Code for solving AX=YB

1. Problem that can be solved with this code:
   1. Estimate X and Y given (A\_i, B\_i) where X, Y, A\_i and B\_i are elements of SE(3), i.e., 4X4 transformation matrices.
   2. Estimate X and Y given (A\_i, B\_i) where X, Y, A\_i and B\_i are elements of SO(3), i.e., 3X3 rotation matrices.
2. Methods implemented in this code:
   1. Local geometric optimization: Gradient descent method that respects geometric structure of SO(3) and SE(3). Initial guess is computed through one of classical hand-eye calibration approaches. Objective function is weighted sum of rotation error (in radian) and translation error.
   2. Stochastic global geometric optimization: Solving local geometric optimizations as many times with different initial guesses as it reaches certain statistic certainty that the global minimizer is captured.
   3. Quaternion-based approach: Closed-form solution derived through quaternion representation of rotation matrix. Due to ‘one to two map’ from rotation matrix to quaternion, there are 2^n choices of these closed-form solutions where n is number of data pairs (A\_i, B\_i).
3. Code Examples:
   1. main\_dataGeneration\_SO3.m / main\_dataGeneration\_SE3.m: Generate random synthetic data of X, Y, A\_i and B\_i in SO(3)/SE(3).
   2. main\_solve\_AXYB\_SO3.m / main\_solve\_AXYB\_SE3.m: Estimate X and Y for given (A\_i, B\_i) and compare them with ground truths.
4. Code Instructions:
   1. Call **[X, Y] = solveAXYB\_SE3(A,B,alpha,param)**. As opposed to the function name, it deals with SO(3) case as well as SE(3) case.
   2. Inputs:
      1. A and B should be 3X3Xn matrices in SO(3) case and 4X4Xn matrices in SE(3) case.
      2. alpha: weight of translation error. Ex) if Alpha = pi/180, it weighs 1 unit length of translation error to be equally effective to 1deg rotation error. This value is not used in SO(3) case.
      3. param: defines optimization methods, tolerances, and so on. See basic instruction below:
         1. Define it as **param = defaultParam()**.
         2. Choose method by setting param.globalOptMethod.

|  |  |
| --- | --- |
| param.globalOptMethod | Method |
| 0 (default) | Local optimization – using an existing AX=XB method for initial guess. |
| 1 | Global optimization (method a) – multiple local optimizations with many initial points |
| 2 (recommended) | Global optimization (method b) – Faster than 1. Perform local optimizations only for good initial guesses. |
| -1 | Quaternion-based method (method c) – booking keeping is used to pick a solution out of 2^n solutions. Not very reliable. |
| -2 (recommended) | Quaternion-based method (method c) – genetic algorithm picks the best solution out of 2^n solutions. |

* 1. Outputs:
     1. X,Y: estimated frames