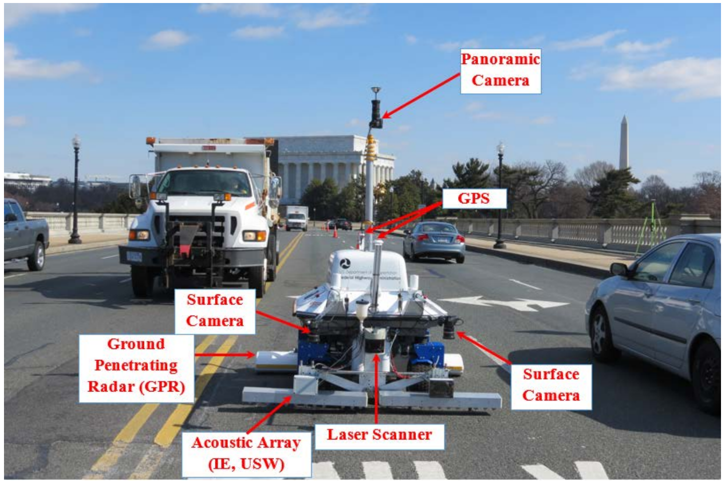
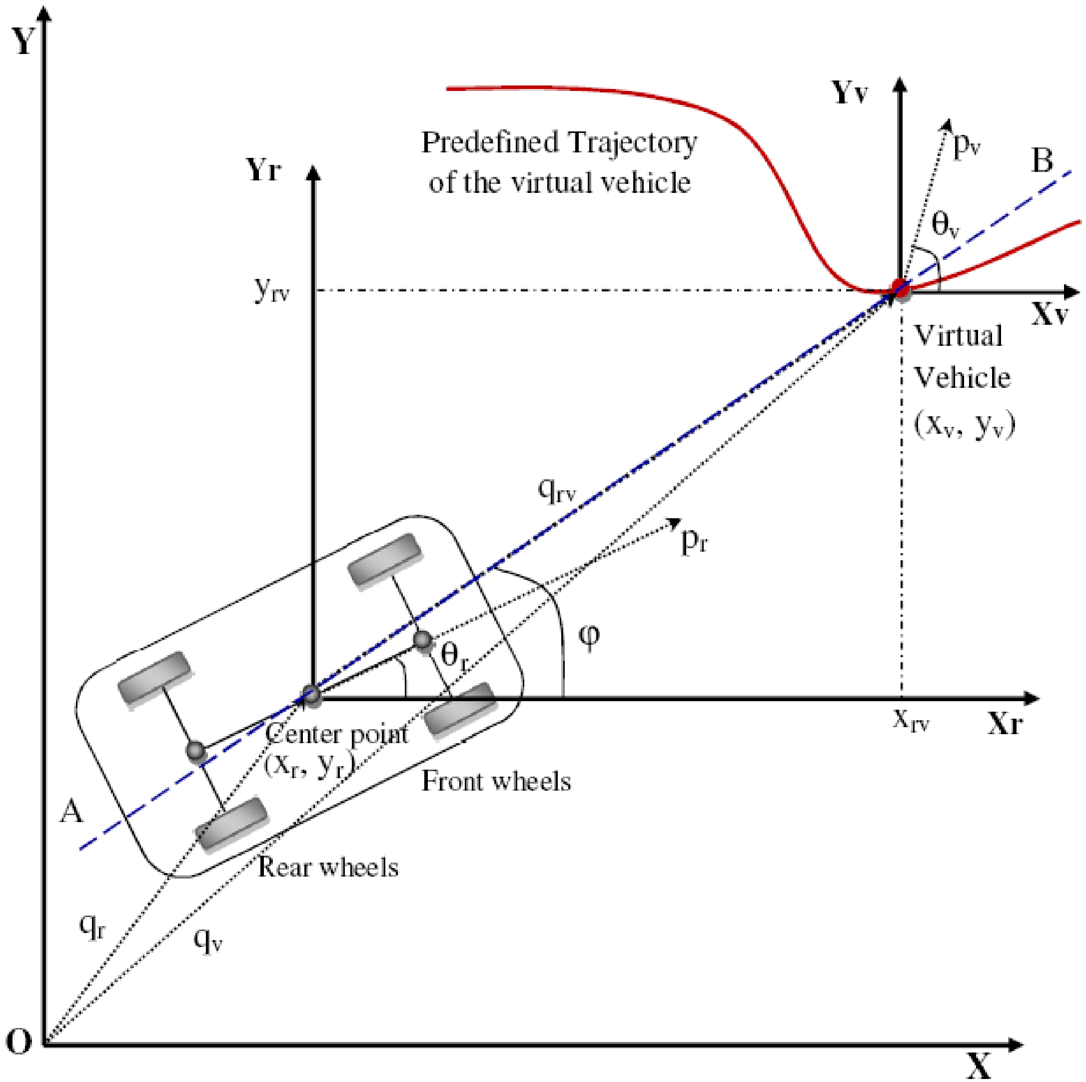
ECEN 470/670--Mobile Robotics

Project 2-Instruction



Dr. Hung La
Department of Computer Science and Engineering
University of Nevada, Reno
Fall 2016

Navigation based on Virtual Target/Vehicle



Robot Dynamics

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} v\cos\theta_r \\ v\sin\theta_r \\ \omega \end{bmatrix}$$

Relative Position

$$\begin{cases} x_{rv} = x_v - x_r + \epsilon_r^x \\ y_{rv} = y_v - y_r + \epsilon_r^y, \end{cases}$$

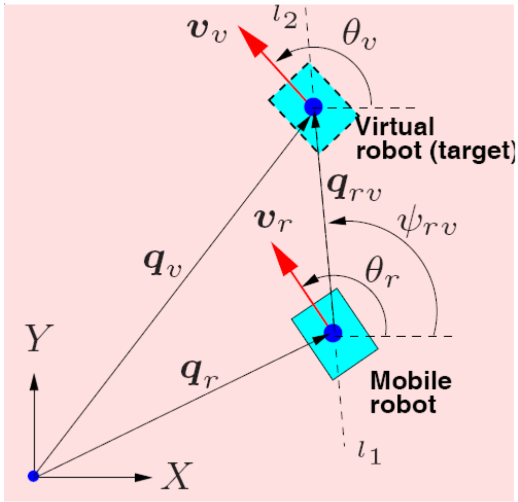
Relative Heading

$$\varphi = atan2(y_{rv}, x_{rv}).$$

Motion Planning Algorithm

- Potential Field Approach
  - Attractive Potential Function: $V_a = \frac{1}{2}\lambda\|q_{rv}\|^2 = \frac{1}{2}\lambda q_{rv}^T q_{rv}$
  - Static Point:

$$p_r^d = \nabla_{q_{rv}} V_a = \lambda q_{rv},$$



  - Moving Point (Virtual vehicle)

$$p_r^d = p_v + \lambda q_{rv}.$$

  - Control velocity and heading of the robot:

$$\begin{cases} \varphi = atan2(y_{rv}, x_{rv}) \\ v_r^d = \sqrt{\|p_v\|^2 + 2\lambda\|q_{rv}\|\|p_v\||cos(\theta_v - \varphi)| + \lambda^2\|q_{rv}\|^2} \\ \theta_r^d = \varphi + sin^{-1}(\frac{\|p_v\|sin(\theta_v - \varphi)}{\|p_r\|}). \end{cases}$$

- Project to x and y direction:

$$p_r^d = [p_r^x, p_r^y]^T = [v_r^d cos(\theta_r), v_r^d sin(\theta_r)]^T$$

(Omni-motion control)

- Constraints: ???

$$\theta_r^d = \varphi + sin^{-1}(\frac{\|p_v\| sin(\theta_v - \varphi)}{\|p_r\|}).$$

$\frac{\|p_v\| sin(\theta_v - \varphi)}{\|p_r\|}$ could return bigger than 1,

Therefore we need to design $\|p_r\| \geq \|p_v\|$

Code - Initialization

```
% 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;%0.05;
t = 0:delta_t:5;% set time for computing qt and
theta_t
lambda = 8.5;%scaling factor of attractive and
reputsive potential field
pr_max = 50; %Set max of robot velocity
```

Code

```
%==================Set TERGET==================
qt = zeros (length(t),n); %Initial positions of target
pt = 1.2; %Set velocity of target
theta_t = zeros (length(t),1); % Initial heading of the
target

%===========Set ROBOT ==================
%Set initial state of robot (robot)
qr = zeros (length(t),n); %initial position of robot
pr =  zeros (length(t),1); %Initial velocity of robot
theta_r = zeros (length(t),1); % Initial heading of the
robot
```

Code

```
%============Set relative states between
robot and TARGET===================

qrt = zeros (length(t),n); %Save relative
postions between robot and target
prt = zeros(length(t),n); %Save relative
velocities between robot and target
```

Code

```
%====Compute initial relative states
between robot and Target====
qrt(1,:) = qt(1,:) - qr(1,:);%Compute the
initial ralative position
%Compute the initial ralative velocity
prt(1,:) = [pt*cos(theta_t(1))-
pr(1)*cos(theta_r(1)), pt*sin(theta_t(1))-
pr(1)*sin(theta_r(1))];
```

Code

```
%====Set noise mean and standart
deviation====
noise_mean = 0.5;
noise_std = 0.1; %try 0.5 also
```

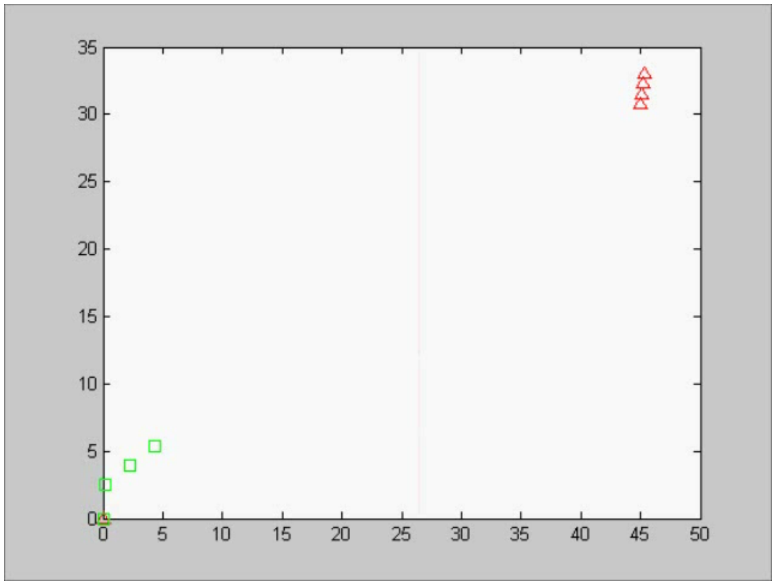
Main Program

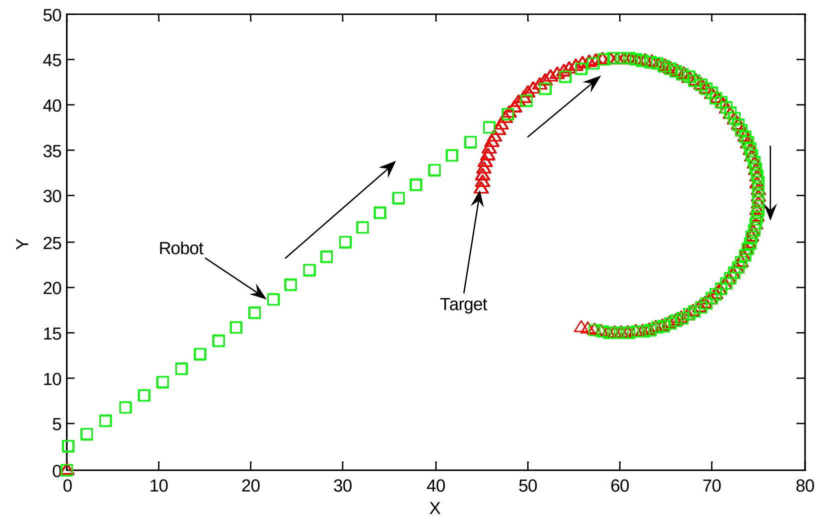
```
%=========MAIN PROGRAM===================
for i =2:length(t)
%     %Set target trajectory WITHOUT noise
%     qt_x = 60 - 15*cos(t(i));
%     qt_y = 30 + 15*sin(t(i));
%     qt(i,:) = [qt_x, qt_y]; %compute position of
target
%
    %Set target trajectory WITH noise
    qt_x = 60 - 15*cos(t(i))+ noise_std * randn +
noise_mean;
    qt_y = 30 + 15*sin(t(i)) + noise_std * randn +
noise_mean;
    qt(i,:) = [qt_x, qt_y]; %compute position of
target
```

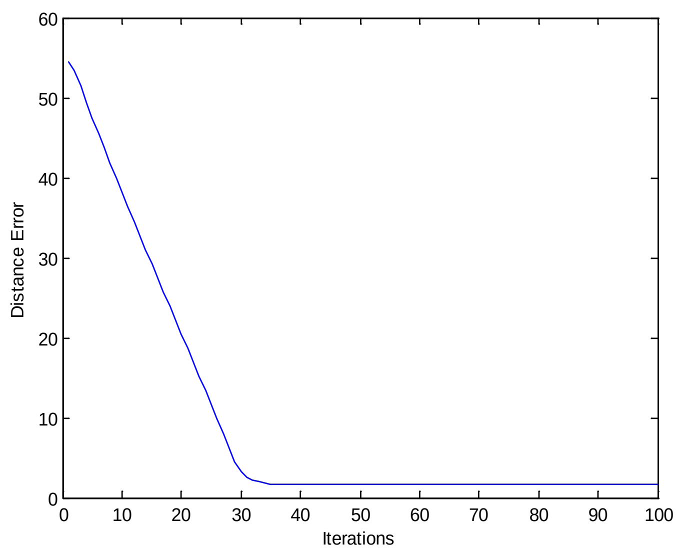
Main Program

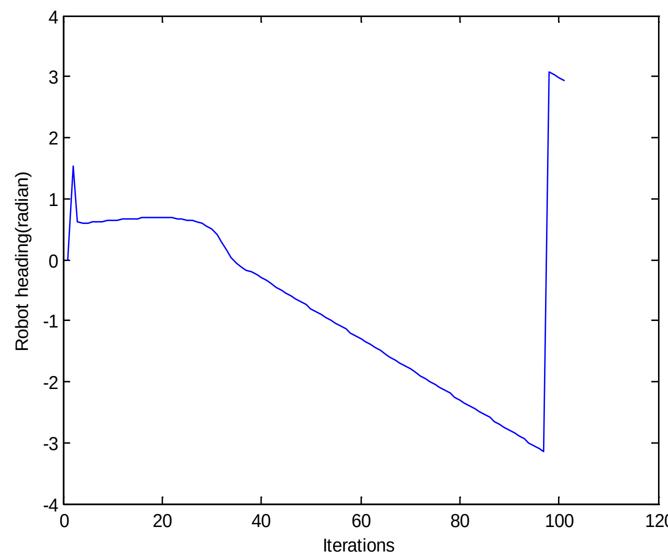
```
 ... (Your code is here)

    plot(qt(:,1),qt(:,2),'r>')
    hold on
    %plot postions qt of robot
    plot(qr(:,1),qr(:,2),'g>')
    M = getframe(gca);
end
figure(2), plot(error(2:length(t)), 'b.')
legend('Distance error between robot and target')
figure(3), plot(pr, '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')
```

Circular Path Planning (Simulation)



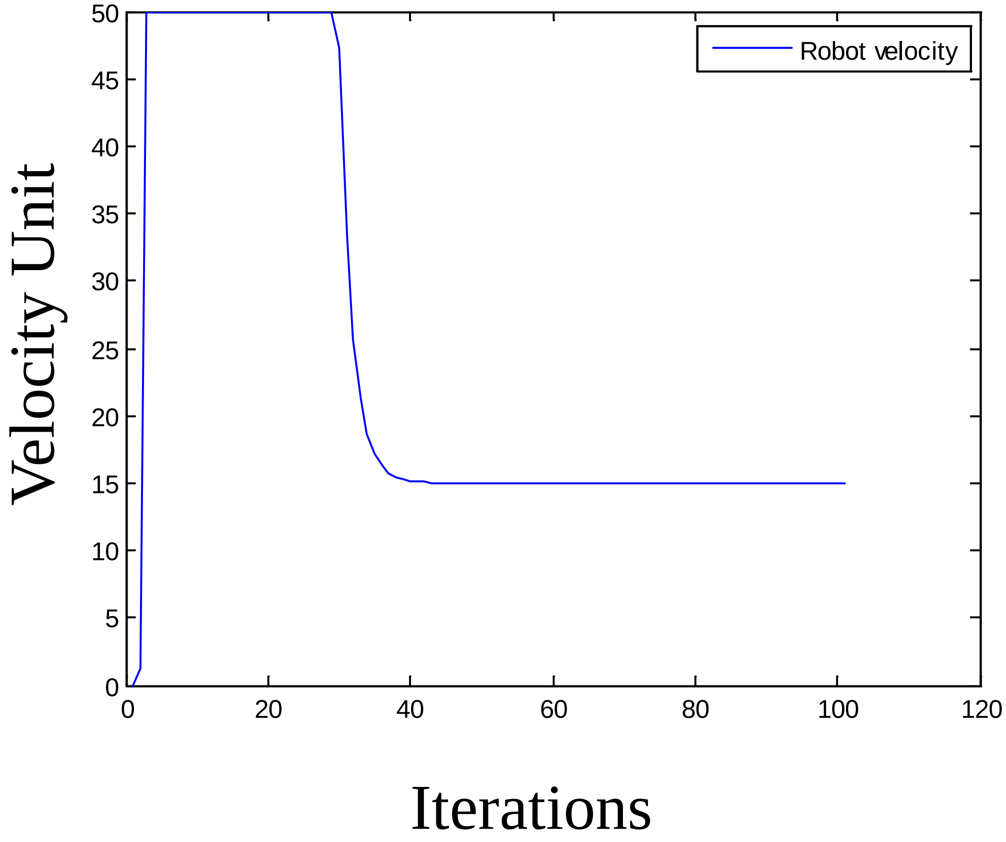


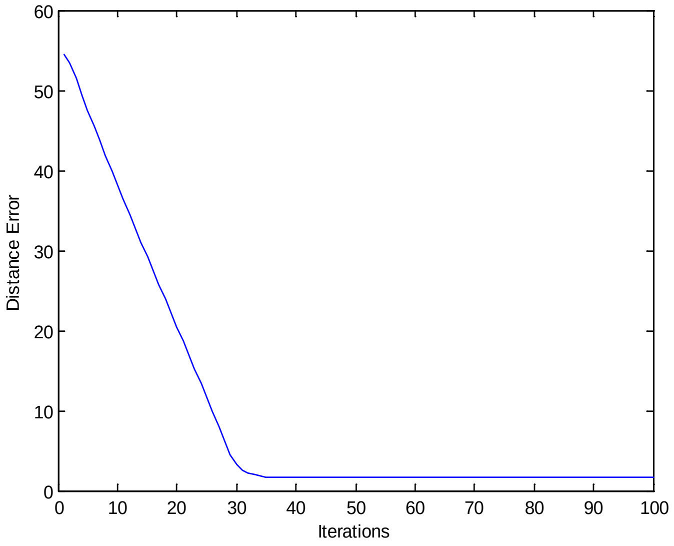




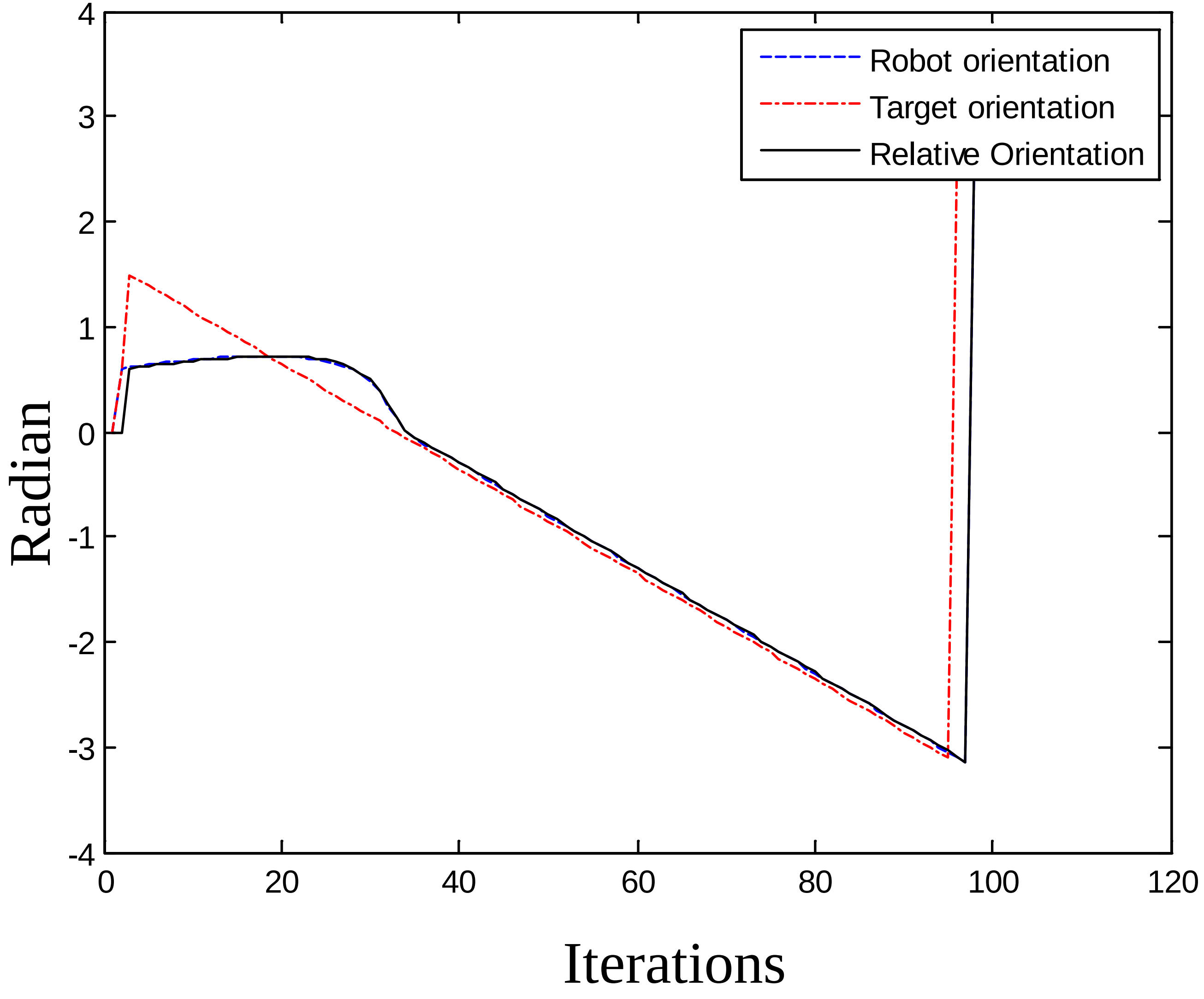
CASE 1: Robot Velocity with Limit Function

- pr(i) = min(pr(i),pr_max);

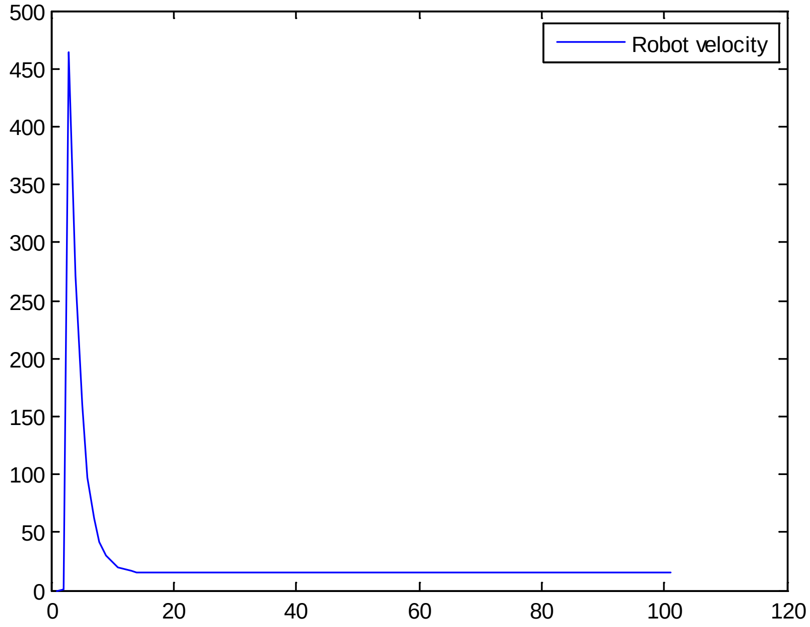




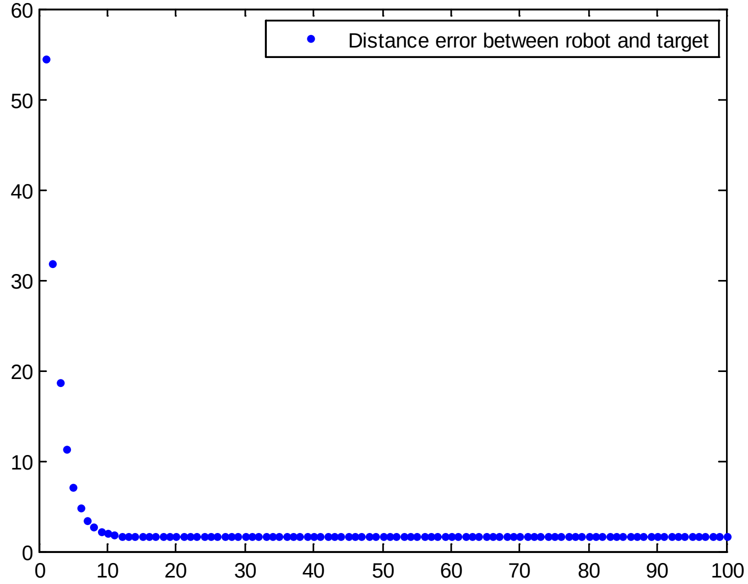
Heading/Orientation Comparison



CASE 1: Robot Velocity without Limit Function

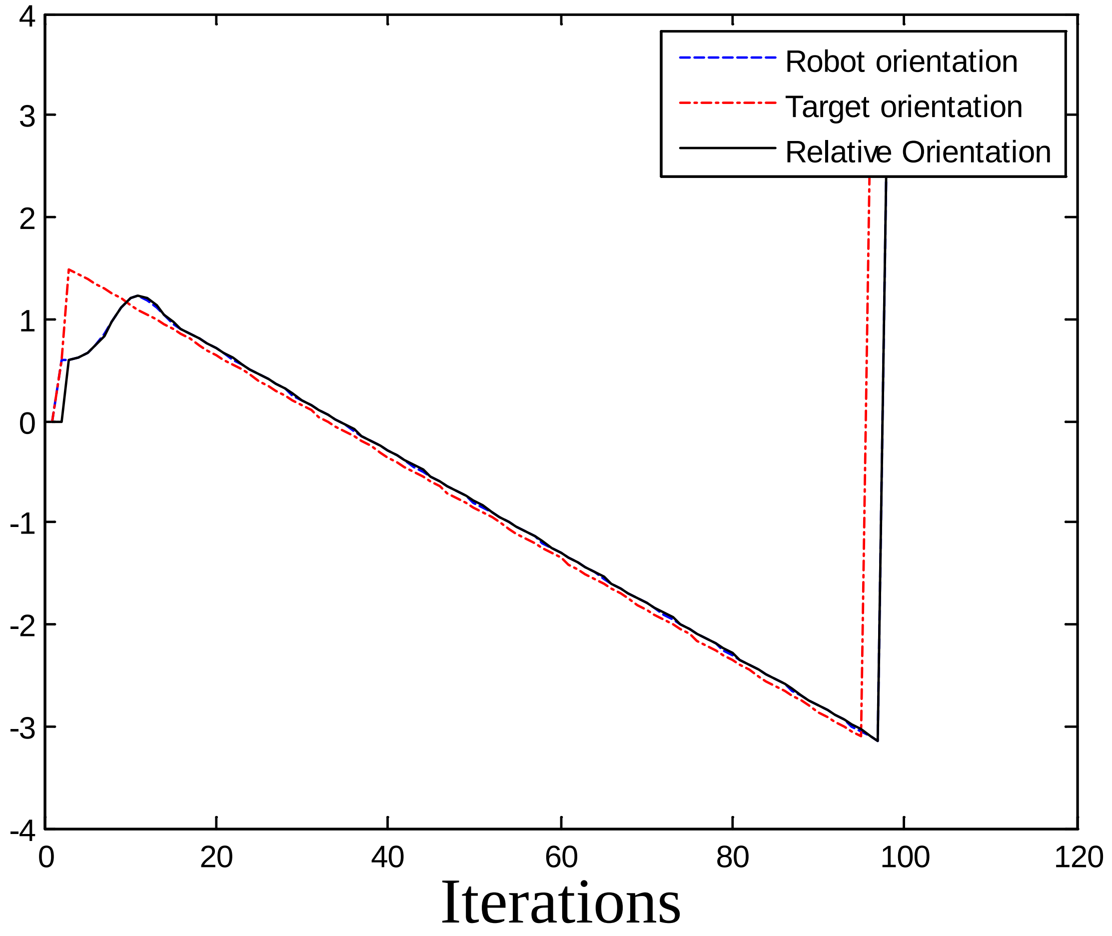


Iterations

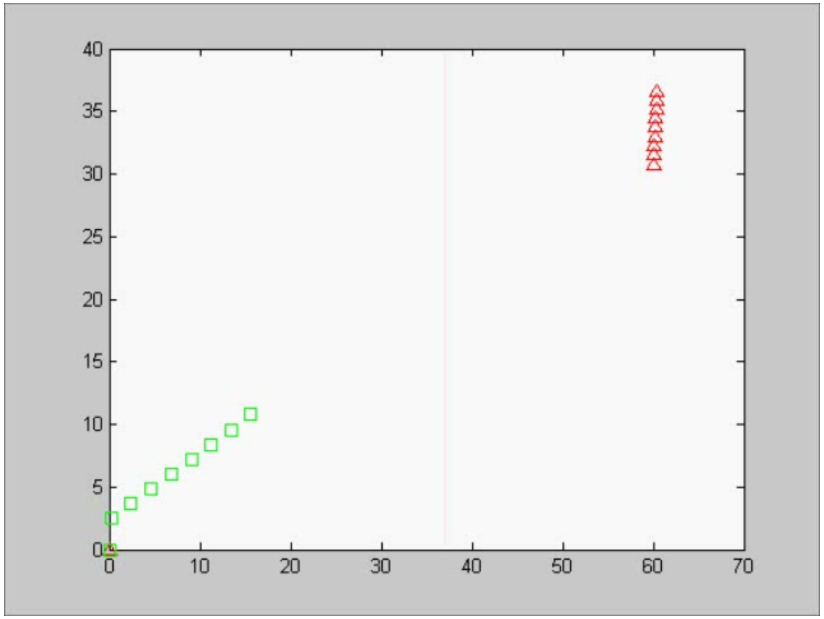


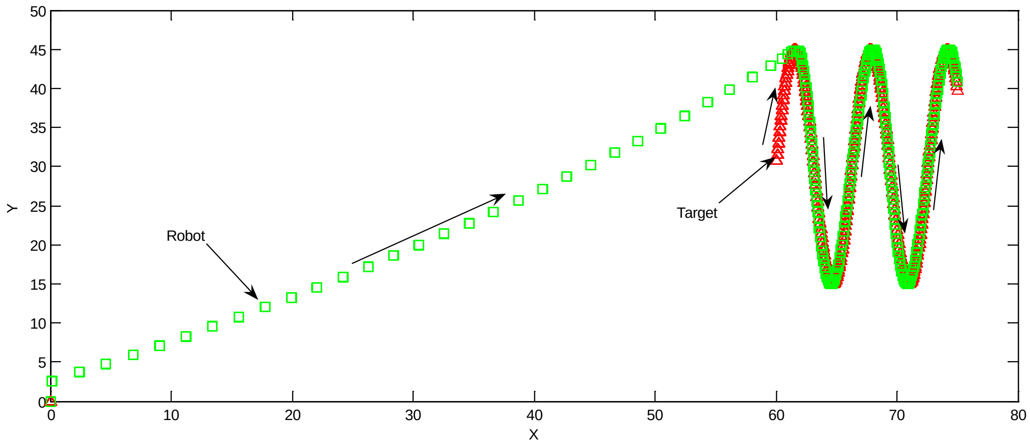
Iterations

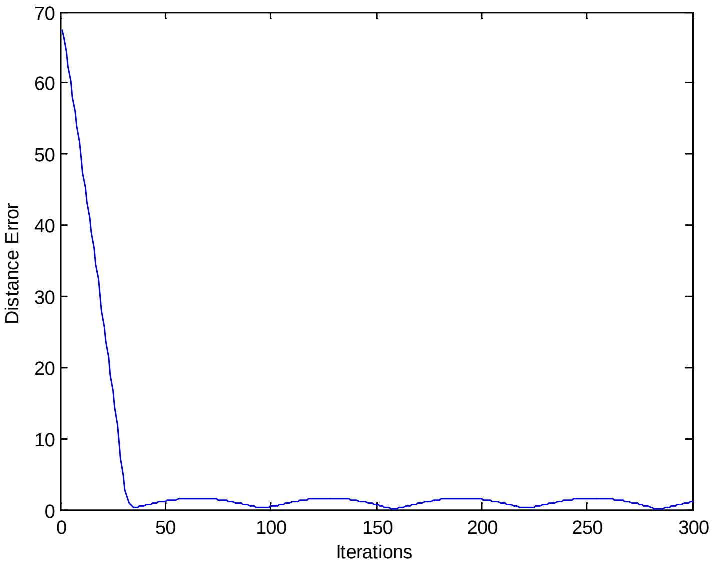
Heading/Orientation Comparison

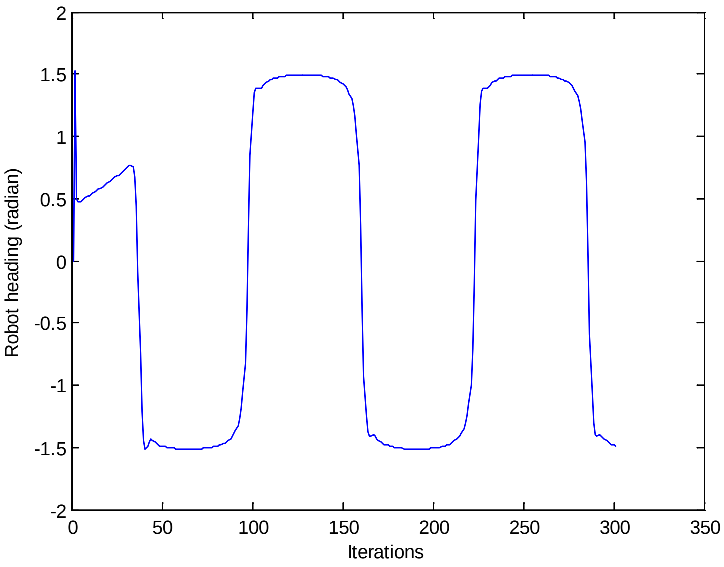


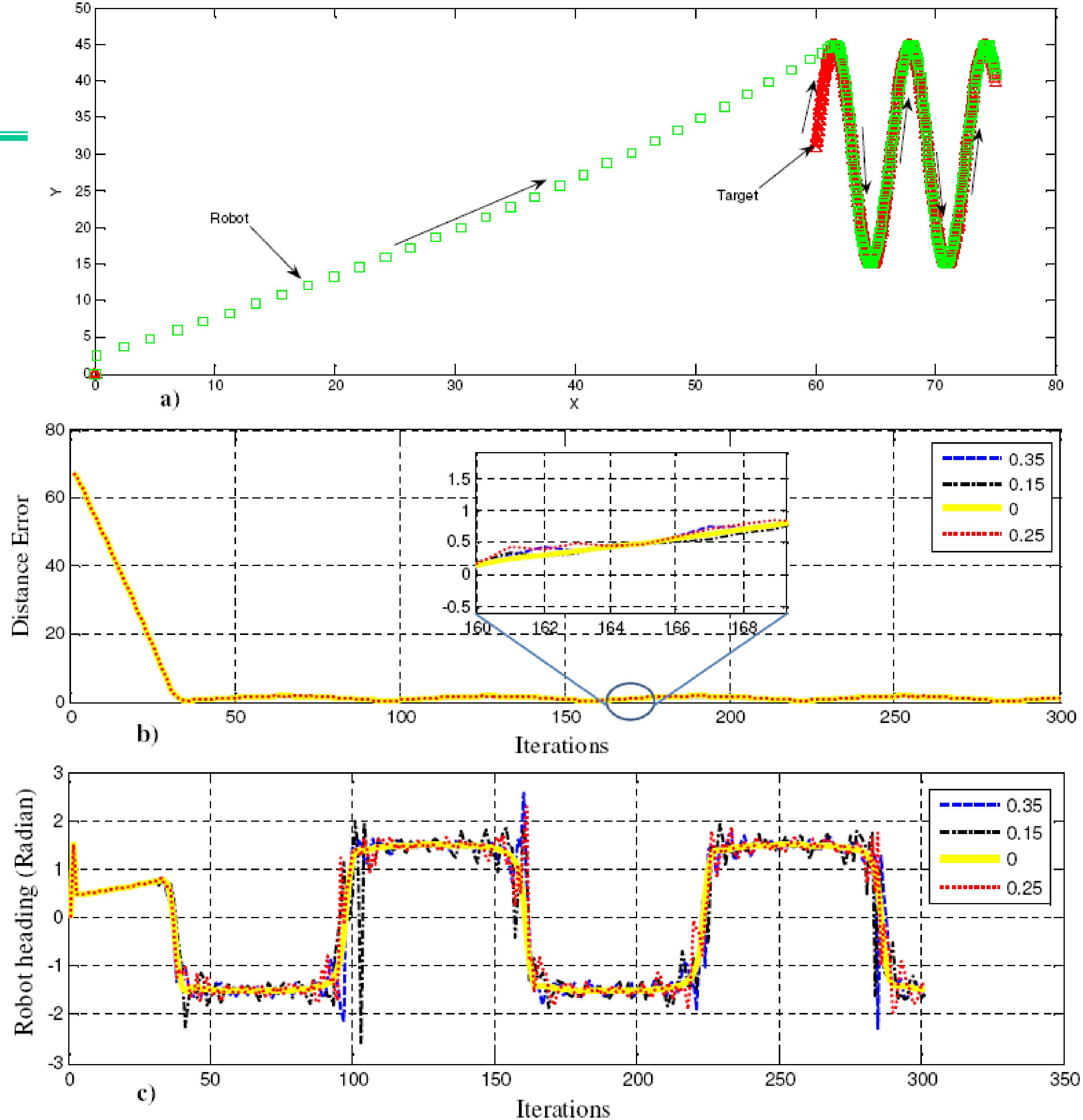
Sine Wave Path Planning (Simulation)











Robot
Target
Y
X
a)
Distance Error
Iterations
b)
0.35
0.15
0
0.25
Robot heading (Radian)
Iterations
c)