```
t0 = [0;0;0]; % setting initial transformation
R0 = [1,0,0;0,1,0;0,0,1]; % setting initial rotation
X = load('pclX.txt');
Y = load("pclY.txt");
n_x = size(X,1); % calculating length of X points
n_y = size(Y,1); % calculating length of Y points
d_max = 0.25; % setting maximum distance threshold

num_iter = 30;

% assigning initial transformation matrices
t=t0;
R=R0;


for i = 1:1:num_iter %setting num_iter here
    disp(i)
C = []; % creating blank correspondence matrix

    for i = 1:1:n_x % looping through every X point in list
        Xi = X(i,:)'; % extracting points from X Matrix and making it a
column vector
        Xi_trans = R*Xi + t; % transforming point X to its image under T

        % Reset min_dist for each new point in X + create default condition
        min_dist = inf;
        best_j = -1; % index of Y that best matches X_i

        for j = 1:1:n_y %  looping through Y
            Yj = Y(j,:)'; % extracting points from Y matrix and making it a
column vector
            distance = norm(Yj - Xi_trans); % Euclidean distance between Y_j
and transformed X_i

            if distance < min_dist
                min_dist = distance; % updating default condition to find the
point Y that best corresponds to X_i
                best_j = j; % updating default condition index to find the Y
that best corresponds to X_i
            end
        end

        % Checking if min point < d_max
        if min_dist < d_max % parent ICP condition
            C = [C; i, best_j]; % Add correspondence as new row where the
elements are the indices of the parent X and Y matrices that correspond
        end
    end

    % Horns Method

    K = size(C, 1) ;     % finding the dimension of the correspondence matix
```

```
for calcilating weighted means

    X_corr = X(C(:,1), :);     % extracting correlated X and Y points per C
matrix
    Y_corr = Y(C(:,2), :);

    % Centroids:
    x_bar = mean(X_corr, 1)';  % finding centroid of X dataset as a column
vector
    y_bar = mean(Y_corr, 1)';      % finding centroid of Y dataset as a
column vector

    %Calculate deviations of each point from the centroid of its pointcloud:

    X_prime = (X_corr - x_bar')';      %re-transposing X_bar/Y_Bar to find
the deviation from the points before transposing into column vector
    Y_prime = (Y_corr - y_bar')';

    %cross covariance matrix
    W = (Y_prime * X_prime') / K ;

    %symmetric value decomposition

    [U,S,V] = svd(W);

    %Constructing optimal rotation.
    diag_matrix = eye(3);
    diag_matrix(3, 3) = det(U * V');

    %finding rotation matrix
    R_hat = U * diag_matrix * V';
    %finding translation component
    t_hat = y_bar - R_hat*x_bar;

    %updating parent rotation and translation matrices ahead of next
    %iteration

    R=R_hat;
    t=t_hat;


end

%calculating RMSE in 2 steps.

%STEP 1 - SSE of Corresponded points
SSE = 0;
for k = 1:1:K %looping over length of C

    i = C(k, 1);  % indices of X
    j = C(k, 2);  % indices of Y

    x_trans = R * X(i,:)' + t; %transformed X Coorindates with final R and t
    error_vec = Y(j,:)' - x_trans; % calculating error between the
```

```
transformed X and Y
    SSE = SSE + norm(error_vec)^2;
end

RMSE = sqrt(SSE / K);
disp(RMSE)
```

```
     1

     2

     3

     4

     5

     6

     7

     8

     9

    10

    11

    12

    13

    14

    15

    16

    17

    18

    19

    20

    21

    22

    23

    24
```

```
    25

    26

    27

    28

    29

    30

    0.0090
```

Published with MATLAB® R2024b