Optimal orthogonal transformation

A rigid transformation $\mathbb{R}^k \to \mathbb{R}^k$ is the composition of a rotation and a translation. The position vector $\mathbf{x}$ of a point on a solid gets mapped into a new position vector $\mathbf{y} = Q\mathbf{x} + \mathbf{b}$, where the matrix Q determines the rotation, and the vector $\mathbf{b}$ determines the translation.

The task is to find the matrix Q and the vector $\mathbf{b}$ knowing the position vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of some characteristic points of the solid before the rigid transformation and position vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$ of these points after the transformation. This is a well known problem appearing in computer graphics, cheminformatics and bioinformatics.

Naïve approach

Given data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and $\mathbf{y}_1, \ldots, \mathbf{y}_n$ we have the following system of equations:

$$\begin{aligned} Q\mathbf{x}_i + \mathbf{b} &= \mathbf{y}_i,\ i = 1, \ldots, n \\ Q^\mathsf{T} Q &= I \end{aligned}$$

with unknowns

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1k} \\ q_{21} & q_{22} & \cdots & q_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ q_{k1} & q_{k2} & \cdots & q_{kk} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}.$$

(This is a system of $kn + k^2$ equations in $k^2 + k$ unknowns.) Problem: k^2 equations determined by $Q^\mathsf{T} Q = I$ are *not* linear. Naïve solution: Ignore $Q^\mathsf{T} Q = I$ and solve the remaining kn equations in $k^2 + k$ unknowns:

$$Q\mathbf{x}_i + \mathbf{b} = \mathbf{y}_i,\ i = 1, \ldots, n.$$

Assume that $n \geq 2k$, which gives us an (over)determined system of linear equations.

Write down the matrix of this system and find the linear least squares solution of this system; matrix Q' and vector $\mathbf{b}$. Since Q' is not necessarily orthogonal we make a (naïve) correction: Find the QR decomposition $Q' = QR$ of Q' and replace Q' with Q. The solution to the problem now consists of the matrix Q and the vector $\mathbf{b}$.

Kabsch algorithm

We can obtain the translation vector $\mathbf{b}$ as the translation of the center of mass of the points $\mathbf{x}_i$ and $\mathbf{y}_i$. If we set

$$\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \quad \text{and} \quad \overline{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i,$$

then $\mathbf{b} = \overline{\mathbf{y}} - Q\overline{\mathbf{x}}$.

Now set $\mathbf{x}'_i = \mathbf{x}_i - \overline{\mathbf{x}}$ and $\mathbf{y}'_i = \mathbf{y}_i - \overline{\mathbf{y}}$. The rotation matrix Q is an orthogonal matrix, which must satisfy the conditions $Q\mathbf{x}'_i = \mathbf{y}'_i$ for all $i = 1,\dots,n$. (Check that!) If we denote by X' and Y' the matrices

$$X' = [\mathbf{x}'_1,\dots,\mathbf{x}'_n] \quad \text{and} \quad Y' = [\mathbf{y}'_1,\dots,\mathbf{y}'_n],$$

we can merge our conditions into $QX' = Y'$. We then obtain Q using SVD with the following steps:

1. Evaluate the $k \times k$ covariance matrix $C = Y'X'^{\mathsf{T}}$.
2. Find the SVD of the matrix C, $C = USV^{\mathsf{T}}$.
3. Replace the matrix S with the diagonal matrix

$$D = \begin{bmatrix} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \\ 0 & \cdots & 0 & d \end{bmatrix},$$

   where $d = \pm 1$ is the sign of $\det(C)$ or $\det(UV^{\mathsf{T}})$.
4. The matrix Q is then $Q = UDV^{\mathsf{T}}$.

Extra credit: Nonlinear improvement of the naïve solution

The naïve approach ignored the equation $Q^{\mathsf{T}}Q = I$ in the system

$$\begin{aligned} Q\mathbf{x}_i + \mathbf{b} &= \mathbf{y}_i,\ i = 1,\dots,n \\ Q^{\mathsf{T}}Q &= I \end{aligned}$$

and returned (as a solution of the system determined only by the first equation) the matrix Q' and the column $\mathbf{b}$. Use this as the initial guess for the Gauss–Newton iteration on the full (nonlinear) system of equations. Write down the corresponding vector-valued function and its Jacobi matrix.

Task

1. Derive and write down the matrix and the right-hand side of the linear system obtained with the 'naïve approach'.

2. Write a Julia function `naive(X, Y)` which, given input data $X = [X_1,\ldots,X_n]$ and $Y = [Y_1,\ldots,Y_n]$, returns a tuple `(Q, b)` where the matrix Q and the vector **b** are as described in the 'naïve approach'. *Stick to specifications: X and Y are vectors of n-tuples representing the points, for example* `Xi = (Xi1, Xi2, ..., Xik)`, Q *is a square matrix,* **b** *is a column vector.*

3. Explain why the determinants of the matrices C and UV^T from Kabsch algorithm are of the same sign.

4. Write a Julia function `kabsch(X, Y)` which, given input data $X = [X_1,\ldots,X_n]$ and $Y = [Y_1,\ldots,Y_n]$, returns a tuple `(Q, b)` where the matrix Q and the vector **b** are as described in the Kabsch algorithm. *Stick to specifications: X and Y are vectors of n-tuples representing the points, for example* `Xi = (Xi1, Xi2, ..., Xik)`, Q *is a square matrix,* **b** *is a column vector.*

5. Write down the vector-valued function corresponding to the nonlinear system and its Jacobi matrix.

6. Write a Julia function `notSoNaive(X, Y)` which, given input data $X = [X_1,\ldots,X_n]$ and $Y = [Y_1,\ldots,Y_n]$, returns a tuple `(Q, b)` where the matrix Q and the vector **b** are obtained via the nonlinear improvement of the naïve solution. *Specifications: X and Y are vectors of n-tuples representing the points, e.g.* `Xi = (Xi1, Xi2, ..., Xik)`, Q *is a square matrix,* **b** *is a column vector.*

Submission

Use the online classroom to submit the following:

1. A Julia file **oot.jl** containing the functions `naive` and `kabsch`. Both functions should be well commented.

2. A report file **solution.pdf** which contains the necessary derivations and answers to questions above. Your report should also include a statement explaining:

   - Who or what (AI tool) did you consult while solving this home assignment?

- Did you use AI tools for coding? To what extent is the code your own work and to what extent is it AI-generated?

3. If you do the extra credit task include the function `notSoNaive` in the file **oot.jl** and include the necessary derivations in your report.

While you can discuss solutions of the problems with your colleagues, the programs and report must be your own creation. You can use all Julia functions from lab sessions.
Full solution to this assignment (without the extra credit task) is worth 15 points. The assignment will first be reviewed and graded by teaching assistants – (maximum) 10 points. Then, at an appropriate time, you will take a short quiz in the online classroom regarding your solution to the assignment – (maximum) 5 points.