

Structure-from-motion: factorization method

Today we will look at a problem that has received an enormous amount of attention in the last two decades - called “structure from motion”. The general version of the problem is to take a video camera, move it around a scene and record a video, and from this video you compute the 3D scene geometry and camera motion. It sounds simple, ...

Today we look at one of the first methods for solving this problem, which introduced the method of “factorization”. The method is now called Tomasi-Kanade factorization¹. Here is what we assume:

- F image frames
- N corresponding image points (x_{ki}, y_{ki}) , $i = 1, \dots, N$ that are visible in each frame k . Typically, these points are detected and tracked from frame to frame. We have not discussed the problem of *tracking* a point – but we have discussed how to find interest points (Harris corners) and how to register two images. So you can imagine that we could find the motion of an interest point from one frame to the next.

The N points have *fixed* 3D coordinates $\{(X_i, Y_i, Z_i), i = 1, \dots, n\}$ which are expressed in unknown object coordinate system. Without loss of generality, we assume

$$\sum_{i=1}^N \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (1)$$

that is, we assume the origin is the centroid of the object.

- In each of the F frames, the average depth of the points is about Z_o and the range of depths is small, relative to Z_o . An example is that you are holding a small object at arm’s length, you rotate it in your hand. Another example is shown below. This was photographed using $f = 80$ mm, which gives a small field of view.



- We do not know the \mathbf{K} matrix and so we just assume it has the simplest form: $\alpha_x = \alpha_y$ and $s = 0$.

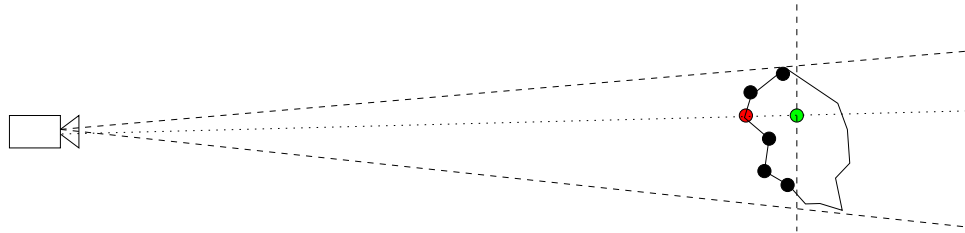
The model uses an “orthographic projection” which we have not discussed explicitly yet². It is closely related to the weak perspective approximation. To understand this model, consider the

¹Tomasi and Kanade, “Shape and motion from image streams: a factorization method”, IJCV 1992

²though we basically used such a model already, namely when we did shape from shading

sketch below. Compare the coordinates of the red point and green points, which lie on a line through the center of projection. The green point lies at depth Z_0 . The X, Y coordinates of these two points are almost exactly the same, in particular, their difference is far less than the difference in their Z values.

Orthographic projection is based on the above observation. Orthographic projection projects a point (X, Y, Z) to the $Z = 0$ plane, namely to $(x, y) = (X, Y)$. The idea is that if you don't know (or care) about the details of the camera internals or the distance Z_0 to the object, then orthographic projection is all you need.



Let \mathbf{A} be the image measurement matrix, which is constructed from the 2×1 matrices $\begin{bmatrix} x_{ki} \\ y_{ki} \end{bmatrix}$, i.e. point i seen in the k^{th} frame, $k = 1, \dots, F$. \mathbf{A} is $2F \times N$ data matrix. Think of it as two $F \times N$ matrices, one for the x_{ki} values and one for the y_{ki} values.

Each frame k defines its own mapping $\mathbf{R}_k[\mathbf{I} \mid -\mathbf{C}_k]$ from the scene XYZ coordinate system into the k^{th} camera coordinate system. So, for object point i and frame k , we have

$$\begin{bmatrix} x_{ki} \\ y_{ki} \\ Z_0 \end{bmatrix} \approx \mathbf{R}_k[\mathbf{I} \mid -\mathbf{C}_k] \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

The three rows of \mathbf{R}_k are the camera's unit X, Y, Z axes, expressed in the scene's coordinate frame – in particular, the third row of \mathbf{R}_k is the camera's Z axis, and the third element of $\mathbf{R}\mathbf{C}_k$ is the depth of the origin of the scene's coordinate system $(X, Y, Z) = (0, 0, 0)$ and this is our Z_0 . The model is approximating the Z values of all points as having this depth.

The approximation for the first two rows is very good if the assumptions on page 1 are met, whereas the approximation in the third row is not good, since it ignores the depth variations of the points in camera k . Because the third row's approximation is not good (and because we don't need it anyhow), we ignore it. That is, we project orthographically as described above.

Let $\tilde{\mathbf{R}}_k$ be the 2×3 matrix which is the first two rows of \mathbf{R}_k . Then,

$$\begin{bmatrix} x_{ki} \\ y_{ki} \end{bmatrix} = \tilde{\mathbf{R}}_k[\mathbf{I} \mid -\mathbf{C}_k] \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

The problem to be solved is now: Given the $2F \times N$ matrix \mathbf{A} , decompose it into the camera orientation $\tilde{\mathbf{R}}_k$ and position \mathbf{C}_k , and the scene "structure", i.e. the 3D positions (X_i, Y_i, Z_i) of the

N points, as expressed in the scene coordinate frame. Note that the camera position is not at all interesting in the problem setup, since we use orthographic projection. So we will concentrate on estimating $\tilde{\mathbf{R}}_k$ and (X_i, Y_i, Z_i) . This should be more clear in what follows.

For any camera k , consider the average positions of the points

$$\begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix} = \frac{1}{N} \sum_{i=1, \dots, N} \begin{bmatrix} x_{ki} \\ y_{ki} \end{bmatrix} = \frac{1}{N} \tilde{\mathbf{R}}_k [\mathbf{I} \mid -\mathbf{C}_k] \sum_{i=1, \dots, N} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Applying Eq. (1) to the right side, we get

$$\frac{1}{N} \sum_{i=1, \dots, N} \begin{bmatrix} x_{ki} \\ y_{ki} \end{bmatrix}_{2F \times K} = -\tilde{\mathbf{R}}_k \mathbf{C}_k_{2 \times 1}$$

Thus, for each camera k , we can compute $\tilde{\mathbf{R}}_k \mathbf{C}_k$, namely by taking the average of the (x_{ki}, y_{ki}) values over all i :

$$\begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix}_{2F \times K} \equiv \frac{1}{N} \sum_{i=1, \dots, N} \begin{bmatrix} x_{ki} \\ y_{ki} \end{bmatrix}_{2F \times K} = -\tilde{\mathbf{R}}_k \mathbf{C}_k_{2 \times 1}$$

This gives:

$$\begin{bmatrix} x_{ki} - \bar{x}_k \\ y_{ki} - \bar{y}_k \end{bmatrix}_{2F \times N} = [\tilde{\mathbf{R}}_k]_{2F \times 3} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}_{3 \times N} \quad (2)$$

We have two equations for each frame k , so we have stacked them to get a matrix product $2F \times N = (2F \times 3) \times (3 \times N)$. That is, we have a $2F \times N$ data matrix (left side) which is factored into the product of a matrix whose rows are the true X, Y camera axes of the k cameras – the “motion” – , and a matrix whose columns are the true 3D point points written in scene coordinates – the “structure”.

Keep in mind that the above is a model only. If there were no image noise and we could measure the keypoints (x_{ki}, y_{ki}) exactly, (and if the orthographic approximation in the projection model were exact), then the data matrix on the left would have rank 3, that is, the column space would span a (maximum) 3 dimensional space only. To see this, note that the right side of Eq. (2) maps from \mathcal{R}^N to \mathcal{R}^3 and then from \mathcal{R}^3 to \mathcal{R}^{2m} . The column space of the $2F \times 3$ matrix of \mathbf{R} vectors has dimension at most 3 and so this limits the dimension of the vectors reached by the mapping (i.e. the range).

We would like to estimate these $\tilde{\mathbf{R}}$ matrices as well as the points (X_i, Y_i, Z_i) . How can we do it?

We decompose the data matrix \mathbf{A} using the SVD:

$$\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{V}^T$$

where $\mathbf{U}_{2F \times N}$, $\mathbf{\Lambda}_{N \times N}$, $\mathbf{V}_{N \times N}^T$. If the model were exact, then \mathbf{A} would have rank 3, i.e. only three of the singular values would be non-zero. The best rank 3 approximation may be obtained by keeping only the largest 3 singular values and their corresponding eigenvectors:

$$\mathbf{A} \approx \tilde{\mathbf{A}} = \tilde{\mathbf{U}}_{2F \times 3} \tilde{\mathbf{\Lambda}}_{3 \times 3} \tilde{\mathbf{V}}_{3 \times N}^T$$

where $\tilde{\mathbf{U}}_{2F \times 3}$, $\tilde{\mathbf{\Lambda}}_{3 \times 3}$, $\tilde{\mathbf{V}}_{3 \times N}^T$. As long as the ignored singular values are small, the column space of \mathbf{A} will be approximately the same as the column space of $\tilde{\mathbf{U}}$.

[ASIDE: In class and in the slides, I gave a more detailed justification of this fact, namely that the best rank r approximation of a matrix is obtained by taking the SVD and setting singular values $r + 1$ and above to zero. By “best” here, we mean best according to the Frobenius norm which is the sum of squared entries in the matrix.]

Notice that the $\tilde{\mathbf{U}}$ matrix and the $\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T$ have the same dimensions as the rotation (motion) and (X, Y, Z) (structure) matrices, respectively. However, we cannot just assign these as our solution to the structure from motion problem, since there is no reason why rows $2k - 1$ and $2k$ of $\tilde{\mathbf{U}}$ should be orthonormal, and we *need* the two rows of $\tilde{\mathbf{R}}_k$ to be orthonormal. To solve this problem, we find matrices that are close to the $\tilde{\mathbf{U}}$ matrices, and that are orthonormal.

To do so, we insert write

$$\tilde{\mathbf{U}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T = \tilde{\mathbf{U}}\mathbf{Q}\mathbf{Q}^{-1}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T$$

where \mathbf{Q} is any 3×3 invertible matrix. Such a \mathbf{Q} matrix takes three linear combinations of the columns of $\tilde{\mathbf{U}}$, namely each column of \mathbf{Q} defines a linear combination of the columns of $\tilde{\mathbf{U}}$. We would like to take linear combinations of the three columns of $\tilde{\mathbf{U}}$ such that the F pairs of rows of $\tilde{\mathbf{U}}\mathbf{Q}$ are orthonormal, or as close to orthonormal as we can make them.

Let \mathbf{u}_{2k-1} and \mathbf{u}_{2k} be the k^{th} pair of 1×3 rows of $\tilde{\mathbf{U}}$. We want to choose \mathbf{Q} such that the following orthonormality condition applies:

$$(\mathbf{u}_{2k-1}\mathbf{Q}) \cdot (\mathbf{u}_{2k-1}\mathbf{Q}) = 1$$

$$(\mathbf{u}_{2k}\mathbf{Q}) \cdot (\mathbf{u}_{2k}\mathbf{Q}) = 1$$

$$(\mathbf{u}_{2k-1}\mathbf{Q}) \cdot \mathbf{u}_{2k}\mathbf{Q} = 0$$

in which case $(\mathbf{u}_{2k-1}\mathbf{Q})$ and $\mathbf{u}_{2k}\mathbf{Q}$ can serve as our rows of $\tilde{\mathbf{R}}_k$. This defines $3F$ equations that are second order in the 9 elements of \mathbf{Q} . We want to find a \mathbf{Q} that simultaneously solves these three equations – or at least solves them as close as possible. (We have more equations than unknowns and we have made approximations along the way, so we don’t expect an exact solution.) This requires we solve a *non-linear* least squares problem. Recall lecture 18.

One final point (not mentioned in class). Even if we use the factorization on the right side of

$$\tilde{\mathbf{U}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T = (\tilde{\mathbf{U}}\mathbf{Q})(\mathbf{Q}^{-1}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T)$$

there is still an ambiguity since we can always insert any 3×3 orthonormal matrix \mathbf{W}

$$(\tilde{\mathbf{U}}\mathbf{Q})(\mathbf{Q}^{-1}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T)(\tilde{\mathbf{U}}\mathbf{Q}\mathbf{W})(\mathbf{W}^T\mathbf{Q}^{-1}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^T)$$

and this will be an equally good solution/factorization i.e. since the quadratic constraints above are unaffected by the \mathbf{W} . What are these extra rotation/reflection degrees of freedom? They just allow for the fact that we cannot solve for the orientation of the scene and camera coordinate XYZ axes. They are always known only up to a 3D rotation.