1. compute_fundamental()
    a. Given data1 and data2, we find the normalizing matrices for each set of data.
    b. Using the normalizing matrices, normalize the points in data1 and data2.
    c. Using corresponding pairs in normalized data1 and data2, we construct the constraint matrix A and perform SVD on A, getting the solution of F as the last row of Vh.
    d. Construct the initial F matrix from the solution.
    e. Enforce the singularity constraint on the initial F, to get the final estimate F matrix.
    f. Normalized the F matrix by making $f_{33} = 1$.
2. compute_essential()
    a. Given data1, data2 and K matrix, normalize data1 and data2 using K^{-1}.
    b. Using corresponding pairs in normalized data1 and data2, we construct the constraint matrix A and perform SVD on A, getting the solution of E as the last row of Vh.
    c. Construct the initial E matrix from the solution.
    d. Enforce the singularity constraint on E, to get the final estimated E matrix.
3. decompose_e()
    a. Given E, perform SVD on E.
    b. Construct matrix t from the 3rd column of U from SVD(E) by
        i.
```
t = e_u[:, 2].reshape((3, 1))
```
    c. Construct matrices r1 and r2 from SVD(E) and matrix W by
        i.
```
r1 = np.matmul(np.matmul(e_u, W), e_vh)
r2 = np.matmul(np.matmul(e_u, np.transpose(W)), e_vh)
```
    d. Check for determinant of r1 and r2
        i. If determinant is <0, invert the sign of each r1 and r2.
    e. Construct the 4 possible P primes.
    f. For each possible P primes
        i. Find the 3D intersection point between corresponding points.
        ii. Check the depth of the intersection point form both pose P and P prime
        iii. If both depths are positive, means intersection point is in front of both poses
        iv. Repeat above till all corresponding points are checked and return the P primes with the most intersection points that are in front of both.
4. pnp_algo()
    a. Given points2d, homogenize the points.
    b. For each point given, get the depths by
        i. Picking 2 more points from the remaining 9 points to form a set of 3 points.
        ii. Calculate the distance-squared between each 3d point in the set.
        iii. Calculate the cos-theta angle between each 2d point in the set.
        iv. Add the extracted coefficients from equation to constraints.
        v. Repeat above till all other points combination are used, forming a 36x5 matrix A
        vi. Perform SVD(A) and the solution isthe last column of Vh (t0, t1, t2, t3, t4).
        vii. Calculate x = average(t1/t0, t2/t1, t3/t2, t4/t3)
        viii. Final depth of the chosen point is sqrt(x)
        ix. Repeat b for every point given, getting a list of depth for each point (X).
    c. Reconstruct the 3D points using reconstruct_3d()
    d. Calculate r and t using icp()