The last several lectures we have looked at image properties such as local intensity gradients and how these are can be directly tied to scene structure. The remaining lectures will be concerned mostly with geometry (as opposed to radiometry i.e. what to do with image intensities). We start with camera calibration and work our way to various versions of the multiview geometry problem.

Camera calibration

To estimate geometric properties of 3D scenes, it helps to know the camera parameters, both external and internal. The problem of finding all these parameters is typically called *camera calibration*.

Recall at the end of lecture 3, we had a transformation that takes a point $(X, Y, Z, 1)$ in world coordinates to a pixel position (wx, wy, w):

$$\mathbf{P} = \mathbf{KR}[\ \mathbf{I}\ |\ -\mathbf{C}]$$

where $\mathbf{K}$, $\mathbf{R}$, and $\mathbf{I}$ are 3×3 matrices, $\mathbf{C}$ is a 3×1 vector, and so $\mathbf{P}$ is 3×4. $\mathbf{P}$ is called a *finite projective camera*. Recall that it assumes a pinhole only.

Estimating P using least squares

How could we estimate $\mathbf{P}$? The standard method assumes we have a set of known points (X_i, Y_i, Z_i) in the scene. For example, we might have a cube of known size and whose faces have a checkerboard pattern on them (also of known size). The corners of the squares on the checkerboard would define 3D points in a world coordinate system defined by a corner of the cube.

If we take a photograph of the cube, we could then (by hand) find the pixel coordinates of the known points on the checkerboard, then we would have a set of constraints relating the pixel positions (x_i, y_i) and the scene positions (X_i, Y_i, Z_i).

$$x_i = \frac{wx_i}{w} = \frac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}}$$

$$y_i = \frac{wy_i}{w} = \frac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}}$$

and so

$$x_i(P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}) = P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}$$

$$y_i(P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}) = P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}$$

We arrange these equations using a $2N \times 12$ matrix such that

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ \vdots & & & & & & & & & & & \\ \vdots & & & & & & & & & & & \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -x_NX_N & -x_NY_N & -x_NZ_N & -x_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -y_NX_N & -y_NY_N & -y_NZ_N & -y_N \end{bmatrix} \begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \\ P_{14} \\ \vdots \\ P_{31} \\ P_{32} \\ P_{33} \\ P_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$\mathbf{P}$ has 12 parameters, so if we want to solve for $\mathbf{P}$ then we need $N \geq 6$ corresponding point pairs i.e. we need 12 or more data values.

If we have exactly $N = 6$ points, then we will have a linear system with 12 equations and 12 unknowns and we can try to solve for $\mathbf{P}$. If the (x_i, y_i) really are related to the (X_i, Y_i, Z_i) according to the projection model, then there will be a $\mathbf{P}$ that solves the above system, and so $\mathbf{P}$ will lie in the null space of the matrix (and the matrix will have to be non-invertible).

In practice, howeover, the values of $(x_i, y_i, X_i, Y_i, Z_i)$ will contain some measurement error and the projection model will only be an approximation to the actual physical situation. The result is that, in practice, the 12×12 matrix formed from $N = 6$ data points will typically be invertible – i.e. it won't have a null space – and the equations might only have the trivial solution $\mathbf{P} = \mathbf{0}$. Rather than trying to solve the equations, one instead re-poses the problem in terms of least squares.

Least squares

Let's step back and consider a few general least squares problems. We have seen instances of these in the course and we will see more in the upcoming lectures. For this reason it is good to have a general picture of these problems. We look at two versions.

Version 1: Given A, minimize $\| \mathbf{Ax} \|_2$ subject to constraint on x

An example is the problem we just looked at, where $\mathbf{x}$ was our 12×1 $\mathbf{P}$ vector.

For the general problem, we are trying to minimize $\| \mathbf{Ax} \|_2^2$. This is trivial if we set $\mathbf{x}$=0 but this is typically not the solution we are interested in. Instead we want to minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}$ subject to some condition such as $\| \mathbf{x} \| = 1$.

As we saw in lecture 13, we can solve this problem using Lagrange multipliers. Say our condition is $\| \mathbf{x} \| = 1$, or equivalently $\mathbf{x}^T\mathbf{x} = 1$. Then we minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{Ax} + \lambda(\mathbf{x}^T\mathbf{x} - 1)$. Taking derivatives with respect to the $\mathbf{x}$ components gives[1]

$$\mathbf{A}^T\mathbf{Ax} + \lambda\mathbf{x} = 0$$

and setting the λ derivative to 0 just gives our $\mathbf{x}$ constraint. Together we get that the minimum is achieved when $\mathbf{x}$ is an eigenvector of $\mathbf{A}^T\mathbf{A}$.

Our goal is to minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}$. So which eigenvector gives the least value of $\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}$? Clearly $\mathbf{A}^T\mathbf{A}$ is symmetric and so all eigenvalues are non-negative, and thus we want the eigenvector with the *smallest eigenvalue*.

For the case of our camera calibration problem, $\mathbf{A}$ is the $2N \times 12$ coefficient matrix, and $\mathbf{x}$ is the matrix $\mathbf{P}$ written as a 12-vector. We solve for $\mathbf{P}$ by finding the eigenvector of $\mathbf{A}^T\mathbf{A}$ having smallest eigenvalue.

Version 2: Given A,b, minimize $\| \mathbf{Ax} - \mathbf{b} \|_2$

While we are discussing least squares solutions, let's consider an alternative problem which arises often (not just in computer vision, but in many other fields). Suppose we have an $m \times n$ matrix $\mathbf{A}$ and an m vector $\mathbf{b}$ and would like to find the vector $\mathbf{x}$ that minimizes $\| \mathbf{Ax} - \mathbf{b} \|_2$, where

[1] You should work out the details here. They are straightforward, but make sure you can do it.

the 2-subscript indicates the usual Euclidean distance (in $\Re^m$). We have seen such least squares problems in lecture 10, when we did image registration

If $\mathbf{b}$ is $\mathbf{0}$ then we have the same problem above, so let's assume $\mathbf{b} \neq \mathbf{0}$. We don't need Lagrange multipliers in this case (since the trivial solution $\mathbf{x} = 0$ is no longer a solution so we don't need to avoid it).

We want to find the $\mathbf{x}$ that minimizes

$$\| \mathbf{Ax} - \mathbf{b} \|_2^2 \;=\; (\mathbf{Ax} - \mathbf{b})^T(\mathbf{Ax} - \mathbf{b}) \;=\; \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} - 2\mathbf{b}^T\mathbf{Ax} + \mathbf{b}^T\mathbf{b}.$$

We take partial derivatives with respect to the $\mathbf{x}$ variables and set them to 0. This gives

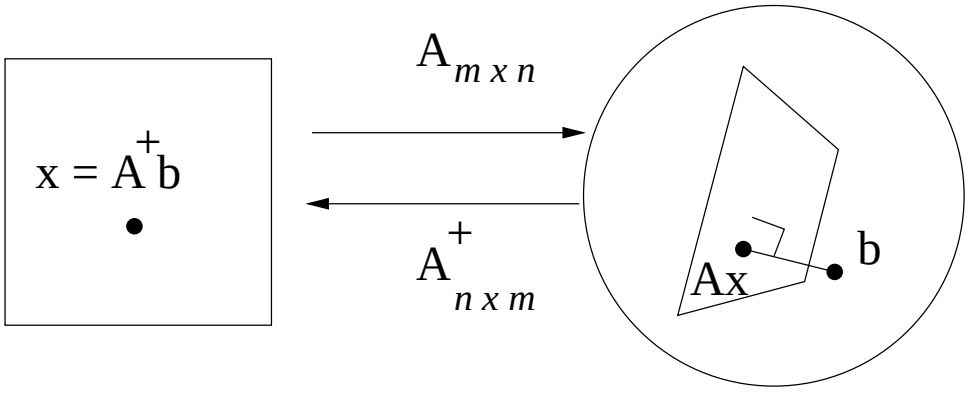
$$2\mathbf{A}^T\mathbf{Ax} - 2\mathbf{A}^T\mathbf{b} = 0.$$

or

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}. \tag{1}$$

which we can solve using basic matrix methods.

What is the geometric interpretation of this solution? We can *uniquely* write $\mathbf{b}$ as a sum of a vector in the column space of $\mathbf{A}$ and a vector in the space orthogonal to the column space of $\mathbf{A}$. To minimize $\| \mathbf{Ax} - \mathbf{b} \|_2$, we want to find the $\mathbf{x}$ such that the distance from $\mathbf{Ax}$ to $\mathbf{b}$ is as small as possible. This is done by choosing $\mathbf{x}$ such that $\mathbf{Ax}$ is the component of $\mathbf{b}$ that lies in the column space of $\mathbf{A}$. Equivalently, we want the vector $\mathbf{Ax}$ - $\mathbf{b}$ to be entirely within the space that is orthogonal to the column space of $\mathbf{A}$, and so we require $\mathbf{A}^T(\mathbf{Ax} - \mathbf{b}) = \mathbf{0}$ which is just Eq. (1)! Note that if $\mathbf{b}$ already belongs in the column space of $\mathbf{A}$ – that is, it can be represented as a linear combination of the columns of $\mathbf{A}$ – then the least squares "error" is 0 and there is an exact solution.



Pseudoinverse of A

For each $\mathbf{b}$, there is an $\mathbf{x}$ that solves our minimization problem. If $\mathbf{A}^T\mathbf{A}$ is invertible - this happens if the columns of $\mathbf{A}$ are linearly independent – our solution can be written

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}.$$

We define the *pseudoinverse* of $\mathbf{A}$ to be the $n \times m$ matrix,

$$\mathbf{A}^+ \equiv (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$$

(Note that if $\mathbf{A}$ itself is invertible – in particular, it must be a square matrix – then $\mathbf{A}^+ = \mathbf{A}^{-1}$.) Also,

$$\mathbf{A}\mathbf{A}^+ = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$$

projects any vector $\mathbf{b} \in \Re^m$ onto the column space of $\mathbf{A}$, that is, it removes from $\mathbf{b}$ the component that is perpendicular to the column space of $\mathbf{A}$. This was illustrated in the above figure, namely $\mathbf{A}\mathbf{A}^+$ projects $\mathbf{b}$ onto the column space of $\mathbf{A}$.

The pseudoinverse maps in the reverse direction of $\mathbf{A}$, namely it maps $\mathbf{b}$ in an m-D space to an n-D space and, rather than inverting, only "inverts" the component of $\mathbf{b}$ that belongs to the column space of $\mathbf{A}$, i.e. $\mathbf{A}^+\mathbf{A} = \mathbf{I}$ but $\mathbf{A}\mathbf{A}^+$ only equals $\mathbf{I}$ when $\mathbf{A}$ itself is invertible.

Factoring P into internal and external camera parameters

Getting back to our problem,... we have seen how to estimate the $\mathbf{P}$ matrix. Note that it is only estimated up to an overall scale factor[2]: $\mathbf{P}$ operates on vectors (X, Y, Z) and (x, y) written in homogeneous coordinates, so $\mathbf{P}$ is the same as $a\mathbf{P}$ for any a.

We would now like to decompose $\mathbf{P}$ into the product of a 3×3 upper triangular matrix $\mathbf{K}$ and a 3×4 matrix $\mathbf{R}[\mathbf{I}| - \mathbf{C}]$.

Step 1: normalize P

We note that the elements $(P_{31}P_{32}P_{33})$ are not all zero. (Recall from Exercise 1 Q4d that this 3-vector is normal to the projection plane. It cannot be all 0's if we indeed have a projection matrix.) The first step is to divide each element of $\mathbf{P}$ by $\parallel (P_{31}, P_{32}, P_{33}) \parallel$ so that this vector is a unit normal.

For the next steps, we consider only the left 3×3 submatrix of $\mathbf{P}$ whose 3rd row is now of length 1. Let $\tilde{\mathbf{P}}$ be the 3×3 matrix which is the first three columns of $\mathbf{P}$. We decompose $\tilde{\mathbf{P}}$ into $\mathbf{KR}$. To to this, we want to find a rotation matrix such that $\tilde{\mathbf{P}}\mathbf{R}^T = \mathbf{K}$. We will construct $\tilde{\mathbf{P}}$ by building three rotation matrices that rotate about the canonical axes.

Step 2: find R

We will construct a rotation matrix $\mathbf{R}$ in three steps.

First, we find a rotation matrix $\mathbf{R}_{Z,\theta}$ that $\tilde{\mathbf{P}}\mathbf{R}_{Z,\theta}$ has its $(3, 1)$ element equal to zero. Define

$$\mathbf{R}_{Z,\theta} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then we want

$$P_{31}\cos\theta - P_{32}\sin\theta = 0$$

and so we need

$$\theta = \arctan(P_{31}/P_{32}).$$

[2] The Lagrange multiplier technique forced the solutions to be of unit length. But if it would have achieved the same solution had used any non-unit fixed length, namely the solution is that $\mathbf{P}$ is an eigenvector of $\mathbf{A}^T\mathbf{A}$. Note that if $\mathbf{P}$ is an eigenvector than $a\mathbf{P}$ is an eigenvector for any a.

Note that a Z-rotation does not affect the 3rd column of $\tilde{\mathbf{P}}$.

Second, we find a rotation matrix

$$\mathbf{R}_{X,\beta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix}$$

such that $\mathbf{P}\mathbf{R}_{z,\theta}\mathbf{R}_{X,\beta}$ has its (3,1) and (3,2) elements equal to 0. Using the same method as above, take

$$\beta = \arctan\ (\ \frac{(\tilde{\mathbf{P}}\mathbf{R}_{Z,\theta})_{3,2}}{(\tilde{\mathbf{P}}\mathbf{R}_{Z,\theta})_{33}}\ ).$$

Note that this rotation does not affect the first column, and so the last row of $\tilde{\mathbf{P}}\mathbf{R}_{z,\theta}\mathbf{R}_{X,\beta}$ must be $(0, 0, \pm 1)$. (Recall that we began by normalizing such that the last row of $\tilde{\mathbf{P}}$ was of unit length, and the two rotations about will not change this property since rotations preserve length. Note that the 3rd element could be -1 though.)

Third, we find a rotation matrix $\mathbf{R}_{Z,\gamma}$ which sets element $(2, 1)$ to 0. Again this is done by a suitable choice of rotation angle, namely

$$\gamma = \arctan\ (\ \frac{(\tilde{\mathbf{P}}\mathbf{R}_{Z,\theta}\mathbf{R}_{X,\beta})_{2,1}}{(\tilde{\mathbf{P}}\mathbf{R}_{Z,\theta}\mathbf{R}_{X,\beta})_{2,2}}\ )\ .$$

Note that this X rotation doesn't affect the third row since its first two elements are 0. We are left with an upper triangular matrix $\tilde{\mathbf{P}}\mathbf{R}_{Z,\theta}\ \mathbf{R}_{X,\beta}\ \mathbf{R}_{Z,\gamma}$.

[ASIDE: I have just described a standard matrix decomposition in linear algebra called the *RQ decomposition*. In the *RQ* decomposition, R refers to a right upper triangular matrix (everything below the diagonal is zero) and Q refers to an orthonormal matrix. Curiously, Matlab doesn't implement the RQ decomposition. It only implements the QR decomposition.]

Finally, strictly speaking, we need the diagonal elements (1,1), (2,2) and (3,3) of $\mathbf{K}$ to be non-negative, so if these elements of $\tilde{\mathbf{P}}\mathbf{R}_{Z,\theta}\ \mathbf{R}_{X,\beta}\ \mathbf{R}_{Z,\gamma}$ are negative, then we need to reflect about the X, Y, Z axis using

$$\mathbf{K} = \tilde{\mathbf{P}}\mathbf{R}_{Z,\theta}\ \mathbf{R}_{X,\beta}\ \mathbf{R}_{Z,\gamma} \begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix}.$$

This gives us $\mathbf{K} = \tilde{\mathbf{P}}\mathbf{Q}$ where $\mathbf{Q}$ is orthonormal, and so $\tilde{\mathbf{P}} = \mathbf{K}\mathbf{Q}^T = \mathbf{K}\mathbf{R}$.

[ASIDE: Strictly speaking, the $\mathbf{Q}$ matrix might not be a rotation (it might have determinant -1, rather than 1) since we are allowing for reflections. This is not a problem, though it does contradict what I wrote in lecture 3: When I first set up the camera matrix, I said that the left 3×3 submatrix $\mathbf{R}$ was a "rotation." But in fact I never used the constraint that the matrix has determinant 1 (rather than -1). So it would have been better in lecture 3 if I had just stated that $\mathbf{R}$ was just orthonormal and could be a rotation and/or reflection.]

Step 3: obtain camera position C

Since **K R (-C)** is the fourth column of **P**, we can obtain **-C** by multiplying

$$\mathbf{R}^T\mathbf{K}^{-1}\begin{bmatrix} P_{14} \\ P_{24} \\ P_{34} \end{bmatrix} = -\mathbf{C}$$

i.e. $\mathbf{R}^T$ is a rotation matrix, so $\mathbf{R}^T = \mathbf{R}^{-1}$.