Calibration Project Update

Code Walkthrough: CalibrateOffsetCube.m

Part 1: Robot Imports

Creates 2 robot variables from the URDF and configurations to match their poses

```
%% URDF Import for Tormach ZA6
% Using the designed calibration cube points and CMM Measured points
% Script calculates the Rotation and translation vectors between tool and
% CMM for 2 robots
% Created by Andrew Schneider, June 27, 2024
% Edited: Sep 2, 2024
clear, clc, close all, format compact
%% Robot 1 Import
robot1 = importrobot('targetTest.urdf');
config1 = homeConfiguration(robot1);
config1(1).JointPosition = deg2rad(0);
config1(2).JointPosition = deg2rad(21);
config1(3).JointPosition = deg2rad(45);
config1(4).JointPosition = deg2rad(0);
config1(5).JointPosition = deg2rad(360+(-66));
config1(6).JointPosition = deg2rad(0);

%% Robot 2 Import

robot2 = importrobot('targetTest.urdf');
config2 = homeConfiguration(robot2);
config2(1).JointPosition = deg2rad(0);
config2(2).JointPosition = deg2rad(21);
config2(3).JointPosition = deg2rad(45);
config2(4).JointPosition = deg2rad(0);
config2(5).JointPosition = deg2rad(360+(-66));
config2(6).JointPosition = deg2rad(0);
```

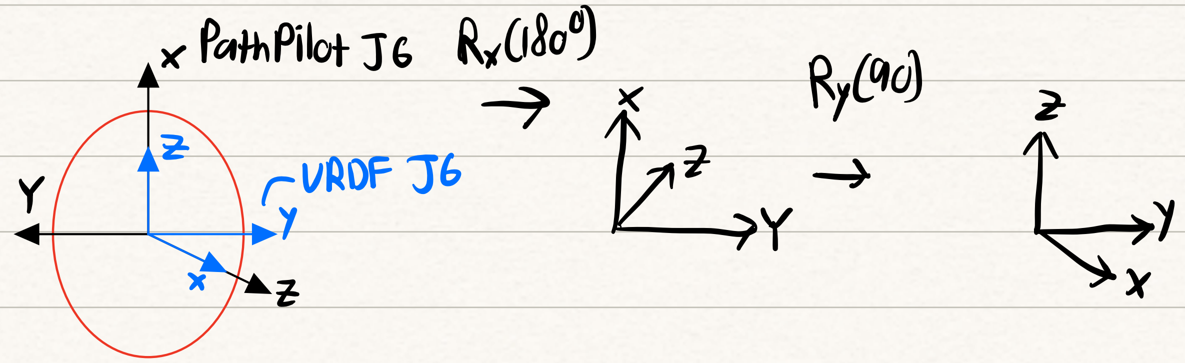
Part 2: Calibration Cube coords

Creates a 3x20 Matrix of the coordinates of the tool points. Converts from mm to m, as URDF is in meters.

```
%% Designed tool points
%       X       Y       Z
v1 = [-4.079; 19.427; 10.007];
v2 = [-4.122; 19.510; 20.063];
v3 = [5.950; 18.968; 10.149];
v4 = [5.856; 19.036; 20.137];

v5 = [12.144; 12.003; 10.203];
v6 = [12.067; 12.033; 20.247];
v7 = [11.013; -13.005; 10.468];
v8 = [10.916; -12.972; 20.413];

v9  = [4.103; -19.281; 10.399];
v10 = [4.057; -19.205; 20.430];
v11 = [-5.956; -18.801; 10.426];
v12 = [-6.012; -18.705; 20.391];

v13 = [-12.311; -11.923; 10.357];
v14 = [-12.324; -11.816; 20.325];
v15 = [-11.146; 13.019; 10.140];
v16 = [-11.196; 13.157; 20.113];

v17 = [-4.643; 12.833; 26.825];
v18 = [5.487; 12.386; 26.861];
v19 = [-5.740; -12.115; 27.011];
v20 = [4.243; -12.584; 27.049];

% 3x20 matrix
V =[v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17, v18, v19, v20;]/1000;
```

Part 3: Reorient Cube points

Cube points were taken in the PathPilot J6 Frame. The CMM was used to create a frame on the Flange, and then measured the points in Part 2. This Section reorients the points into the URDF J6.




```
%% Align coordinate points frame to urdf frame
% Note: The flange frame used in the urdf and Tormach PathPilot differ
% The CMM built frame was aligned with PathPilot, this code snippet
% aligns the points with the flange frame used in the URDF.
% The CMM/PathPilot Flange frame is: Z out of End Effector face, X up.

eul = [pi, pi/2, 0];
rotmXYZ = eul2rotm(eul,'XYZ');
V1 = rotmXYZ*V;
```

Part 4: Cube points in CMM Frame

40 total points, 20 per Cube/Robot, Converted to meters

```
%% Robot 1 Measurement Points, output of CMM
%       X       Y       Z
w1 = [173.535; -21.405; 430.405];
w2 = [163.495; -20.839; 430.512];
w3 = [173.371; -21.330; 440.624];
w4 = [163.398; -20.785; 440.501];

w5 = [172.980; -27.951; 447.174];
w6 = [162.949; -27.439; 447.069];
w7 = [171.506; -52.941; 447.352];
w8 = [161.574; -52.433; 447.229];

w9  = [171.270; -59.567; 440.781];
w10 = [161.255; -59.009; 440.705];
w11 = [171.266; -59.614; 430.710];
w12 = [161.318; -59.040; 430.624];

w13 = [171.669; -53.090; 424.004];
w14 = [161.717; -52.502; 423.960];
w15 = [173.092; -28.160; 423.863];
w16 = [163.137; -27.544; 423.780];

w17 = [156.418; -27.199; 430.324];
w18 = [156.360; -27.118; 440.363];
w19 = [155.024; -52.132; 430.534];
w20 = [154.964; -52.075; 440.528];

% 3xX matrix
W1 = [w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14, w15, w16, w17, w18, w19, w20]./1000;
```

```
%% Robot 2 Measurement Points, output of CMM
%       X       Y       Z
w1 = [-220.342; -46.656; 432.081];
w2 = [-219.297; -46.395; 431.954];
w3 = [-220.150; -46.779; 442.095];
w4 = [-210.169; -46.502; 441.944];

w5 = [-220.274; -40.179; 448.688];
w6 = [-210.266; -39.922; 448.584];
w7 = [-220.852; -15.189; 449.015];
w8 = [-210.917; -14.862; 448.867];

w9  = [-221.181; -8.525; 442.480];
w10 = [-211.157; -8.257; 442.375];
w11 = [-221.203; -8.417; 432.413];
w12 = [-211.244; -8.172; 432.298];

w13 = [-221.054; -14.814; 425.636];
w14 = [-211.133; -14.655; 425.575];
w15 = [-220.481; -39.860; 425.350];
w16 = [-210.509; -39.643; 425.251];

w17 = [-203.770; -39.459; 431.782];
w18 = [-203.677; -39.598; 441.835];
w19 = [-204.419; -14.505; 432.144];
w20 = [-204.323; -14.618; 442.132];

% w1 = [; ; ];...

% 3x20 matrix
W2 = [w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14, w15, w16, w17, w18, w19, w20]/1000;
```

Part 5: Use Sub-Function

Using the designed points, the measured points, the URDF Robot model, and the pose calculate the Transformation Matrix between the CMM and a given Robot base.

```
%% Use Function

H_CMMB1 = Cmm2Robot(V1,W1,robot1,config1);
H_CMMB2 = Cmm2Robot(V1,W2,robot2,config2);
```

Part 5.2: The subfunction Cmm2Robot.m

Solving $RA+t=B$

A: Cube points in J6 Frame
B: Cube points in CMM Frame

$Centroid_A$
$Centroid_B$

$$H = \underbrace{(A - Centroid_A) \cdot (B - Centroid_B)^T}$$

Sets both sets of points to their origins. Thus only difference is rotation about origin

$[U,S,V] = svd(H);$

$R = V \cdot Transpose(U)$

I do not understand How/why this works.

$$R \times A + t = B \Big\} \text{Overall equation to solve}$$

$$\downarrow$$

$$t = P - {}^{A}R_{B} \times Centroid_{A}$$

$$^{J6}H_{cmm} = \begin{bmatrix} [R] & [t] \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

$$^{cmm}H_{Base} = {}^{cmm}H_{J6} \cdot {}^{J6}H_{Base}$$

$$= ({}^{J6}H_{cmm})^{T} \cdot {}^{J6}H_{Base}$$

☆ This makes sense to me, but I believe is wrong

$$^{cmm}H_{Base} = {}^{J6}H_{cmm} \cdot {}^{Base}H_{link\_6}$$

↳ URDF Name

```
function [H_CMMBase] = Cmm2Robot(V,W,robot,config)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
%% Find Rigid Transformation matrix between tool and CMM
% Cite: cs.hunter.edu/~ioannis/registerpts_allen_notes.pdf
A = V; % tool data
B = W; % CMM data

centroid_A = [mean([A(1,:)]); mean([A(2,:)]); mean([A(3,:)])]; % centroid of A
centroid_B = [mean([B(1,:)]); mean([B(2,:)]); mean([B(3,:)])]; % cetroid of A

H = (A-centroid_A)*transpose(B-centroid_B);

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

R_AB = V*transpose(U);

% account for special reflection Citation: nghiaho.com Finding optimal
% Rotation and Translation Between Corresponding 3D points
if det(R_AB) < 0
    [U,S,V] = svd(H);
    V(1:3,3) = V(1:3,3)*-1;
    R_AB = V*transpose(U);
end
t_AB = centroid_B - R_AB*centroid_A;

H_J6CMM = [R_AB, t_AB; 0, 0, 0, 1];

H_CMMBase = H_J6CMM*getTransform(robot,config,"base","link_6");
end
```

Part 6: World Frame Between Bots

Creates a New origin between the robots by Averaging their translations, And $\frac{1}{2}$ meter up from Zero

Then Outputs the PathPilot Values for transformation and rotation for Frame

```
%% Find and convert to world between bots
T_mid = [(H_CMMB1(1,4)+H_CMMB2(1,4))/2, (H_CMMB1(2,4)+H_CMMB2(2,4))/2,((H_CMMB1(3,4)+H_CMMB2(3,4))/2)+.5];
H_CMMMid = [1 0 0 T_mid(1);
            0 1 0 T_mid(2);
            0 0 1 T_mid(3);
            0 0 0     1   ;];

H_MB1 = inv(H_CMMMid)*H_CMMB1;
H_B1M = inv(H_MB1);
T_B1M = H_B1M(1:3,4)*1000
eul1 = rotm2eul(H_B1M(1:3,1:3));
eul1 = flip(rad2deg(eul1))


H_MB2 = inv(H_CMMMid)*H_CMMB2;
H_B2M = inv(H_MB2);
T_B2M = H_B2M(1:3,4)*1000
eul2 = rotm2eul(H_B2M(1:3,1:3));
eul2 = flip(rad2deg(eul2))
```

Outputs in rot about Z,Y,X order

Flips to X,Y,Z order