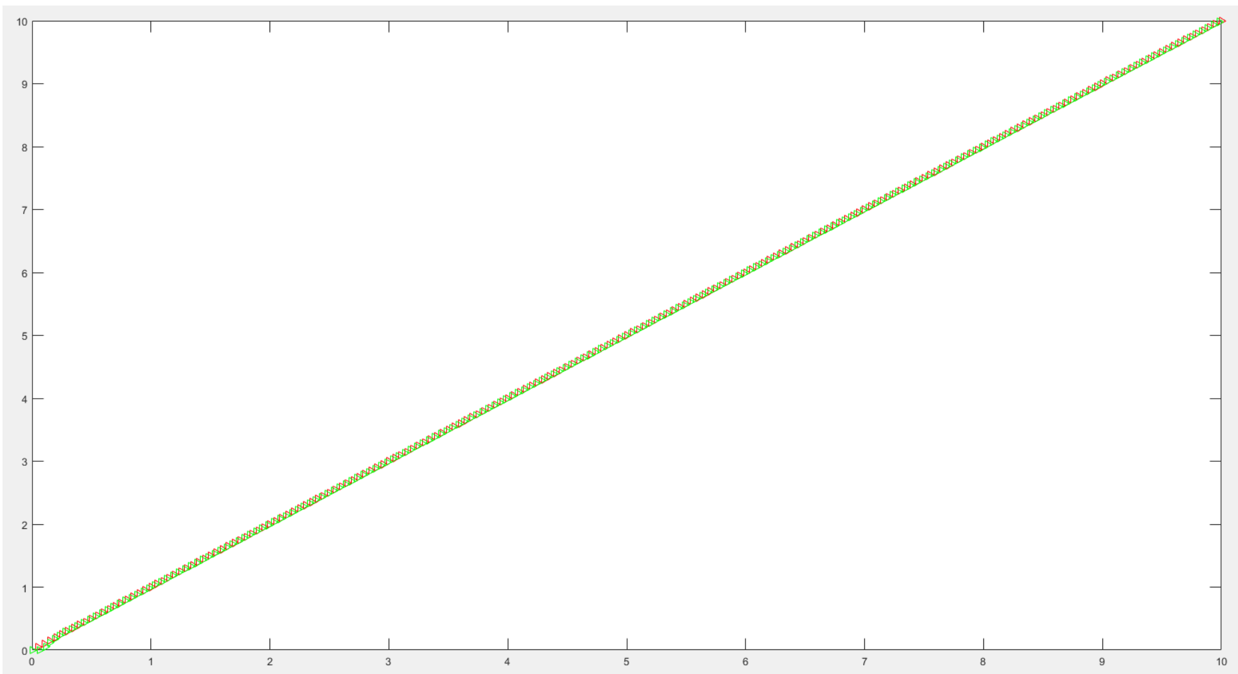
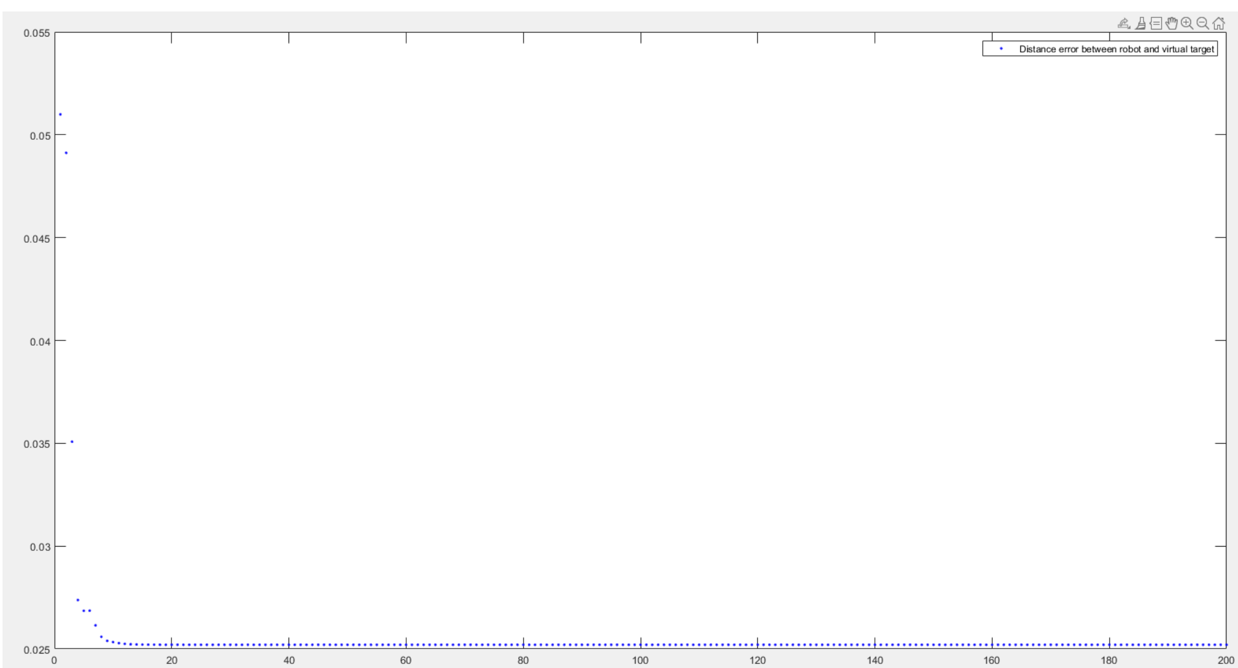
Madeline Veric
Project 2 Report
CPE 471
04/04/2024

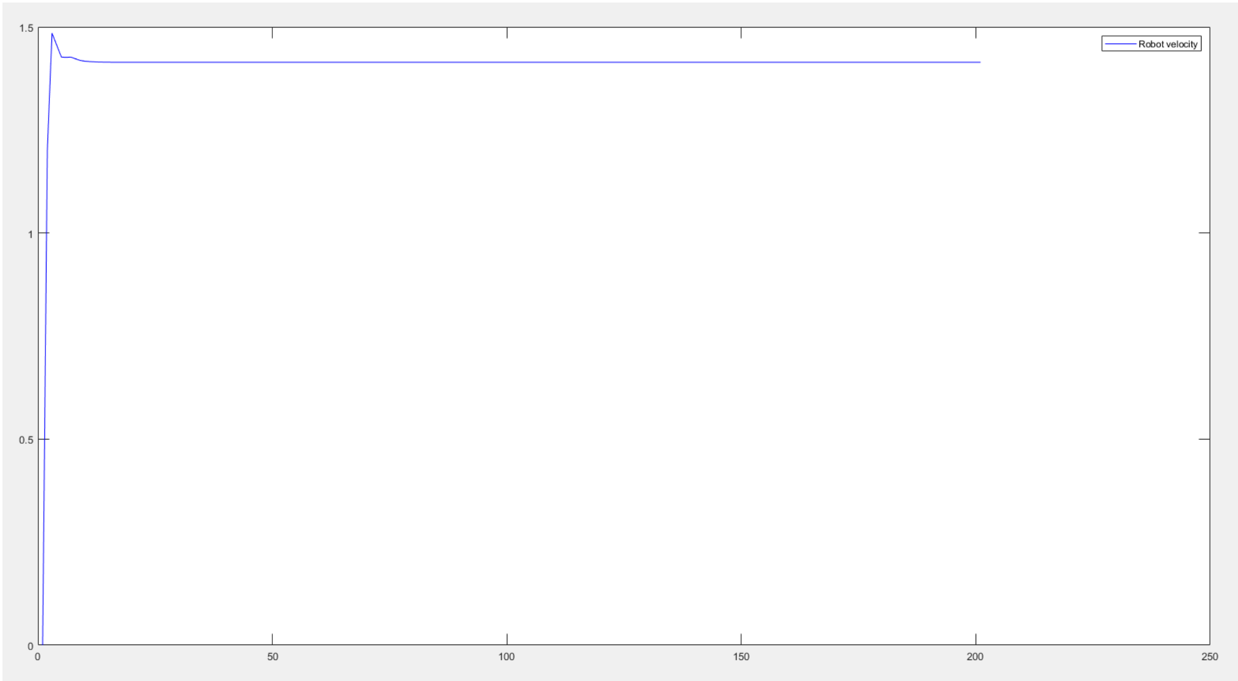
3a) Noise free environment

Linear Trajectory

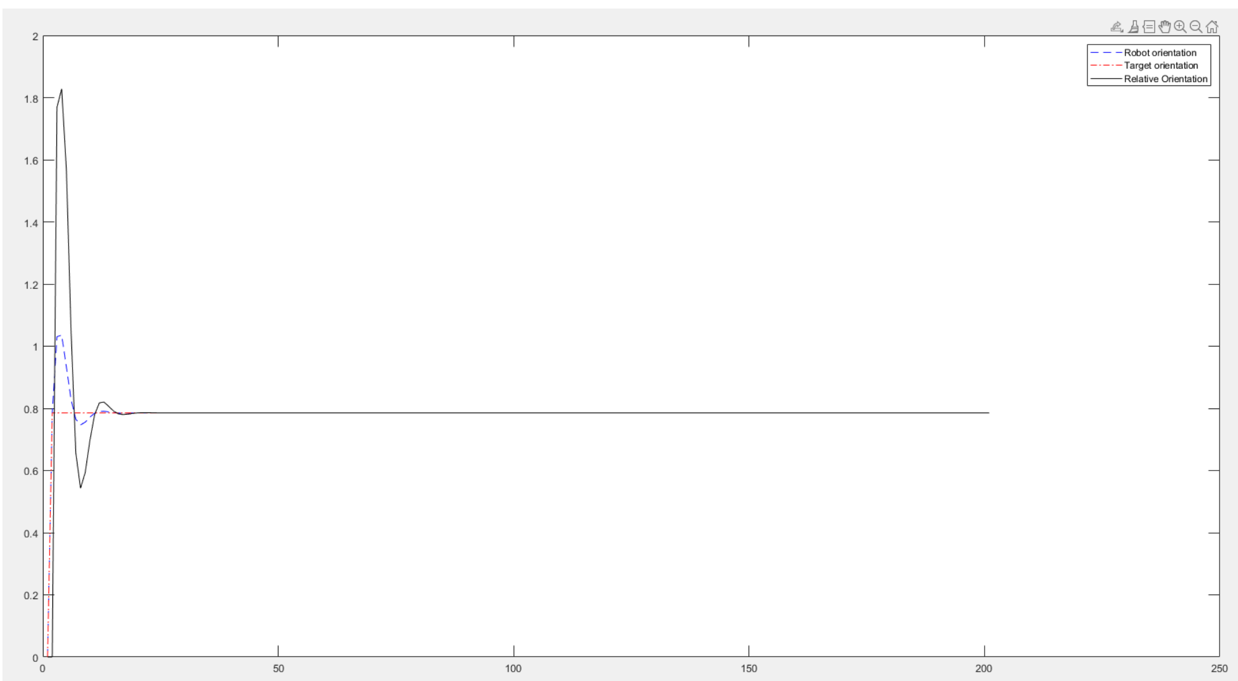
- Here we see that the tracking results in a straight line in a linear trajectory where most similar to a graph like y = x. See below for the given graphs.







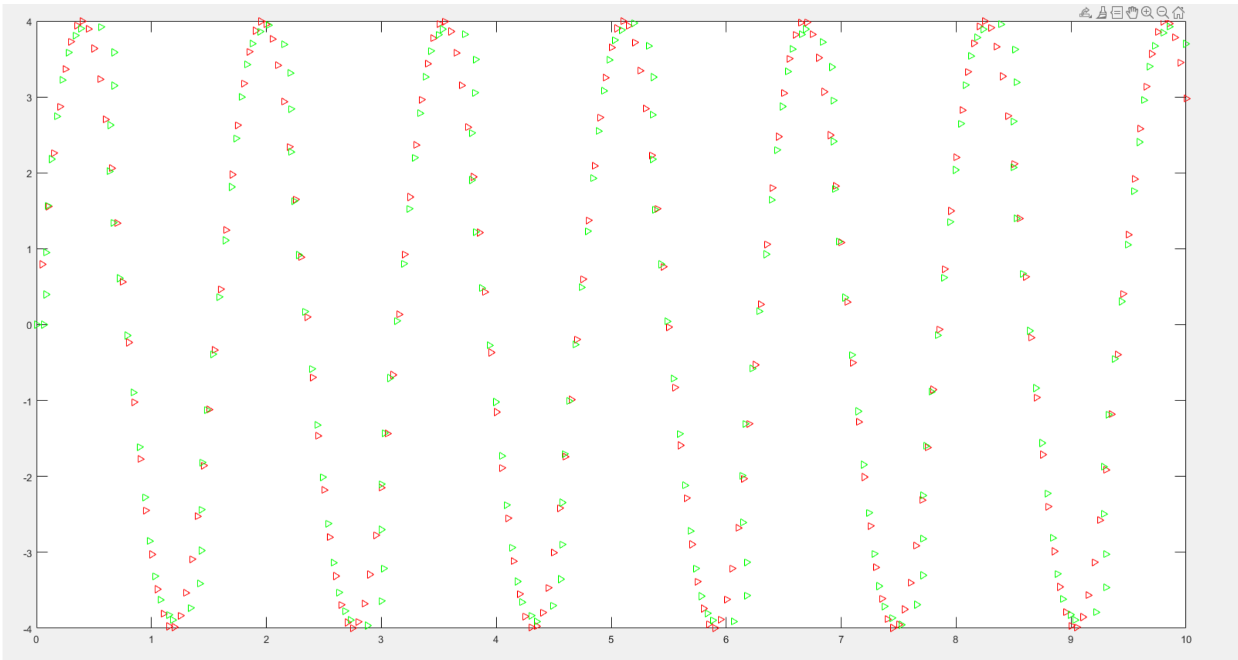
Robot velocity
1.5
1
0.5
0
0
50
100
150
200
250

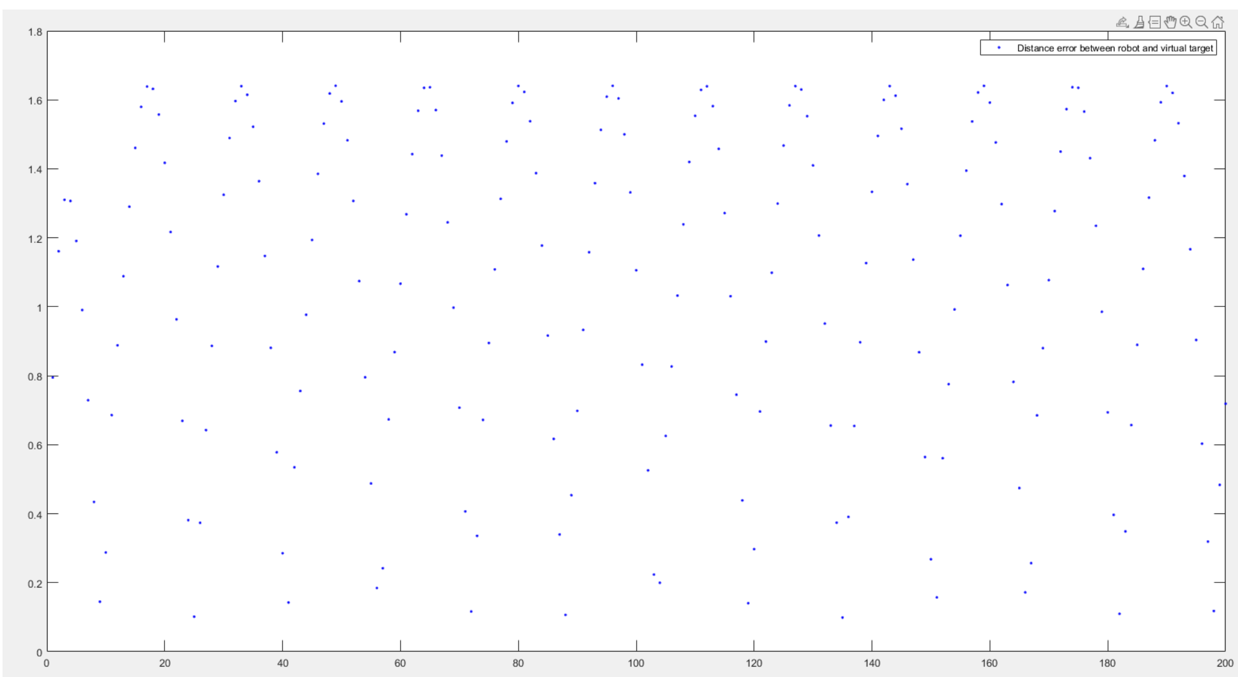


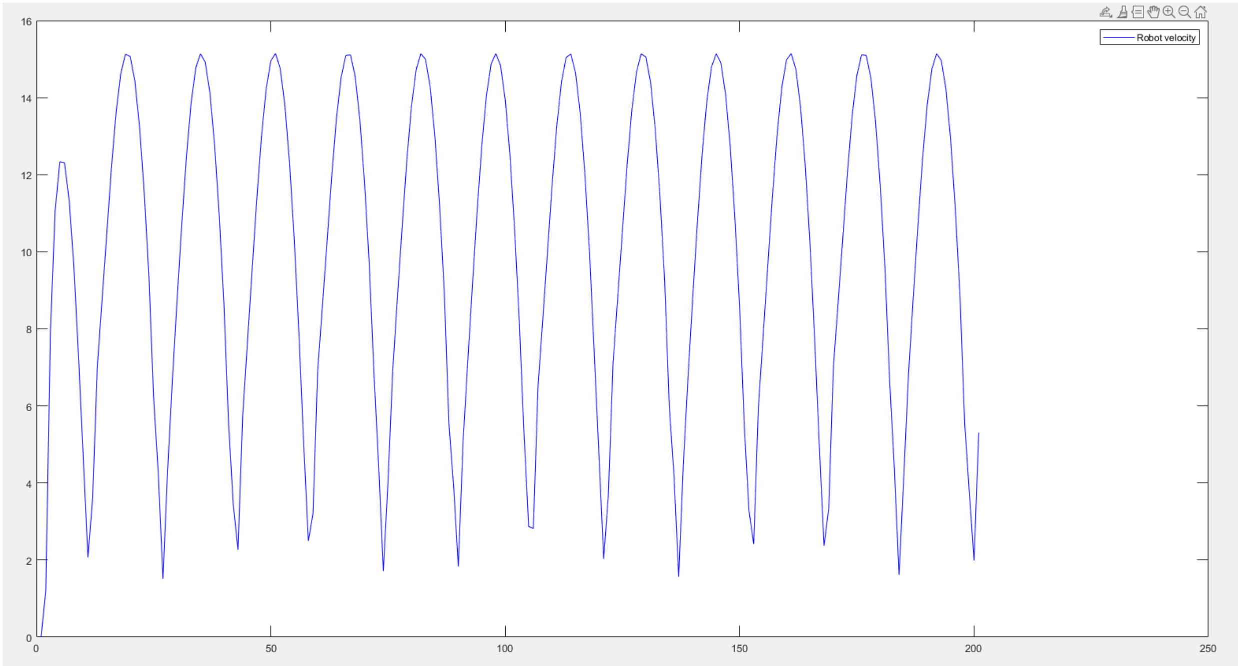
Robot orientation
Target orientation
Relative Orientation
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
250

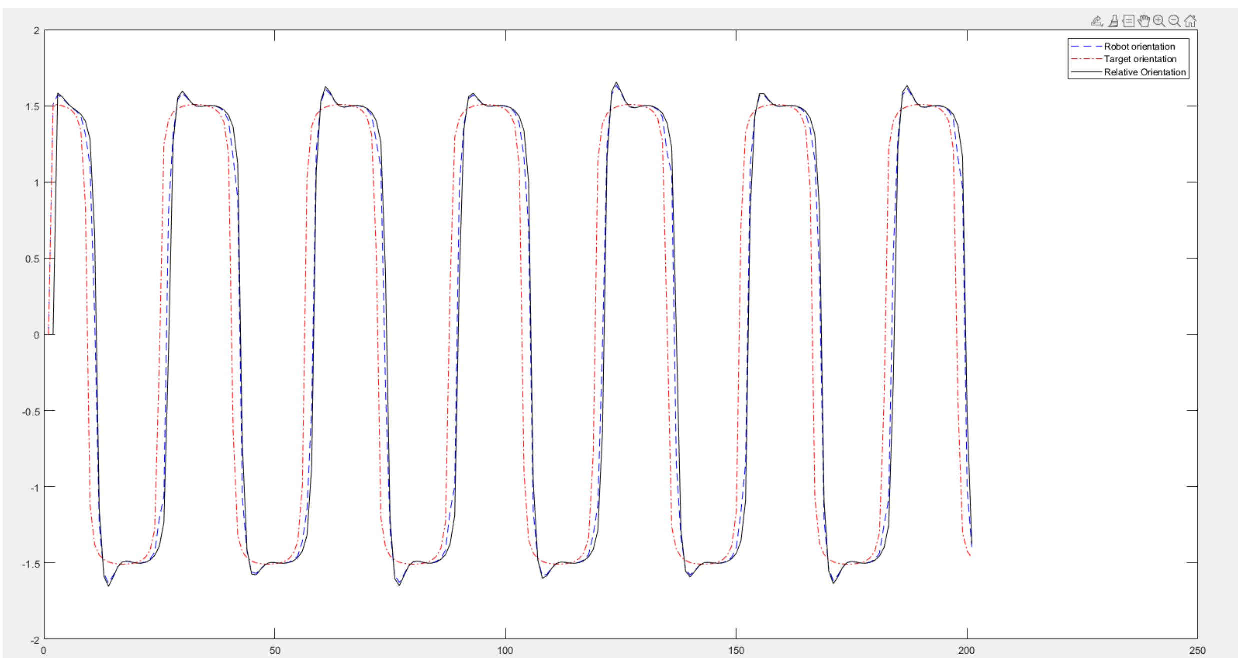
Sine Trajectory

- Here we see that the tracking results in a sine trajectory most similar to a graph like y = 4sin(4x). See below for the given graphs.





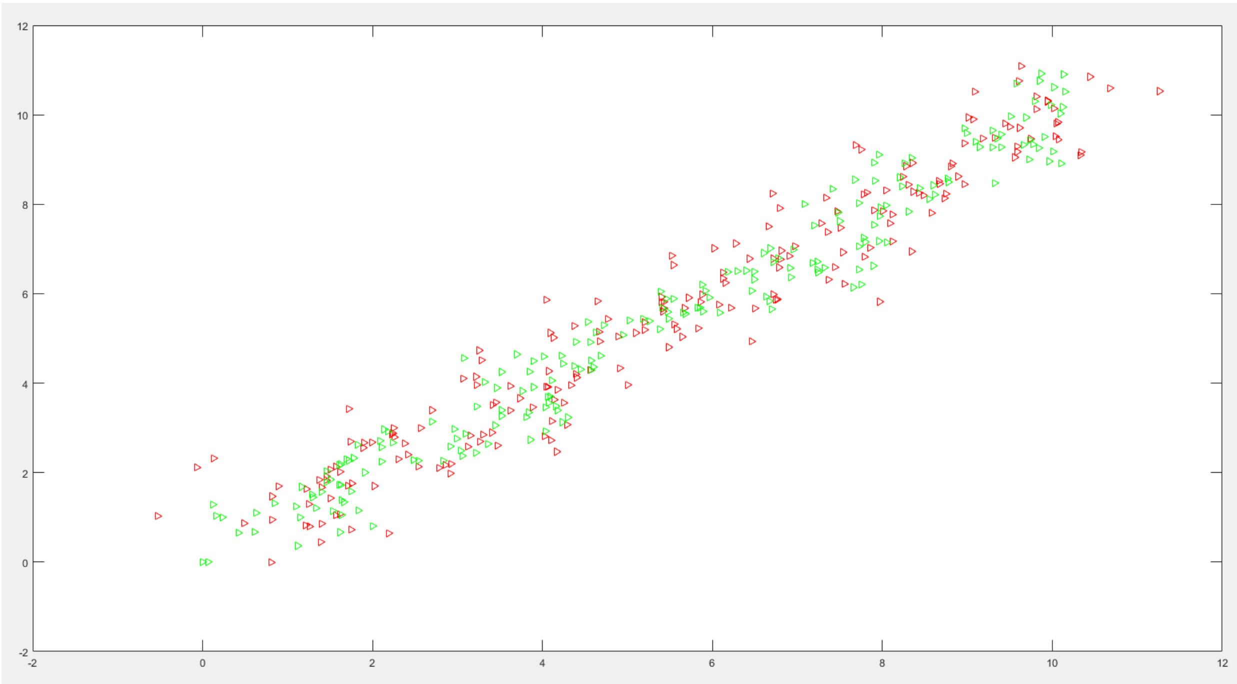

Robot velocity

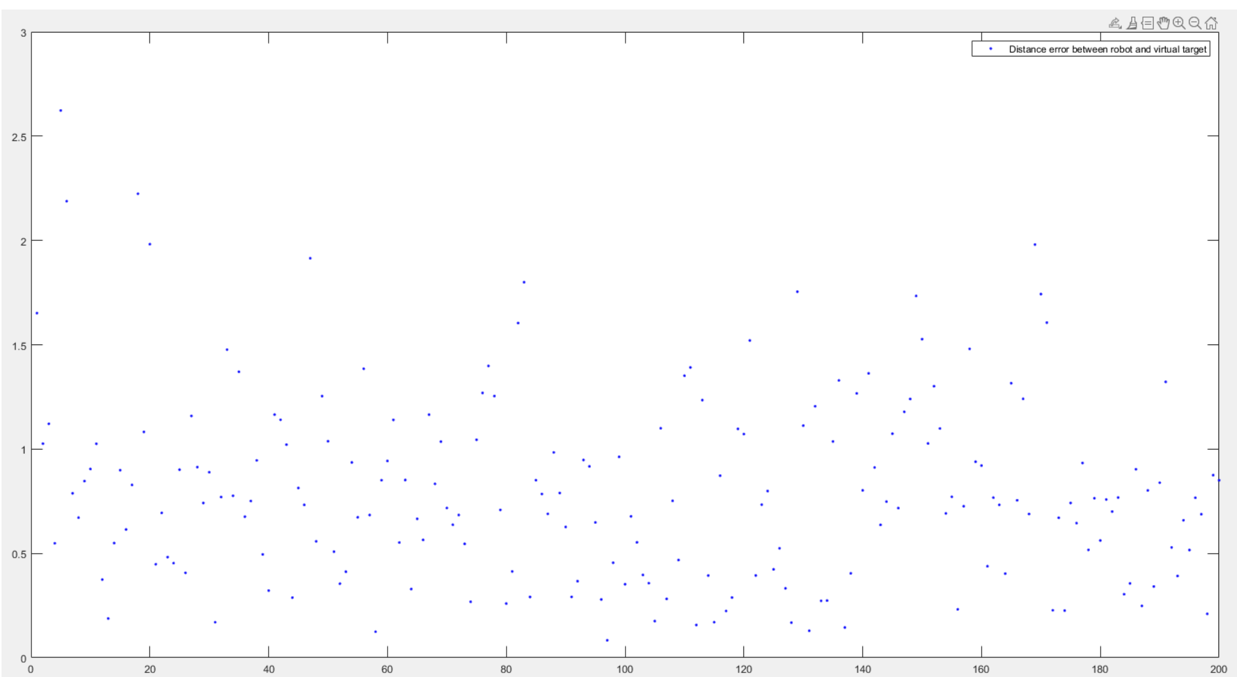

Robot orientation
Target orientation
Relative Orientation

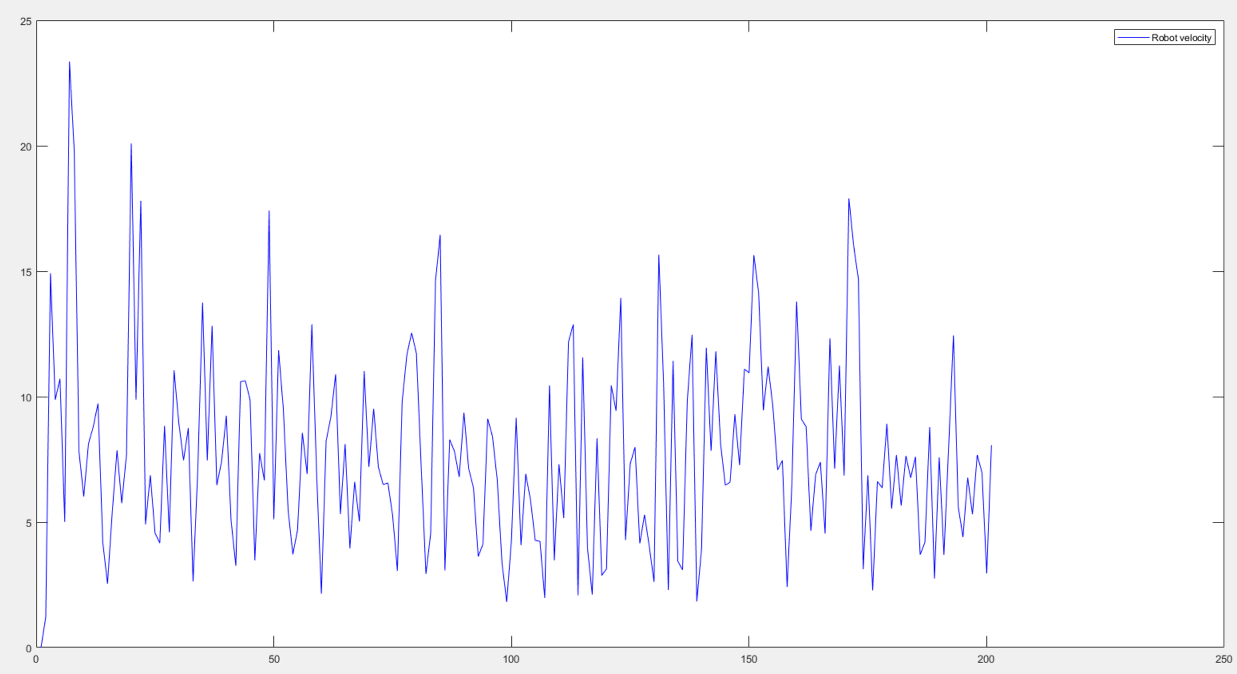
3b) Noisy environment

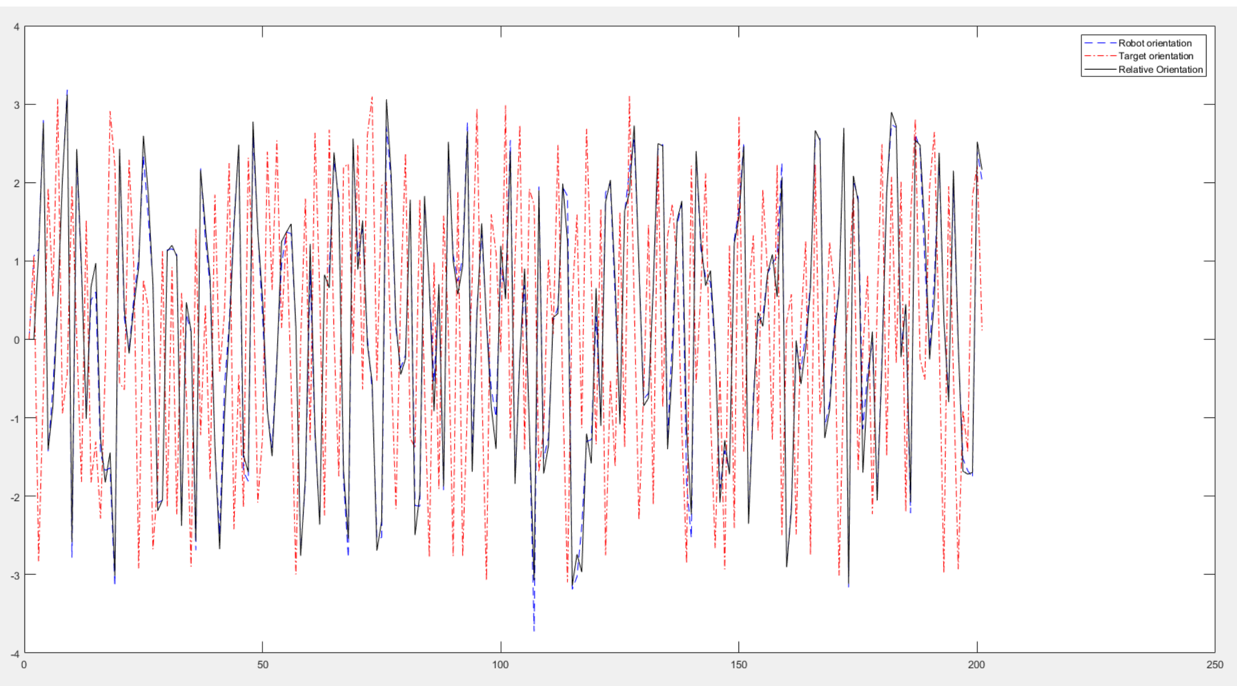
Linear Trajectory

- Here we see that the tracking results in a straight line in a linear trajectory most similar to a graph line y = x with noise. See below for the given graphs.



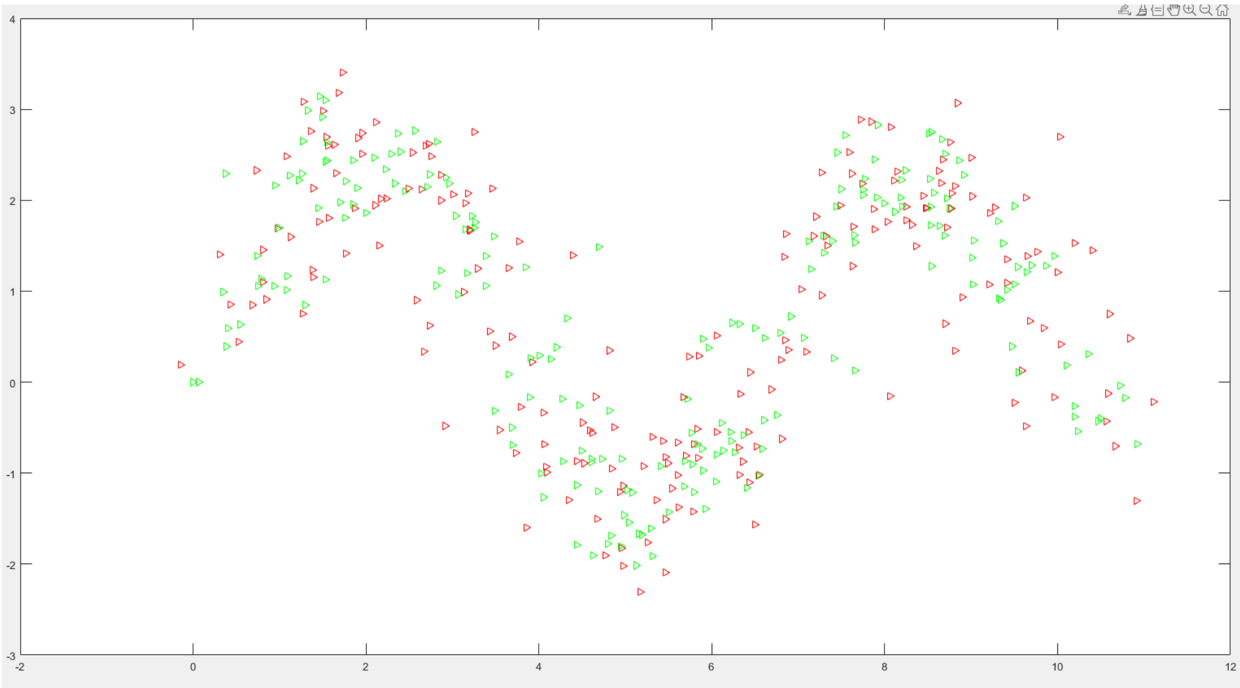




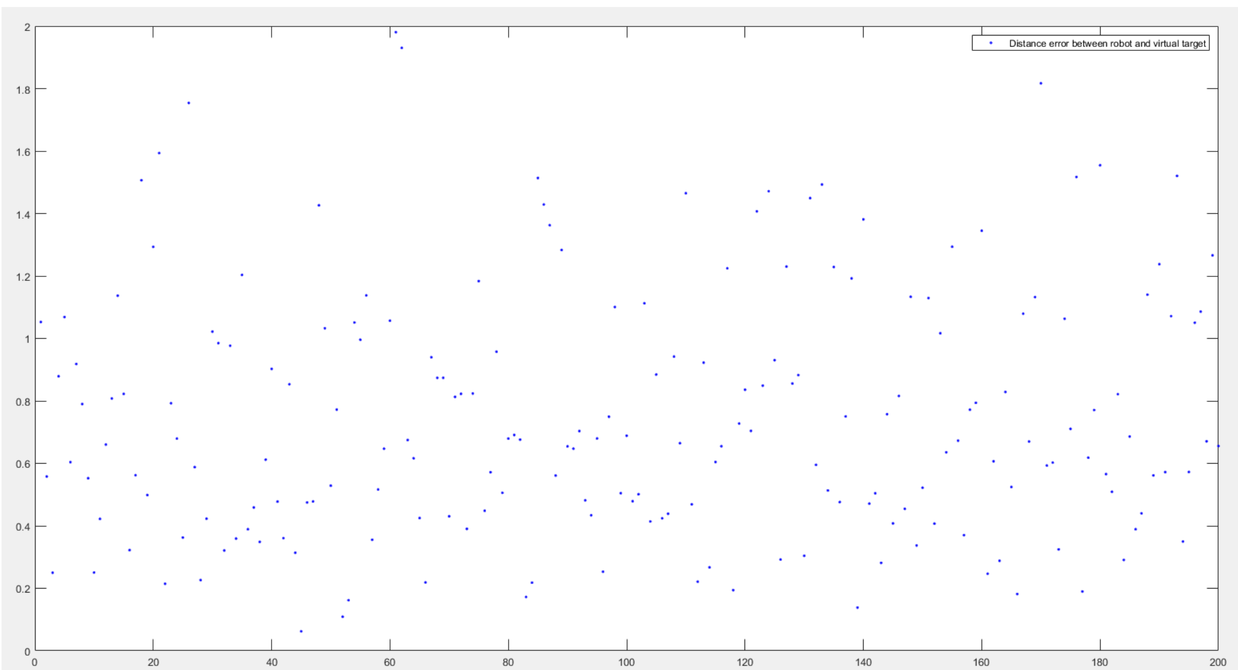


Sine Trajectory

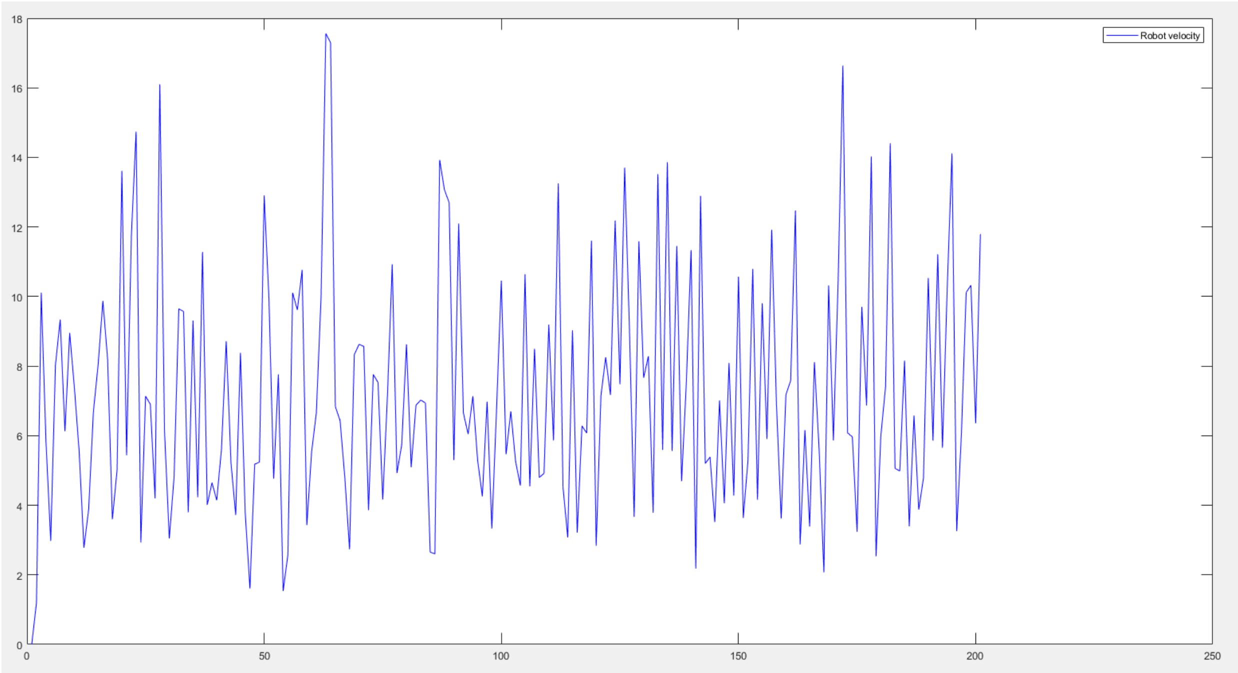
- Here we see that the tracking results in a sine trajectory most similar to a graph like y = 2sin(x) with noise. See below for the given graphs.

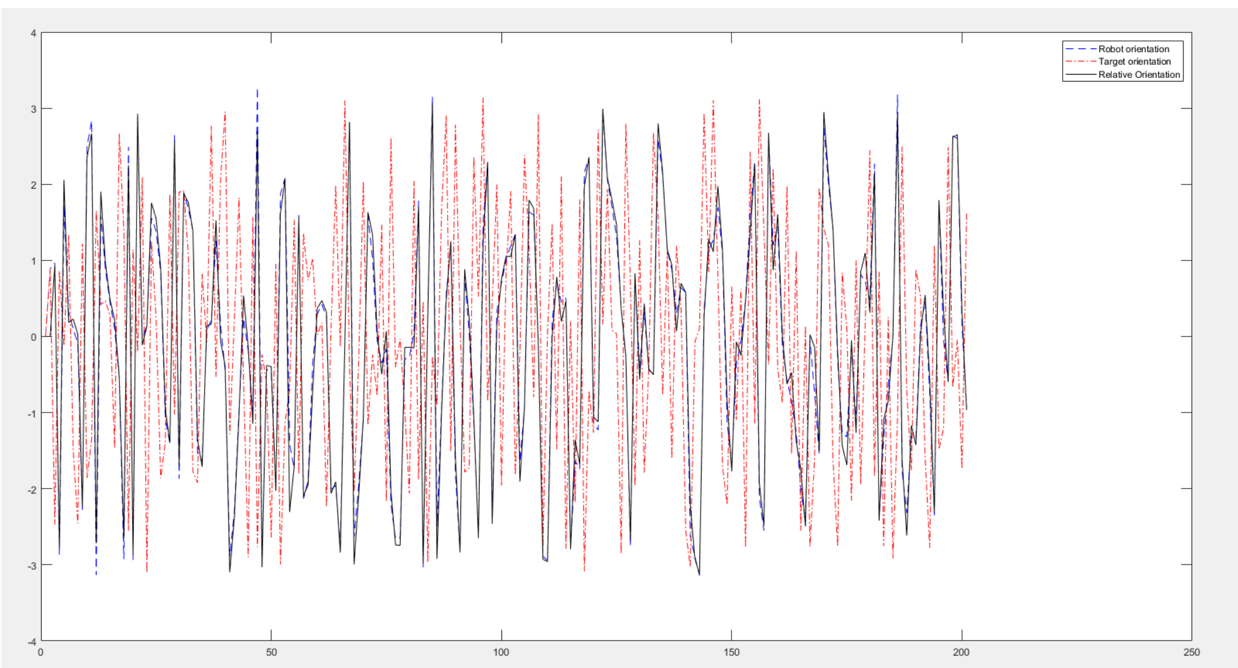


4
3
2
1
0
-1
-2
-3
-2
0
2
4
6
8
10
12



Distance error between robot and virtual target
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
20
40
60
80
100
120
140
160
180
200





Instructions for running code:
Uncomment 3 lines under desired Trajectory (with or without noise).
These lines should be qv_x, qv_y, and qv(i,:).
Then Just run the program and it should produce proper graphs.
All figures in this report can be found in the folders in the zip file.

Appendix Code

```
% CPE470/670 Project 2: Potential Field Path Planning
% ========Set parameters for simulation============
clc,clear
close all
n = 2; % Number of dimensions
delta_t = 0.05; % Set time step
t = 0:delta_t:10;% Set total simulation time
lambda = 8.5; % Set scaling factor of attractive potential field
vr_max = 50; % Set maximum of robot velocity
%===================Set VIRTUAL TARGET=================
qv = zeros (length(t),n); %Initial positions of virtual target
pv = 1.2; %Set velocity of virtual target
theta_t = zeros (length(t),1); % Initial heading of the virtual target
%===========Set ROBOT =================
%Set initial state of robot (robot)
qr = zeros (length(t),n); %initial position of robot
v_rd = zeros (length(t),1); %Initial velocity of robot
theta_r = zeros (length(t),1); % Initial heading of the robot
%============Set relative states between robot and VIRTUAL TARGE
T=================
qrv = zeros (length(t),n); %Save relative positions  between robot and virtual
target
prv = zeros(length(t),n); %Save relative velocities between robot and virtual
target
%====Compute initial relative states between robot and virtual target====
qrv(1,:) = qv(1,:) - qr(1,:);%Compute the initial relative position
%Compute the initial relative velocity
prv(1,:) = [pv*cos(theta_t(1))-v_rd(1)*cos(theta_r(1)),
pv*sin(theta_t(1))-v_rd(1)*sin(theta_r(1))];
%====Set noise mean and standard deviation====
noise_mean = 0.5;
noise_std = 0.5; %try 0.2 also
%=========MAIN PROGRAM==================
for i =2:length(t)
   %++++++++++CIRCULAR TRAJECTORY+++++++++++
   %Set target trajectory moving in CIRCULAR trajectory WITHOUT noise
   %qv_x = 60 - 15*cos(t(i));
   %qv_y = 30 + 15*sin(t(i));
   %qv(i,:) = [qv_x, qv_y]; %compute position of virtual target
   %Set target trajectory moving in CIRCULAR trajectory WITH noise
   %qv_x = 60 - 15*cos(t(i))+ noise_std * randn + noise_mean;
   %qv_y = 30 + 15*sin(t(i)) + noise_std * randn + noise_mean;
   %qv(i,:) = [qv_x, qv_y]; %compute position of target
   %+++++++++++++SINE TRAJECTORY++++++++++
   %set target trajectory moving in SINE trajectory WITHOUT noise
   %qv_x = t(i)
```

```
    %qv_y = 4*sin(4*t(i))
    %qv(i,:) = [qv_x, qv_y];
    %set target trajectory moving in SINE trajectory WITH noise
    %qv_x = t(i) + noise_std * randn + noise_mean;
    %qv_y = 2*sin(t(i)) + noise_std * randn + noise_mean;
    %qv(i,:) = [qv_x, qv_y];
    %++++++++++++++++++ LINEAR TRAJECTORY ++++++++++
    % set target trajectory moving in LINEAR trajectory WITHOUT noise
    %qv_x = t(i);
    %qv_y = t(i);
    %qv(i,:) = [qv_x, qv_y];
    % set target trajectory moving in LINEAR trajectory WITH noise
    %qv_x = t(i) + noise_std * randn + noise_mean;
    %qv_y = t(i) + noise_std * randn + noise_mean;
    %qv(i,:) = [qv_x, qv_y];
    %Compute the target heading
    qt_diff(i,:) = qv(i,:)- qv(i-1,:);
    theta_t(i) = atan2(qt_diff(i,2),qt_diff(i,1));
    Phi(i) = atan2(qrv(i-1,2), qrv(i-1,1));
    v_rd(i) = sqrt(pv^2 + 2*lambda*norm(qrv(i-1,:))*pv*abs(cos(theta_t(i) -
Phi(i))) + (lambda^2) * norm(qrv(i-1,:))^2);
    if(norm(v_rd(i)) >= pv)
    theta_r(i) = Phi(i) + asin((pv*sin(theta_t(i) - Phi(i)))/norm(v_rd(i)));
    end
    %======UPDATE position and velocity of robot==========
    qr(i,:) = qr(i-1,:) + v_rd(i)*delta_t*[cos(theta_r(i-1)),
sin(theta_r(i-1))];
    qrv(i,:) = qv(i,:) - qr(i,:);
    prv(i,:) = [pv*cos(theta_t(i))-v_rd(i)*cos(theta_r(i)),
pv*sin(theta_t(i))-v_rd(i)*sin(theta_r(i))];
    error(i) = norm(qv(i,:)-qr(i,:));
    %plot postions qv of virtual target
    plot(qv(:,1),qv(:,2),'r>')
    hold on
    %plot postions qv of robot
    plot(qr(:,1),qr(:,2),'g>')
    M = getframe(gca);
    %mov = addframe(mov,M);
end
figure(2), plot(error(2:length(t)), 'b.')
legend('Distance error between robot and virtual target')
figure(3), plot(v_rd, 'b')
legend('Robot velocity')
figure(4), plot(theta_r, '--b')
hold on
plot(theta_t, '-.r')
hold on
plot(Phi, 'k')
legend('Robot orientation', 'Target orientation', 'Relative Orientation')
```