CSE 252B: Computer Vision II

Lecturer: Serge Belongie
Scribes: Manmohan Chandraker, Sunny Chow

LECTURE 11
Optimal Pose & Structure

11.1. Implications of noise in measurements

With all the methods and techniques that we have learned so far, we made the assumption that the necessary point correspondences were both known and free of noise. Unfortunately, actual measured point correspondences do possess some amount of noise, and the noise propagates into the results of our linear methods (calculating E, F, $\boldsymbol{\lambda}$, etc.). For example, in the case of the 8-point algorithm, we find that our solution $\boldsymbol{E}^s$ to the null-space problem ($\chi \boldsymbol{E}^s = \mathbf{0}$) is only an approximation of the exact solution. In geometrical terms, given a pair of point correspondences from two different image planes, we find that rays from the optical centers of each camera frame through the imaged point on the respective image planes do not intersect exactly (see Figure 1). Noise, however, does not render our linear methods useless; even in the presence of noise, our linear methods provide pretty good results. These linear methods can serve as an initialization step, from which we can further refine the results by taking into account the noise present in our measurements.

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

May 23, 2006

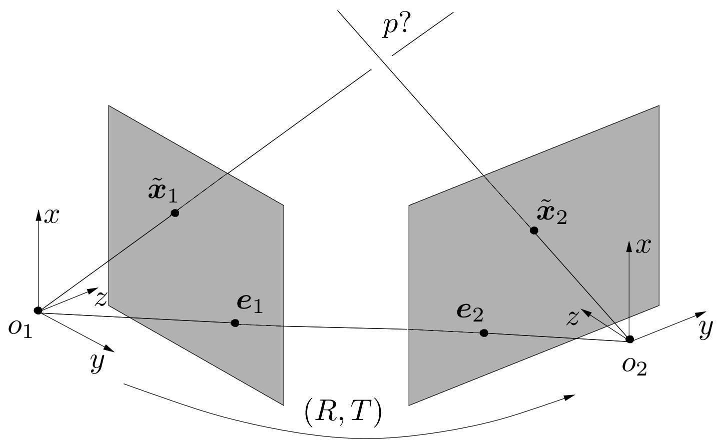

Figure 1. Nonintersection of rays through noisy images of a point.

11.2. Noise Model

We define our noise model by:

$$\tilde{\boldsymbol{x}}_1^j = \boldsymbol{x}_1^j + \boldsymbol{w}_1^j \qquad \tilde{\boldsymbol{x}}_2^j = \boldsymbol{x}_2^j + \boldsymbol{w}_2^j \qquad j = 1, 2, \ldots, n \tag{11.1}$$

where $\boldsymbol{x}_1^j, \boldsymbol{x}_2^j$ are the ideal (platonic) points, $\tilde{\boldsymbol{x}}_1^j, \tilde{\boldsymbol{x}}_2^j$ are the measured (noisy) points, and $\boldsymbol{w}_1^j = (w_{11}^j, w_{12}^j, 0)^\top, \boldsymbol{w}_2^j = (w_{21}^j, w_{22}^j, 0)^\top$ represents the error (or jitter), distributed according to some pdf (probability distribution function). Note that this noise model does not account for incorrect correspondences.

11.3. Calibrated 3D Reconstruction

In the case of calibrated 3D reconstruction, noise-free corresponding points exactly satisfy the epipolar constraint:

$$\boldsymbol{x}_2^j \widehat{T} R \boldsymbol{x}_1^j = 0 \tag{11.2}$$

When noise is present, this does not hold. The optimization problem is that we want to find R and $\boldsymbol{T}$ along with the "corrected" coordinates that satisfy the epipolar constraint exactly. The problem may seem underconstrained, but we know that there is some structure in the noise, e.g., a pdf with a peak at the ideal image point, and this limits the magnitude of the displacements.

There is no universally accepted objective function we can use to solve for $(\boldsymbol{x}, R, \boldsymbol{T})$. Instead, different cost functions exist that minimize different discrepancies. Two commonly used cost functions are ML (Maximum Likelihood) and MAP (Maximum a Posteriori).

11.3.1. ML (Maximum Likelihood)

The ML solution is given by

$$(\boldsymbol{x}^*, R^*, \boldsymbol{T}^*) = \text{argmax } \phi_{ML}(\boldsymbol{x}, R, \boldsymbol{T}) = \sum_{i,j} \log p((\tilde{\boldsymbol{x}}_i^j - \boldsymbol{x}_i^j)|\boldsymbol{x}, R, \boldsymbol{T}) \tag{11.3}$$

The ML function defines the likelihood function as $p(\boldsymbol{w}|\boldsymbol{x}, R, \boldsymbol{T})$ and we seek to maximize it with respect to the unknown parameters. We assume that the likelihood belongs to a family of density functions (for example, Gaussian) since estimating it from first principles is too difficult.

11.3.2. MAP (Maximum a Posteriori)

The main difference between MAP and ML is that MAP incorporates a Bayesian "prior" probability in its calculations. The prior probability is given by $p(\boldsymbol{x}, R, \boldsymbol{T})$; we can exploit it to indicate that certain values of the parameters are more likely than others. The MAP solution is given by

$$(\boldsymbol{x}^*, R^*, \boldsymbol{T}^*) = \text{argmax } \phi_{MAP}(\boldsymbol{x}, R, \boldsymbol{T}) = p(\boldsymbol{x}, R, \boldsymbol{T}|\{\tilde{\boldsymbol{x}}_i^j\}) \tag{11.4}$$

The prior probability is exhibited by applying Bayes' Theorem to ϕ_{MAP}:

$$p(\boldsymbol{x}, R, \boldsymbol{T}|\{\tilde{\boldsymbol{x}}_i^j\}) \propto p(\{\tilde{\boldsymbol{x}}_i^j\}|\boldsymbol{x}, R, \boldsymbol{T})p(\boldsymbol{x}, R, \boldsymbol{T})$$

11.4. Simple Cost Function: Reprojection Error

As a concrete example, we will consider the following simple cost function to find $(\boldsymbol{x}^*, R^*, \boldsymbol{T}^*) = \text{argmin } \phi(\boldsymbol{x}, R, \boldsymbol{T})$ with

$$\phi(\boldsymbol{x}, R, \boldsymbol{T}) = \sum_{j=1}^{n} \sum_{i=1}^{2} ||\tilde{\boldsymbol{x}}_i^j - \boldsymbol{x}_i^j||^2 \tag{11.5}$$

and the following constraints:

$$\boldsymbol{x}_2^{j\top} \hat{T} R \boldsymbol{x}_1^j = 0, \quad \boldsymbol{x}_1^{j\top} \boldsymbol{e}_3 = 1, \quad \boldsymbol{x}_2^{j\top} \boldsymbol{e}_3 = 1, \quad j = 1, 2, \ldots, n \tag{11.6}$$

where $\boldsymbol{e}_3 = (0, 0, 1)^\top$. This cost function is the "reprojection error" and it is a special case of the ML cost function. We can convert the above to an unconstrained minimization problem using Lagrange multipliers (MaSKS Appendix 5.A). The idea here is that we will augment the cost function by attaching some weight to every constraint. Using an initial approximation of $\boldsymbol{x}$ we estimate R and $\boldsymbol{T}$, and then use them to update the estimate for $\boldsymbol{x}$ in an iterative fashion.

This iterative approach, however, is computationally expensive since it requires us to estimate the 3D coordinates and then project them onto the image planes to get an updated estimate. As an alternative, we would like to have a method to estimate errors using only the measured image coordinates (eliminating the need for explicit 3D reconstruction); this means that we want a cost function that depends ONLY on $\tilde{\boldsymbol{x}}$. For this purpose MaSKS proposes to use the distance from the epipolar line:

$$\phi(R,\boldsymbol{T}) = \sum_{j=1}^{n} \frac{(\tilde{\boldsymbol{x}}_2^{j\top}\widehat{T}R\tilde{\boldsymbol{x}}_1^{j\top})^2}{\|\widehat{e}_3\widehat{T}R\tilde{\boldsymbol{x}}_1^j\|^2} + \frac{(\tilde{\boldsymbol{x}}_2^{j\top}\widehat{T}R\tilde{\boldsymbol{x}}_1^{j\top})^2}{\|\tilde{\boldsymbol{x}}_2^{j\top}\widehat{T}R\widehat{e}_3^{\top}\|^2} \tag{11.7}$$

As promised, the equation depends only on $\tilde{\boldsymbol{x}}$. The numerators in the cost function give the distance to the epipolar line times a scale factor, and the denominator provides the normalization factor to make the cost function equal the distance on the image plane. This is illustrated in Figure 2.

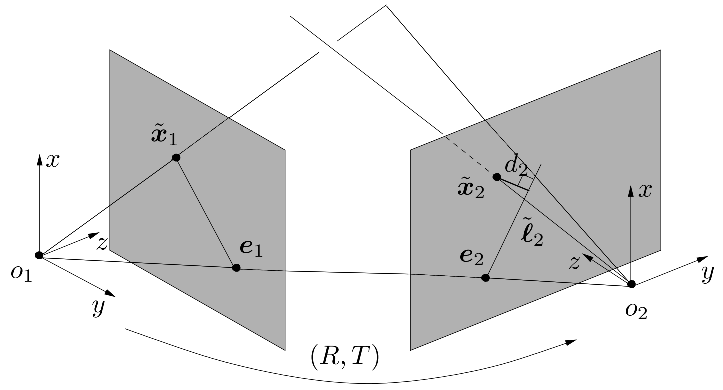
$\tilde{\boldsymbol{x}}_1$
x
z
o_1
y
$\boldsymbol{e}_1$
d_2
$\tilde{\boldsymbol{x}}_2$
$\tilde{\boldsymbol{\ell}}_2$
$\boldsymbol{e}_2$
x
z
y
o_2
(R,T)

Figure 2. Reprojection error (d_2) shown in image plane 2.

Once we estimate the camera pose $(R^*,\boldsymbol{T}^*)$ by minimizing Equation (11.7), we can estimate $(\boldsymbol{x}_1^*,\boldsymbol{x}_2^*)$ that satisfy the epipolar constraint (Equation 11.6) and minimize the reprojection error:

$$\phi(\boldsymbol{x}) = \|\tilde{\boldsymbol{x}}_1 - \boldsymbol{x}_1\|^2 + \|\tilde{\boldsymbol{x}}_2 - \boldsymbol{x}_2\|^2 \tag{11.8}$$

This is called optimal triangulation. If we fixed $(R,\boldsymbol{T})$ to minimize $\phi(\boldsymbol{x})$, we find that the baseline (and epipoles) are also fixed on these constraints. This leaves only the "tilt" of the epipolar plane to be defined, and we find that $\phi(\boldsymbol{x})$ is dependent only on this tilt θ (See Figure 3). Given a value of θ then, we can solve for distances between a point and its corresponding epipolar line.

We can now define an algorithm to find $(\boldsymbol{x},R,\boldsymbol{T})$ that minimizes the reprojection error.

(a) First obtain the measured coordinates of n corresponding points in the two views.
(b) Find a linear estimate of $(R,\boldsymbol{T})$ (using 8 point algorithm for example). Refine the estimate by minimizing Equation (11.7).

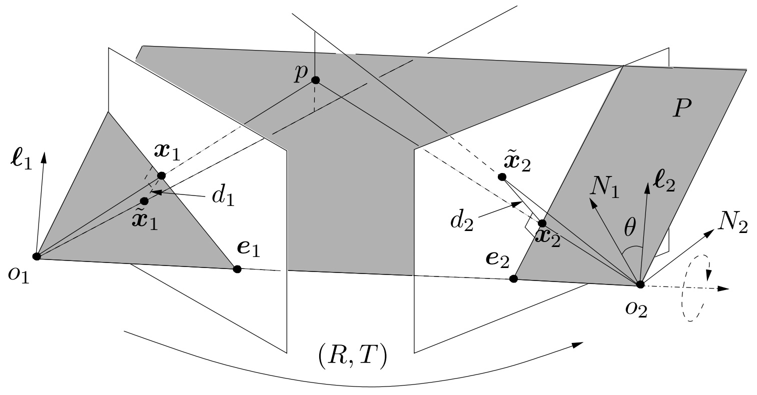

Figure 3. Estimating $\boldsymbol{x}_1^*, \boldsymbol{x}_2^*$ while varying θ

(c) Find the tilt (θ) of the plane to minimize d_1 and d_2 via Equation (11.8).
(d) Rinse and repeat.

11.5. Optimal Pose and Structure – Uncalibrated Case

We have seen earlier that the DLT algorithm minimizes a residual error of the form $\|\chi \boldsymbol{F}^s\|$. While very simple from an implementation point of view, this algebraic quantity is not geometrically meaningful. So, the minimum norm solution may differ from what we expect intuitively, i.e., a solution that minimizes error on the image plane.

We will now have a look at a cost function commonly used in uncalibrated SFM that is easy to compute but corresponds to the geometric error when the error is small. This cost function is called the *Sampson error*, after P. D. Sampson who introduced this approximation for conic fitting to sparse data.

11.5.1. Derivation of the Sampson Distance

Recall that the noise model we are using is additive. The *measured* points $\tilde{\boldsymbol{x}}_i^j$ are related to the *platonic* points $\boldsymbol{x}_i^j$ by an error term $\boldsymbol{w}_i^j$ drawn from a specified pdf:

$$\tilde{\boldsymbol{x}}_i^j = \boldsymbol{x}_i^j + \boldsymbol{w}_i^j \qquad \text{for} \quad i = 1, 2 \qquad j = 1, \ldots, n$$

Collect together, for each putative correspondence $\tilde{\boldsymbol{x}}_1 \leftrightarrow \tilde{\boldsymbol{x}}_2$, the following quantities:

- a *measured vector*, $\tilde{\boldsymbol{X}} = (\tilde{\boldsymbol{x}}_1^\top, \tilde{\boldsymbol{x}}_2^\top)^\top \in \mathbb{R}^4$
- a *corrected vector*, $\boldsymbol{X} = (\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top)^\top \in \mathbb{R}^4$

- a *noise vector*, $\boldsymbol{w} = (w_{11}, w_{12}, w_{21}, w_{22})^\top \in \mathbb{R}^4$.

Given $\tilde{\boldsymbol{X}}_j \in \mathbb{R}^4$, the objective is to find an element on the so-called "variety" $\mathcal{V}_F$ defined by $\boldsymbol{x}_2^{j\top} F \boldsymbol{x}_1^j = 0$ that most nearly passes through $\tilde{\boldsymbol{X}}_j$. To this end, we write the epipolar constraint as a cost function:

$$C_F(\tilde{\boldsymbol{X}}^j) = \tilde{\boldsymbol{x}}_1^{j\top} F \tilde{\boldsymbol{x}}_2^j$$

Expanding the cost function into its first-order Taylor series, we get

$$C_F(\tilde{\boldsymbol{X}}^j + \boldsymbol{\delta}) = C_F(\tilde{\boldsymbol{X}}^j) + \frac{\partial C_F}{\partial \tilde{\boldsymbol{X}}^j}\boldsymbol{\delta} \tag{11.9}$$

Here we have $\boldsymbol{\delta} = \boldsymbol{X}^j - \tilde{\boldsymbol{X}}^j$ and $\boldsymbol{X}^j$ is desired to lie on the variety $\mathcal{V}_F$ so that the epipolar constraint is satisfied; that is, we wish to enforce $C_F(\boldsymbol{X}^j) = C_F(\tilde{\boldsymbol{X}}^j + \boldsymbol{\delta}) = 0$. Denoting the Jacobian above by $\boldsymbol{J}$ and the associated cost by $\boldsymbol{\epsilon}$, Equation (11.9) can be re-written as

$$\boldsymbol{J}\boldsymbol{\delta} = -\boldsymbol{\epsilon} \tag{11.10}$$

Note that in this case, $\boldsymbol{J}$ is a 1×4 row vector.[1] Thus, we are left with a minimization problem:

- Find the vector $\boldsymbol{\delta}$ that minimizes $\|\boldsymbol{\delta}\|$ subject to $\boldsymbol{J}\boldsymbol{\delta} = -\boldsymbol{\epsilon}$.

The solution using Lagrange Multipliers is straightforward (Addendum A) and has the form

$$\boldsymbol{\delta} = -\boldsymbol{J}^\top(\boldsymbol{J}\boldsymbol{J}^\top)^{-1}\boldsymbol{\epsilon} \tag{11.11}$$

This is the Sampson error, and its norm is given by

$$\|\boldsymbol{\delta}\|^2 = \boldsymbol{\delta}^\top\boldsymbol{\delta} = \boldsymbol{\epsilon}^\top(\boldsymbol{J}\boldsymbol{J}^\top)^{-1}\boldsymbol{\epsilon} \tag{11.12}$$

Writing $C_F(\tilde{\boldsymbol{X}}^j)$ explicitly as $\tilde{\boldsymbol{x}}_2^{j\top} F \tilde{\boldsymbol{x}}_1^j$ and the Jacobian as

$$\boldsymbol{J} = \left(\frac{\partial C_F}{\partial \tilde{\boldsymbol{x}}_1^j}, \frac{\partial C_F}{\partial \tilde{\boldsymbol{x}}_2^j}\right)$$

We perform the requisite differentiations (see Addendum B) to obtain the row vector

$$\boldsymbol{J} = (F^\top \tilde{\boldsymbol{x}}_2^j,\ F\tilde{\boldsymbol{x}}_1^j) \tag{11.13}$$

Thus $\boldsymbol{J}\boldsymbol{J}^\top$ is a scalar given by

$$\boldsymbol{J}\boldsymbol{J}^\top = (F\tilde{\boldsymbol{x}}_1^j)_1^2 + (F\tilde{\boldsymbol{x}}_1^j)_2^2 + (F^\top\tilde{\boldsymbol{x}}_2^j)_1^2 + (F^\top\tilde{\boldsymbol{x}}_2^j)_2^2 \tag{11.14}$$

Here, $(\cdot)_k$ represents the k-th component of a vector. The scalar $\boldsymbol{\epsilon}$ is given by $\tilde{\boldsymbol{x}}_2^{j\top} F \tilde{\boldsymbol{x}}_1^j$.

[1]My apologies for breaking with protocol here; I had promised that all vectors would be column vectors, and $\boldsymbol{J}$ is a row vector here.

Substituting the expressions for $\boldsymbol{\epsilon}$ and $\boldsymbol{J}$ in Equation (11.11) and summing over all n correspondences, we get the cost function[2]

$$\phi(F) = \sum_{j=1}^{n} \frac{(\tilde{\boldsymbol{x}}_2^{j\top} F \tilde{\boldsymbol{x}}_1^j)^2}{(F\tilde{\boldsymbol{x}}_1^j)_1^2 + (F\tilde{\boldsymbol{x}}_1^j)_2^2 + (F^\top\tilde{\boldsymbol{x}}_2^j)_1^2 + (F^\top\tilde{\boldsymbol{x}}_2^j)_2^2} \tag{11.15}$$

This gives a first-order approximation of the geometric error provided the conditions for Taylor Series expansion are satisfied, that is, if the truncation error given by the higher (second onwards) order terms is small.

11.5.2. Utility of Sampson Cost Function for Minimization

The cost function in Equation (11.15) can be used in a Maximum Likelihood estimation of the fundamental matrix, given $n \geq 8$ image correspondences. Note that this cost function involves only the parameters of F, so the minimization of the reprojection error is reduced to a 7-dof problem from a $7+3n$ dof problem (7 for the F-matrix and 3 for each 3D point $\tilde{\boldsymbol{X}}^j$, $j = 1, \ldots, n$).

Common optimization routines used in practice for this problem are Gauss-Newton and Levenberg-Marquardt. The last, in particular, is the same as Gauss-Newton applied to a least-squares problem and has been extensively used in computer vision literature, especially in image mosaicing and multiple-view reconstruction.

Addendum A - Lagrange multipliers

Our goal is to find the vector $\boldsymbol{\delta}$ that minimizes $\|\boldsymbol{\delta}\|$ subject to $\boldsymbol{J}^\top\boldsymbol{\delta} = -\boldsymbol{\epsilon}$. Introducing a vector $\boldsymbol{\lambda}$ of Lagrange multipliers, the objective function to be minimized becomes $\boldsymbol{\delta}^\top\boldsymbol{\delta} - 2\boldsymbol{\lambda}^\top(\boldsymbol{J}\boldsymbol{\delta} + \boldsymbol{\epsilon})$, the factor of 2 has been introduced for algebraic simplification. Differentiate with respect to $\boldsymbol{\delta}$ to obtain

$$\boldsymbol{\delta} = \boldsymbol{J}^\top\boldsymbol{\lambda} \tag{11.16}$$

Substituting in the original constraint equation $\boldsymbol{J}\boldsymbol{\delta} = -\boldsymbol{\epsilon}$,

$$\boldsymbol{J}\boldsymbol{J}^\top\boldsymbol{\lambda} = -\boldsymbol{\epsilon}$$

Solving for $\boldsymbol{\lambda}$ and substituting in (11.16), we get

$$\boldsymbol{\delta} = -\boldsymbol{J}^\top(\boldsymbol{J}\boldsymbol{J}^\top)^{-1}\boldsymbol{\epsilon} \tag{11.17}$$

[2]Note: this expression, which is from H&Z, looks different than its counterpart in MaSKS; Ben Ochoa has verified (in a handwritten note) that they are in fact equivalent.

Addendum B - Vector and matrix differentiation

For vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and matrix M, we have

$$\frac{\partial}{\partial \boldsymbol{a}} \boldsymbol{a}^\top M \boldsymbol{b} = M\boldsymbol{b} \quad \text{and} \quad \frac{\partial}{\partial \boldsymbol{b}} \boldsymbol{a}^\top M \boldsymbol{b} = M^\top \boldsymbol{a}$$