**Robotics**

**Exercise 3.1 Pose compositions**

1. Complete the *Matlab* code below (same as in lecture 3) for the robot to describe an 8mx8m square path. Initially the robot is at the left-bottom corner (the origin) heading north and moves at increments of 2m. each step (that is, after 4 steps it will reach the second corner).

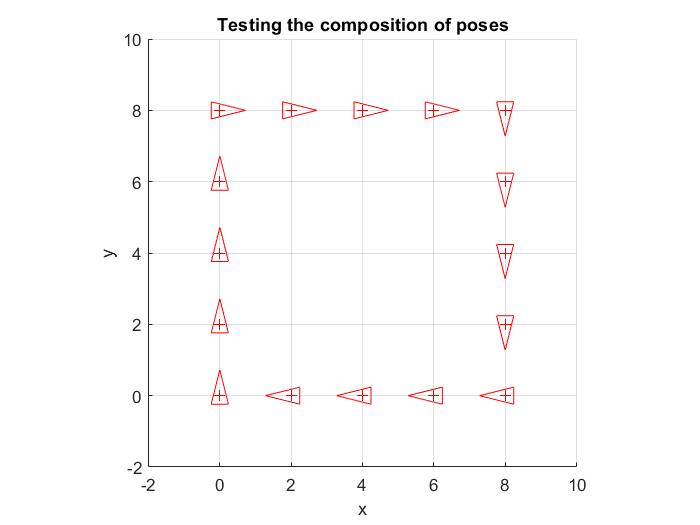


Figure 1

This figure represents the ideal path which will do a robot if we do not have anything that interferes to the robot.

1. In the previous case, the robot motion is error-free. Now, add a Gaussian noise to the motion. Holding the previous plot, draw a random motion of the robot, assuming the incremental motion is:

with (units in m2 and rad2)

Run the program several times to see that the motion (and the path) is different each time. Try also with different values of the covariance matrix.

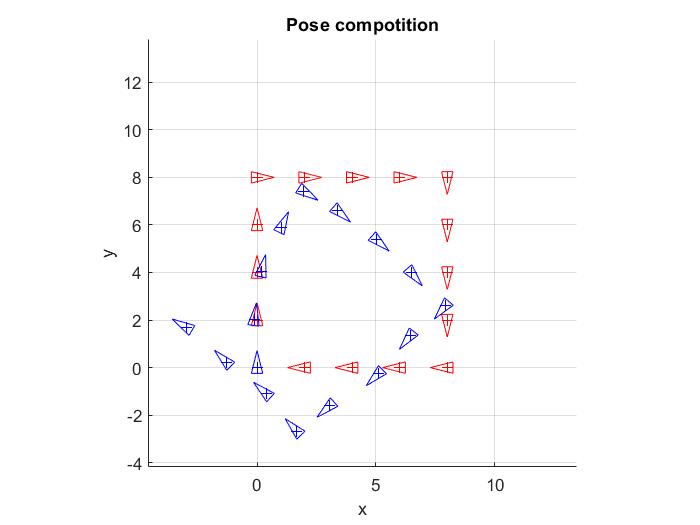
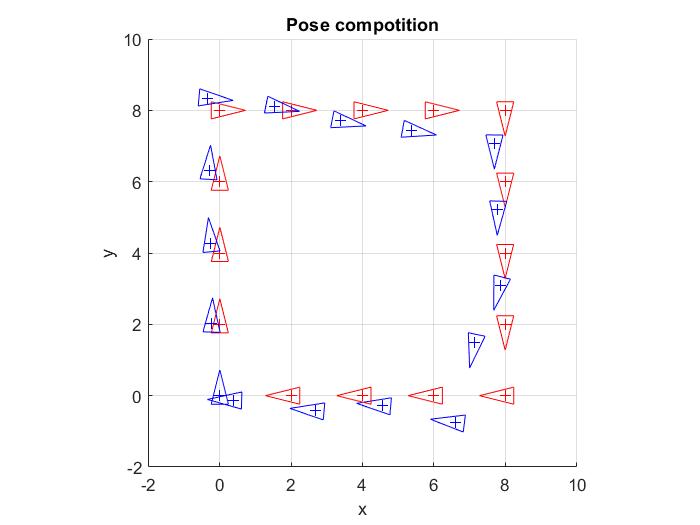


Figure 2

Figure 3

As we see in the both figures, if we put a noise to the ideal path, it will give a simulation of a real robot. It can be happened for a lot of reason, from having irregularities the floor to the robot have different size of its wheels.

% Testing the composition of robot poses

close all;

clear

nSteps = 15; %Number of motions

t = [2;0]; %translation,

ang = -90\*pi/180; %orientation angle

pose\_inc\_straight\_line = [t;0]; %pose increment

pose\_inc\_straight\_line\_and\_rotation = [t; ang]; %pose increment

pose = [0;0; pi/2]; %initial pose of the robot (at k = 0)

figure (1); hold on; axis equal; grid on; axis ([-2 10 -2 10])

xlabel('x'); ylabel('y'); title ('Testing the composition of poses');

DrawRobot(pose,'r'); % Draw initial pose

pause

for k = 1: nSteps %Main loop

if mod(k,4) == 0

pose = tcomp (pose, pose\_inc\_straigh\_line\_and\_rotation);

else

pose = tcomp (pose, pose\_inc\_straigh\_line);

end

DrawRobot(pose,'r');

pause

end;

function tac=tcomp (tab, tbc)

%Composition of transformations tab and tbc given by poses (i.e. vectors)

if size(tab,1) ~= 3, error ('TCOMP: tab not a transformation!'); end;

if size(tbc,1) ~= 3, error ('TCOMP: tbc not a transformation!'); end;

ang = tab (3) +tbc (3);

if ang > pi | ang <= -pi

ang = AngleWrap(ang);

end

s = sin (tab (3)); c = cos (tab (3));

tac = [tab (1:2) + [c -s; s c] \*tbc (1:2); ang];