Table of Contents


```
clear all; clc; close all
```

Task 1 Data

```
% load data
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task1_translation.mat');
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task1_rotation.mat');

% Convert Quaternions to Euler Angles
jaco_task1_rotation = quat2eul(jaco_task1_rotation, 'XYZ');

% Separate data by trial
msg = 1;
ind = [770; 972; 1065; 1200; 1099]; % number of datapoints per trial
 (/tf messages)
jaco_task1_trials_translation = {};
jaco_task1_trials_rotation = {};

for i = 1:length(ind)
    jaco_task1_trials_translation{end+1} =
 jaco_task1_translation(msg:sum(ind(1:i)),:);
    jaco_task1_trials_rotation{end+1} =
 jaco_task1_rotation(msg:sum(ind(1:i)),:);

    msg = msg + ind(i);
end

% Find Changes in Position and Orientation
jaco_task1_vel = {[] [] [] [] []};
jaco_task1_omg = {[] [] [] [] []};

for i = 1:length(ind)
    for j = 1:(size(jaco_task1_trials_translation{i},1)-1)

        vel = jaco_task1_trials_translation{i}(j+1,:) -
 jaco_task1_trials_translation{i}(j,:); % position change
```

```
        omg = jaco_task1_trials_rotation{i}(j+1,:) -
 jaco_task1_trials_rotation{i}(j,:); % orientation change

        if max(abs(vel)) > 1e-4     % ignore small commands
            jaco_task1_vel{i} = [jaco_task1_vel{i}; vel]; %
 translation
        end

        if max(abs(omg)) > 1e-4     % ignore small commands
            jaco_task1_omg{i} = [jaco_task1_omg{i}; omg]; % rotation
        end
    end
end

% Proportion of Axes Used for Each Trial
count1 = zeros(5,6); % count commands from axes [x y z X Y Z] for each
 trial
percent1 = zeros(5,6); %  proportion of points axis used : trial(row)
 vs axis(column)

trans = [425; 539; 601; 630; 556];
rot = [430; 567; 620; 637; 604];

for i = 1:length(ind)
    % Translation
    for j = 1:size(jaco_task1_vel{i},1)
        for k = 1:size(jaco_task1_vel{i},2)
            if jaco_task1_vel{i}(j,k) > 1e-4   % ignore small commands
                count1(i,k) = count1(i,k) + 1;
            end
        end
    end
    percent1(i,1:3) = count1(i,1:3) ./ trans(i);

    % Rotation
    for j = 1:size(jaco_task1_omg{i},1)
        for k = 1:size(jaco_task1_omg{i},2)
            if jaco_task1_omg{i}(j,k) > 1e-4   % ignore small commands
                count1(i,k+3) = count1(i,k+3) + 1;
            end
        end
    end
    percent1(i,4:6) = count1(i,4:6) ./ rot(i);
end
```

Task 2 Data

```
% load data
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task2_translation.mat');
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task2_rotation.mat');
```

```
% Convert Quaternions to Euler Angles
jaco_task2_rotation = quat2eul(jaco_task2_rotation, 'XYZ');

% Separate data by trial
msg = 1;
ind = [835; 847; 758; 774; 897]; % number of datapoints per trial (/tf
 messages)
jaco_task2_trials_translation = {};
jaco_task2_trials_rotation = {};

for i = 1:length(ind)
    jaco_task2_trials_translation{end+1} =
 jaco_task2_translation(msg:sum(ind(1:i)),:);
    jaco_task2_trials_rotation{end+1} =
 jaco_task2_rotation(msg:sum(ind(1:i)),:);

    msg = msg + ind(i);
end

% Find Changes in Position and Orientation
jaco_task2_vel = {[] [] [] [] []};
jaco_task2_omg = {[] [] [] [] []};

for i = 1:length(ind)
    for j = 1:(size(jaco_task2_trials_translation{i},1)-1)

        vel = jaco_task2_trials_translation{i}(j+1,:) -
 jaco_task2_trials_translation{i}(j,:); % position change
        omg = jaco_task2_trials_rotation{i}(j+1,:) -
 jaco_task2_trials_rotation{i}(j,:); % orientation change

        if max(abs(vel)) > 1e-4     % ignore small commands
            jaco_task2_vel{i} = [jaco_task2_vel{i}; vel]; %
 translation
        end

        if max(abs(omg)) > 1e-4     % ignore small commands
            jaco_task2_omg{i} = [jaco_task2_omg{i}; omg]; % rotation
        end
    end
end

% Proportion of Axes Used for Each Trial
count2 = zeros(5,6); % count commands from axes [x y z X Y Z] for each
 trial
percent2 = zeros(5,6); %  proportion of points axis used : trial(row)
 vs axis(column)

trans = [460; 446; 403; 405; 480];
rot = [473; 444; 404; 429; 490];

for i = 1:length(ind)
    % Translation
    for j = 1:size(jaco_task2_vel{i},1)
```

```
            for k = 1:size(jaco_task2_vel{i},2)
                if jaco_task2_vel{i}(j,k) > 1e-4   % ignore small commands
                    count2(i,k) = count2(i,k) + 1;
                end
            end
        end
        percent2(i,1:3) = count2(i,1:3) ./ trans(i);

        % Rotation
        for j = 1:size(jaco_task2_omg{i},1)
            for k = 1:size(jaco_task2_omg{i},2)
                if jaco_task2_omg{i}(j,k) > 1e-4   % ignore small commands
                    count2(i,k+3) = count2(i,k+3) + 1;
                end
            end
        end
        percent2(i,4:6) = count2(i,4:6) ./ rot(i);
    end
```

Task 3 Data

```
% load data
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task3_translation.mat');
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task3_rotation.mat');

% Convert Quaternions to Euler Angles
jaco_task3_rotation = quat2eul(jaco_task3_rotation, 'XYZ');

% Separate data by trial
msg = 1;
ind = [688; 696; 643; 559; 553]; % number of datapoints per trial (/tf
 messages)
jaco_task3_trials_translation = {};
jaco_task3_trials_rotation = {};

for i = 1:length(ind)
    jaco_task3_trials_translation{end+1} =
 jaco_task3_translation(msg:sum(ind(1:i)),:);
    jaco_task3_trials_rotation{end+1} =
 jaco_task3_rotation(msg:sum(ind(1:i)),:);

    msg = msg + ind(i);
end

% Find Changes in Position and Orientation
jaco_task3_vel = {[] [] [] [] []};
jaco_task3_omg = {[] [] [] [] []};

for i = 1:length(ind)
    for j = 1:(size(jaco_task3_trials_translation{i},1)-1)
```

```
        vel = jaco_task3_trials_translation{i}(j+1,:) -
 jaco_task3_trials_translation{i}(j,:); % position change
        omg = jaco_task3_trials_rotation{i}(j+1,:) -
 jaco_task3_trials_rotation{i}(j,:); % orientation change

        if max(abs(vel)) > 1e-4     % ignore small commands
            jaco_task3_vel{i} = [jaco_task3_vel{i}; vel]; %
 translation
        end

        if max(abs(omg)) > 1e-4     % ignore small commands
            jaco_task3_omg{i} = [jaco_task3_omg{i}; omg]; % rotation
        end
    end
end

% Proportion of Axes Used for Each Trial
count3 = zeros(5,6); % count commands from axes [x y z X Y Z] for each
 trial
percent3 = zeros(5,6); %  proportion of points axis used : trial(row)
 vs axis(column)

trans = [400; 410; 332; 291; 291];
rot = [413; 413; 359; 318; 314];

for i = 1:length(ind)
    % Translation
    for j = 1:size(jaco_task3_vel{i},1)
        for k = 1:size(jaco_task3_vel{i},2)
            if jaco_task3_vel{i}(j,k) > 1e-4   % ignore small commands
                count3(i,k) = count3(i,k) + 1;
            end
        end
    end
    percent3(i,1:3) = count3(i,1:3) ./ trans(i);

    % Rotation
    for j = 1:size(jaco_task3_omg{i},1)
        for k = 1:size(jaco_task3_omg{i},2)
            if jaco_task3_omg{i}(j,k) > 1e-4   % ignore small commands
                count3(i,k+3) = count3(i,k+3) + 1;
            end
        end
    end
    percent3(i,4:6) = count3(i,4:6) ./ rot(i);
end
```

Task 4 Data

```
% load data
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task4_translation.mat');
```

```
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task4_rotation.mat');

% Convert Quaternions to Euler Angles
jaco_task4_rotation = quat2eul(jaco_task4_rotation, 'XYZ');

% Separate data by trial
msg = 1;
ind = [871; 858; 915; 894; 1016]; % number of datapoints per trial (/
tf messages)
jaco_task4_trials_translation = {};
jaco_task4_trials_rotation = {};

for i = 1:length(ind)
    jaco_task4_trials_translation{end+1} =
 jaco_task4_translation(msg:sum(ind(1:i)),:);
    jaco_task4_trials_rotation{end+1} =
 jaco_task4_rotation(msg:sum(ind(1:i)),:);

    msg = msg + ind(i);
end

% Find Changes in Position and Orientation
jaco_task4_vel = {[] [] [] [] []};
jaco_task4_omg = {[] [] [] [] []};

for i = 1:length(ind)
    for j = 1:(size(jaco_task4_trials_translation{i},1)-1)

        vel = jaco_task4_trials_translation{i}(j+1,:) -
 jaco_task4_trials_translation{i}(j,:); % position change
        omg = jaco_task4_trials_rotation{i}(j+1,:) -
 jaco_task4_trials_rotation{i}(j,:); % orientation change

        if max(abs(vel)) > 1e-4     % ignore small commands
            jaco_task4_vel{i} = [jaco_task4_vel{i}; vel]; %
 translation
        end

        if max(abs(omg)) > 1e-4     % ignore small commands
            jaco_task4_omg{i} = [jaco_task4_omg{i}; omg]; % rotation
        end
    end
end

% Proportion of Axes Used for Each Trial
count4 = zeros(5,6); % count commands from axes [x y z X Y Z] for each
 trial
percent4 = zeros(5,6); %  proportion of points axis used : trial(row)
 vs axis(column)

trans = [420; 419; 457; 397; 412];
rot = [488; 490; 514; 475; 566];
```

```
for i = 1:length(ind)
    % Translation
    for j = 1:size(jaco_task4_vel{i},1)
        for k = 1:size(jaco_task4_vel{i},2)
            if jaco_task4_vel{i}(j,k) > 1e-4   % ignore small commands
                count4(i,k) = count4(i,k) + 1;
            end
        end
    end
    percent4(i,1:3) = count4(i,1:3) ./ trans(i);

    % Rotation
    for j = 1:size(jaco_task4_omg{i},1)
        for k = 1:size(jaco_task4_omg{i},2)
            if jaco_task4_omg{i}(j,k) > 1e-4   % ignore small commands
                count4(i,k+3) = count4(i,k+3) + 1;
            end
        end
    end
    percent4(i,4:6) = count4(i,4:6) ./ rot(i);
end
```

Task 5 Data

```
% load data
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task5_translation.mat');
load('/home/tem/Documents/MATLAB/BMI/KD/6-28-19_Tem/MAT_Data/
jaco_task5_rotation.mat');

% Convert Quaternions to Euler Angles
jaco_task5_rotation = quat2eul(jaco_task5_rotation, 'XYZ');

% Separate data by trial
msg = 1;
ind = [1048; 1090; 975; 1019; 907]; % number of datapoints per trial
 (/tf messages)
jaco_task5_trials_translation = {};
jaco_task5_trials_rotation = {};

for i = 1:length(ind)
    jaco_task5_trials_translation{end+1} =
 jaco_task5_translation(msg:sum(ind(1:i)),:);
    jaco_task5_trials_rotation{end+1} =
 jaco_task5_rotation(msg:sum(ind(1:i)),:);

    msg = msg + ind(i);
end

% Find Changes in Position and Orientation
jaco_task5_vel = {[] [] [] [] []};
jaco_task5_omg = {[] [] [] [] []};
```

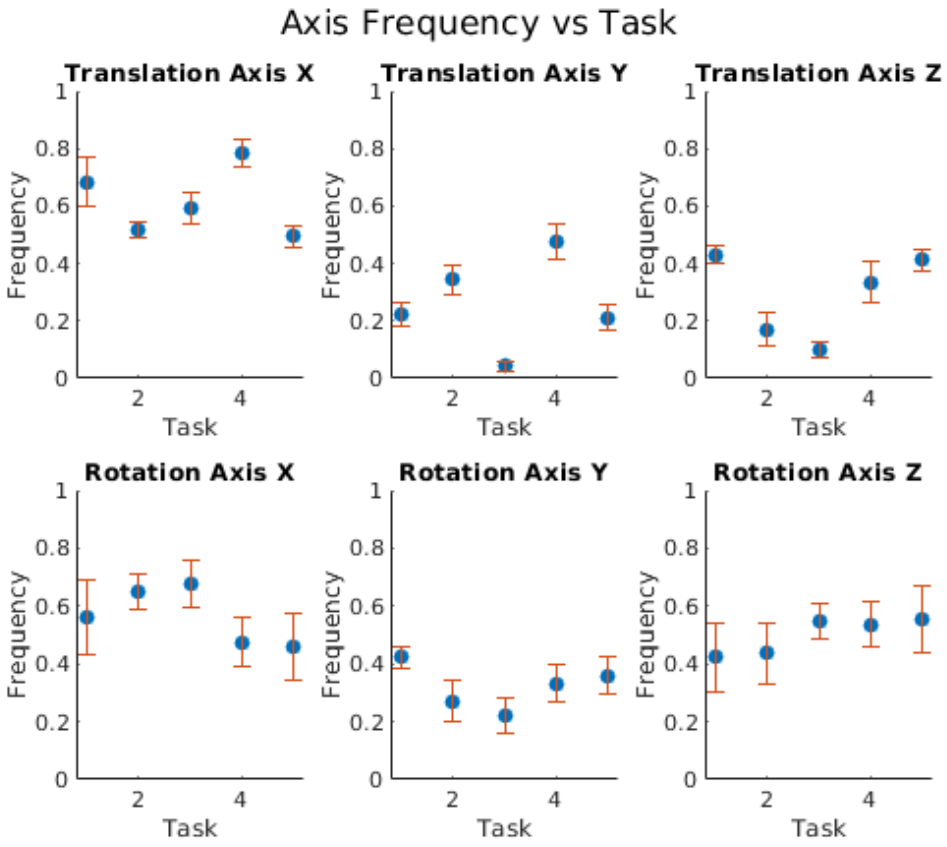
```
for i = 1:length(ind)
    for j = 1:(size(jaco_task5_trials_translation{i},1)-1)

        vel = jaco_task5_trials_translation{i}(j+1,:) -
 jaco_task5_trials_translation{i}(j,:); % position change
        omg = jaco_task5_trials_rotation{i}(j+1,:) -
 jaco_task5_trials_rotation{i}(j,:); % orientation change

        if max(abs(vel)) > 1e-4     % ignore small commands
            jaco_task5_vel{i} = [jaco_task5_vel{i}; vel]; %
 translation
        end

        if max(abs(omg)) > 1e-4     % ignore small commands
            jaco_task5_omg{i} = [jaco_task5_omg{i}; omg]; % rotation
        end
    end
end

% Proportion of Axes Used for Each Trial
count5 = zeros(5,6); % count commands from axes [x y z X Y Z] for each
 trial
percent5 = zeros(5,6); %  proportion of points axis used : trial(row)
 vs axis(column)

trans = [585; 612; 545; 558; 464];
rot = [613; 643; 556; 611; 479];

for i = 1:length(ind)
    % Translation
    for j = 1:size(jaco_task5_vel{i},1)
        for k = 1:size(jaco_task5_vel{i},2)
            if jaco_task5_vel{i}(j,k) > 1e-4   % ignore small commands
                count5(i,k) = count5(i,k) + 1;
            end
        end
    end
    percent5(i,1:3) = count5(i,1:3) ./ trans(i);

    % Rotation
    for j = 1:size(jaco_task5_omg{i},1)
        for k = 1:size(jaco_task5_omg{i},2)
            if jaco_task5_omg{i}(j,k) > 1e-4   % ignore small commands
                count5(i,k+3) = count5(i,k+3) + 1;
            end
        end
    end
    percent5(i,4:6) = count5(i,4:6) ./ rot(i);
end
```

Axis Frequency vs Task

```
% Calculate Means and Standard Deviations
```

```
percent = {percent1 percent2 percent3 percent4 percent5};

means = [];
stds = [];
for i = 1:5
    means = [means; mean(percent{i},1)];
    stds = [stds; std(percent{i},0,1)];
end

% Plot for each Axis

subplot(2,3,1)
scatter(1:5, means(:,1), 'filled');
axis([0.8 5.2 0 1])
xlabel('Task')
ylabel('Frequency')
title('Translation Axis X')
hold on
errorbar(1:5, means(:,1), stds(:,1),'LineStyle','None');
hold off

subplot(2,3,2)
scatter(1:5, means(:,2), 'filled');
axis([0.8 5.2 0 1])
xlabel('Task')
ylabel('Frequency')
title('Translation Axis Y')
hold on
errorbar(1:5, means(:,2), stds(:,2),'LineStyle','None');
hold off

subplot(2,3,3)
scatter(1:5, means(:,3), 'filled');
axis([0.8 5.2 0 1])
xlabel('Task')
ylabel('Frequency')
title('Translation Axis Z')
hold on
errorbar(1:5, means(:,3), stds(:,3),'LineStyle','None');
hold off

subplot(2,3,4)
scatter(1:5, means(:,4), 'filled');
axis([0.8 5.2 0 1])
xlabel('Task')
ylabel('Frequency')
title('Rotation Axis X')
hold on
errorbar(1:5, means(:,4), stds(:,4),'LineStyle','None');
hold off

subplot(2,3,5)
scatter(1:5, means(:,5), 'filled');
axis([0.8 5.2 0 1])
```

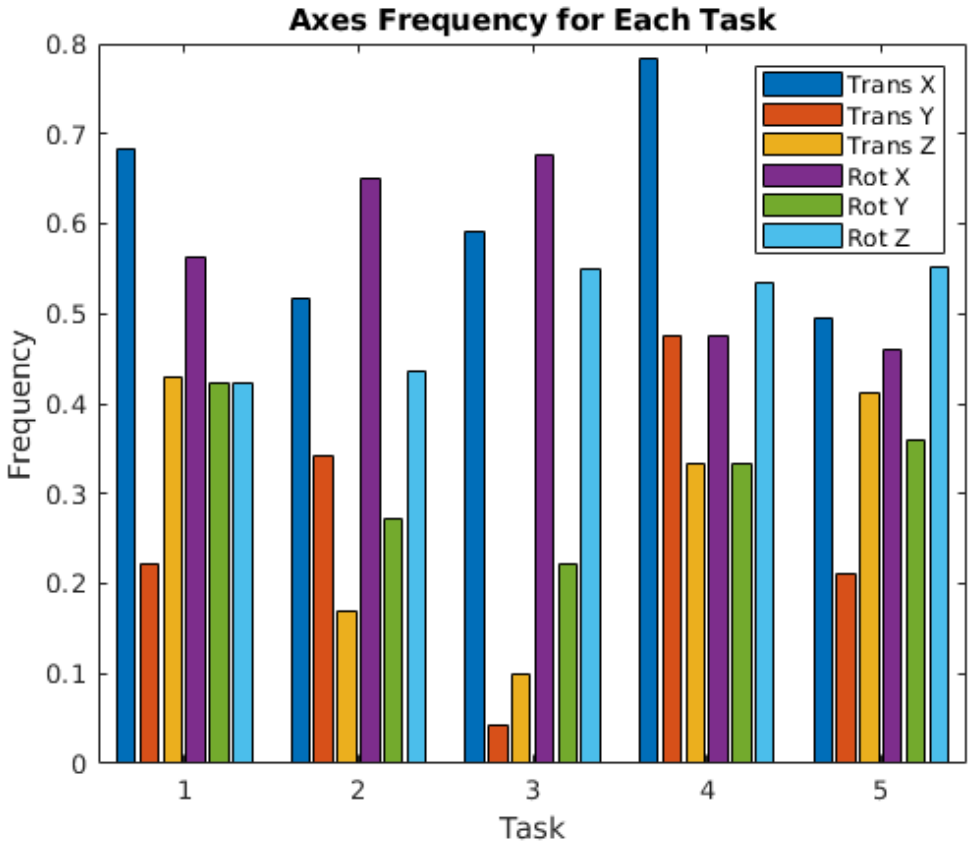
```
xlabel('Task')
ylabel('Frequency')
title('Rotation Axis Y')
hold on
errorbar(1:5, means(:,5), stds(:,5),'LineStyle','None');
hold off

subplot(2,3,6)
scatter(1:5, means(:,6), 'filled');
axis([0.8 5.2 0 1])
xlabel('Task')
ylabel('Frequency')
title('Rotation Axis Z')
hold on
errorbar(1:5, means(:,6), stds(:,6),'LineStyle','None');
hold off

sgtitle('Axis Frequency vs Task')
```



Bar Graph of Axis Frequencies for Each Task

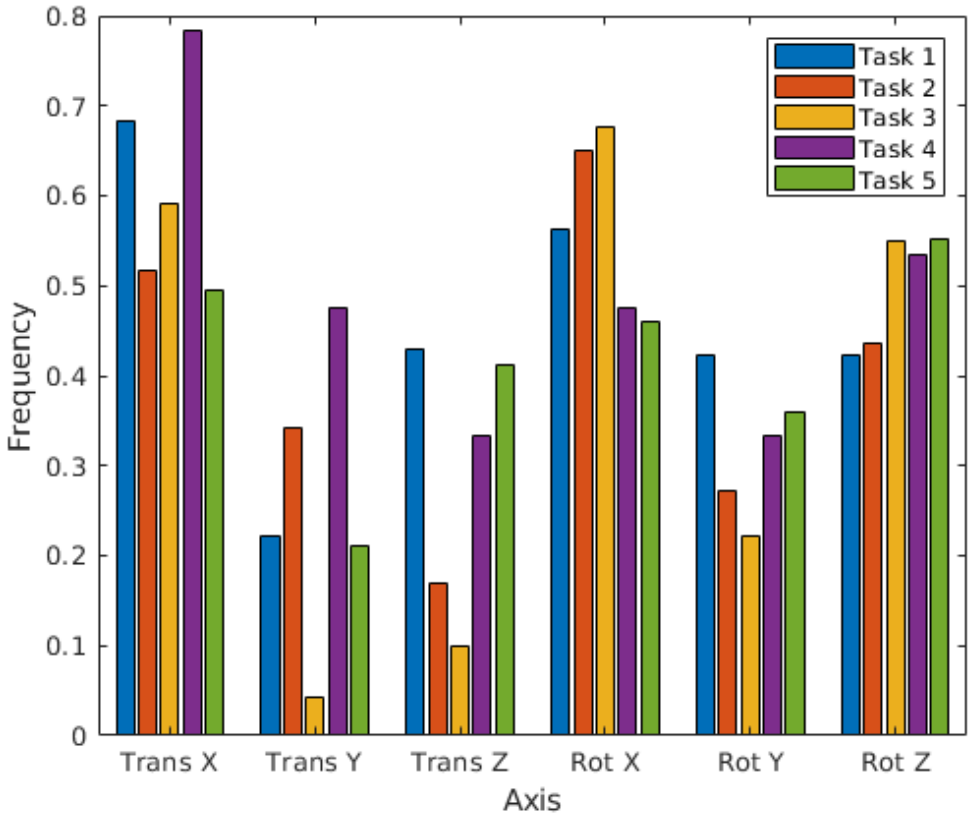
```
figure(2)
bar(means)
% hold on
```

```
% errorbar(1:5, means, stds, 'LineStyle', 'None')
% hold off
title('Axes Frequency for Each Task')
xlabel('Task')
ylabel('Frequency')
legend('Trans X','Trans Y', 'Trans Z', 'Rot X', 'Rot Y', 'Rot Z')
```



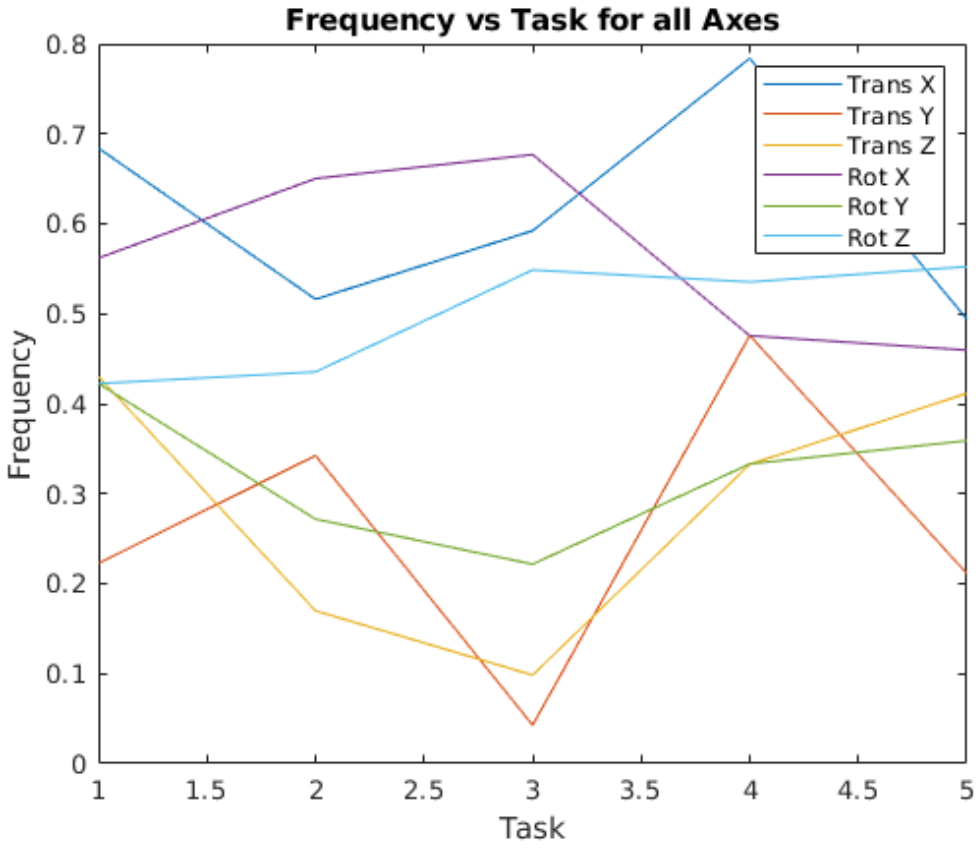
Bar Graph -> Compare

```
figure(3)
bar(means')
xticklabels({'Trans X' 'Trans Y', 'Trans Z', 'Rot X', 'Rot Y', 'Rot
 Z'})
xlabel('Axis')
ylabel('Frequency')
legend('Task 1', 'Task 2', 'Task 3', 'Task 4', 'Task 5')
```



Axis Frequency vs Task (Overlay)

```
figure(4)
plot(1:5, means)
title('Frequency vs Task for all Axes')
xlabel('Task')
ylabel('Frequency')
legend('Trans X','Trans Y', 'Trans Z', 'Rot X', 'Rot Y', 'Rot Z')
```



Published with MATLAB® R2019a