are of course other ways of finding the correspondences using the information in G. Arguably the most direct way is to treat r_{ij}^2 as the cost matrix for a linear assignment problem, *i.e.* a weighted bipartite matching problem. This can be solved in $\mathrm{O}(n^3)$ using the so-called Hungarian algorithm (Kuhn-Munkres).

Definition 10.4. The *Hungarian algorithm* is a combinatorial optimization that finds the minimum cost one-to-one matching between two point sets. It is a linear assignment problem and there is a cost r_{ij} to each assignment. "Dummy nodes" are used so that a point that does not match anything can still be matched to a node. The corresponding assignment has a cost of ϵ (see Figure **??**).

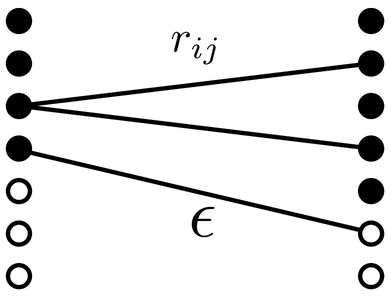

Figure 2. Linear assignment problem. Dummy nodes are represented by open circles.

10.4. Softassign

Another interesting approach is the so-called "Softassign" method of Gold & Rangarajan (*PAMI*'96), which is related to the "invisible hand" algorithm by Kosowsky & Yuille (1994).

Like SLH, they view permutation matrices as special cases of a larger family of matrices; this time, instead of orthogonals, they use doubly stochastic matrices. A *doubly stochastic matrix* has rows and colums that sum to 1, which seems to be a more appealing version of a "soft" permutation matrix, since its rows and columns can be regarded as probability distributions over correspondences.

The algorithm finds the "closest" doubly stochastic matrix to G and applies *Sinkhorn balancing*, or alternating the steps of row and column normalization until convergence, with *annealing* (dropping the value of σ), making the matrix look more like a permutation matrix. Outliers are handled by using slack variables.

10.5. RANSAC

Random Sample Consensus (RANSAC) (Fischler & Bolles, 1981) is a technique from robust statistics. When applied to point matching, it combines stereo matching with epipolar (or homography) constraints.

RANSAC is interesting because it makes use of the *minimum* number of correspondences needed to find relation between images, but it does so many times. An example of RANSAC being used is with robust line fitting (Figure **??**). It requires:

- a transformation model
- the minimum number of correspondences needed to estimate this transformation.

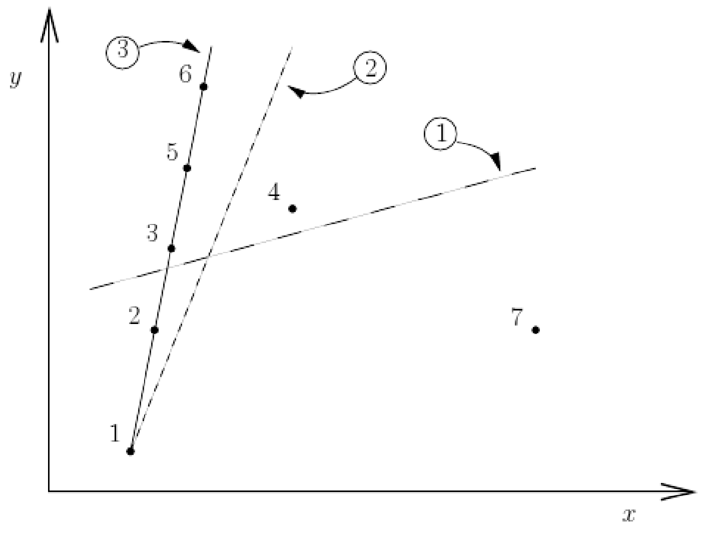

Figure 3. Three possible line fits for data points {1,...,7}. (1) Regular least squares. (2) Regular least squares with one outlier, 7, removed. (3) RANSAC (Torr & Murray, 1997).

10.6. Dense matching

Now let's turn our attention to the dense matching case. Assume now that we have successfully established the epipolar geometry and now the images are rectified so that the correspondence problem is reduced to a 1D search along corresponding scanlines.

In the 1970s, a number of scanline matching algorithms were developed based on dynamic programming. The basic idea is to set up a cost matrix between all pixels of two scanlines. An arc joins two nodes (i, i') and (j, j') when intervals (i, j) and (i', j') match each other, and we can find the minimum cost path through the cost matrix [1].

[1]See Ohta & Kanade or Baker & Binford for more details.

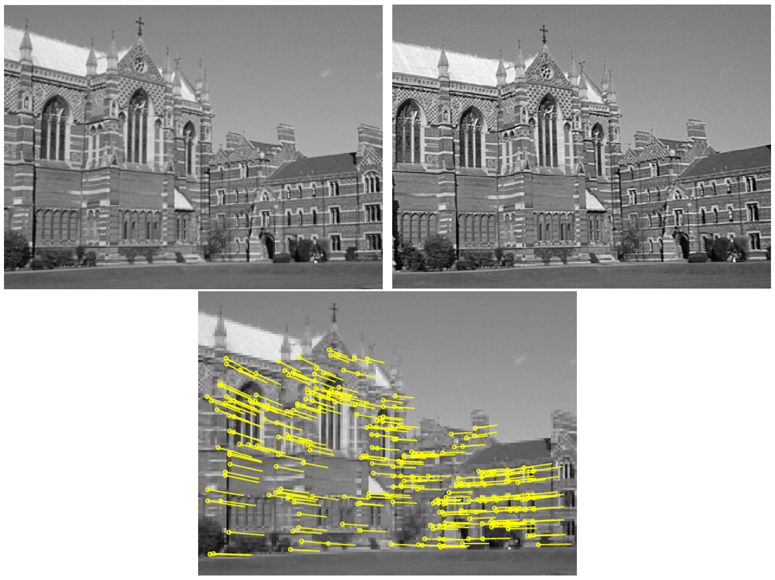

Figure 4. Example of RANSAC used in Structure From Motion. The choice of transformation is a *homography* and requires 4 correspondences (8 parameters and 2 per correspondence). The top images are two views of the same scene. The bottom image contains the correspondences after applying RANSAC.