b) Define uncertainty of the platform by “sampling” algorithms platform Platform model: Deadreckoning.

A GUI has been designed for deploying both Deadreckoning and Odometry models. It can be seen in Figure 1.

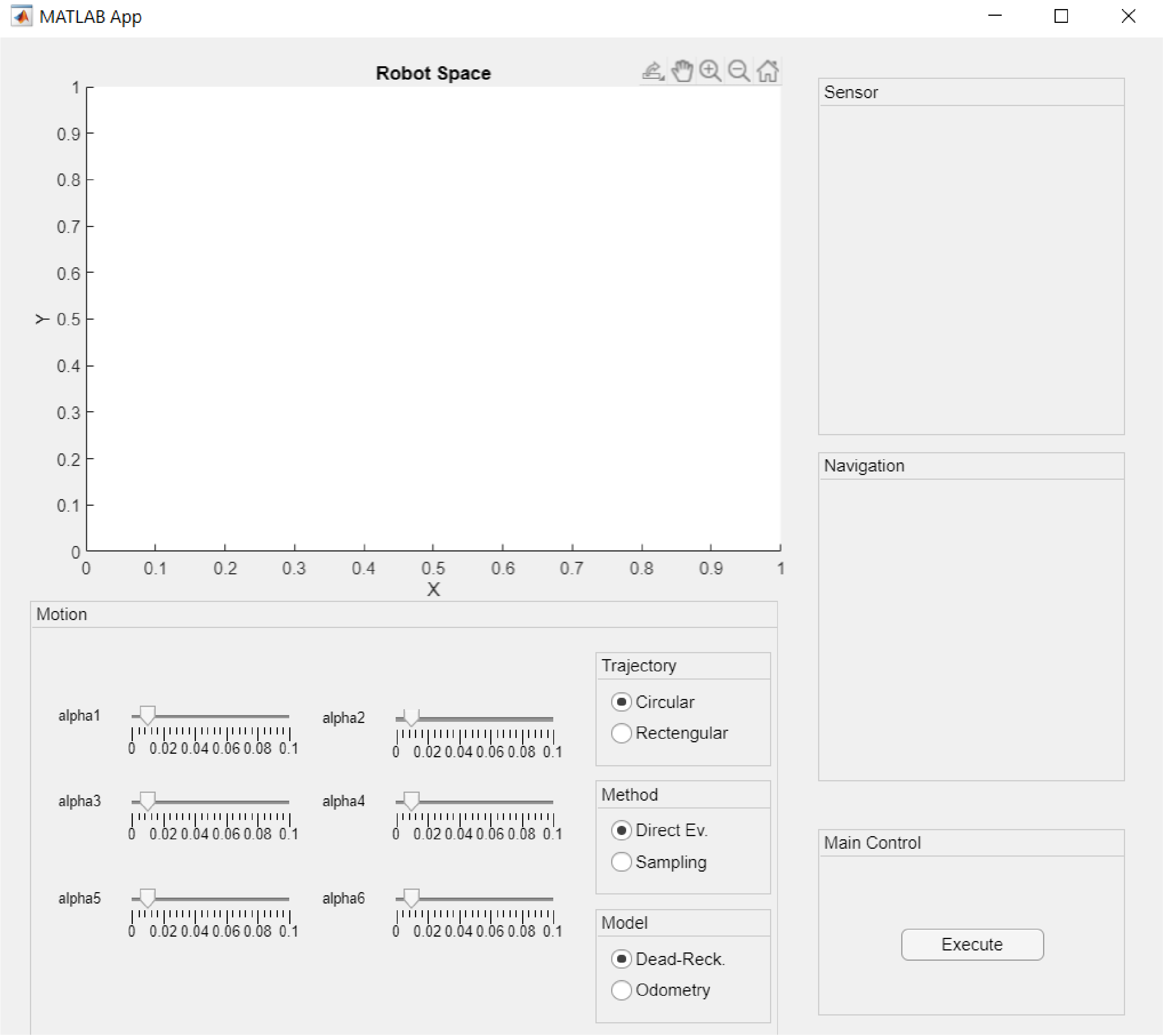

Figure 1 GUI

Common “sample” function for generating gaussian random variables are as below.

```
function sample = normal_dist(app,a1, a2, v,w)
    b = (a1 * abs(v) + a2 * abs(w));
    ran_num  = -b + (b+b).*rand(12,1);
    sum = 0;
    for i=1:12
        sum = sum + ran_num(i);
    end
    sample = 0.5 * sum;
end
```

The MATLAB code of sampling algorithm for Deadreckoning Circular Trajectory model has been given below.

```
    if(app.CircularButton.Value == 1)

        sample_n = 150;
        x=zeros(sample_n,8); y=zeros(sample_n,8); tet = zeros(sample_n,8);
        x(:,1) = 1; tet(:,1) = pi/2;
        r = 1;
        t = 1; %time of traviling for 45 degrees
        w = -pi/4; %Assumed 45 degrees travel in one sec.
        v = w * r;
        plot(app.UIAxes,x(1,1),y(1,1),'k.','LineWidth',2);
        hold(app.UIAxes,"on");
          for i=1:7
                for j=1:sample_n
                    v_hat = v + normal_dist(app,a1, a2, v,w);
                    w_hat = w + normal_dist(app,a3, a4, v,w);
                    gama_hat = normal_dist(app,a5, a6, v,w);
                    x(j,i+1) = x(j,i) -  v_hat/w_hat * sin(tet(j,i)) +  v_hat/w_hat * sin(tet(j,i)
+ w_hat*t);
                    y(j,i+1) = y(j,i) +  v_hat/w_hat * cos(tet(j,i)) -  v_hat/w_hat * cos(tet(j,i)
+ w_hat*t);
                    tet(j,i+1) = tet(j,i) + w_hat*t + gama_hat*t;
                    plot(app.UIAxes,x(j,i+1),y(j,i+1),'k.','LineWidth',2);
                    hold(app.UIAxes,"on");
                end
           end
            hold(app.UIAxes,'off');
    end
    if(app.RectengularButton.Value == 1)
    sample_n = 100;
    x=zeros(sample_n,13); y=zeros(sample_n,13); tet = zeros(sample_n,13);
    x(:,1) = -4; y(:,1) = -4; tet(:,1) = 0; %Initial position.
    t = 1; %time of traviling for 2 meters
    v = 2; %Assumed traveled 2 meters in 1 sec.
    w = 0.001; %Assumed no rotation.
    plot(app.UIAxes,x(1,1),y(1,1),'k.','LineWidth',2);
    hold(app.UIAxes,"on");
        for i=1:12
            if(i == 4 || i == 8)
                w = pi/2;
            else
                w = 0.001;
            end
            for j=1:sample_n
                v_hat = v + normal_dist(app,a1, a2, v,w);
                w_hat = w + normal_dist(app,a3, a4, v,w);
                gama_hat = normal_dist(app,a5, a6, v,w);
                x(j,i+1) = x(j,i) -  v_hat/w_hat * sin(tet(j,i)) +  v_hat/w_hat * sin(tet(j,i) +
w_hat*t);
                y(j,i+1) = y(j,i) +  v_hat/w_hat * cos(tet(j,i)) -  v_hat/w_hat * cos(tet(j,i) +
w_hat*t);
                tet(j,i+1) = tet(j,i) + w_hat*t + gama_hat*t;
                plot(app.UIAxes,x(j,i+1),y(j,i+1),'k.','LineWidth',2);
                hold(app.UIAxes,'on');
            end
        end
        hold(app.UIAxes,'off');
```

```
    end
    end
```

The results for this algorithm for value of 0.01 for alpha1 to alpha6 have been given in Figure 2.

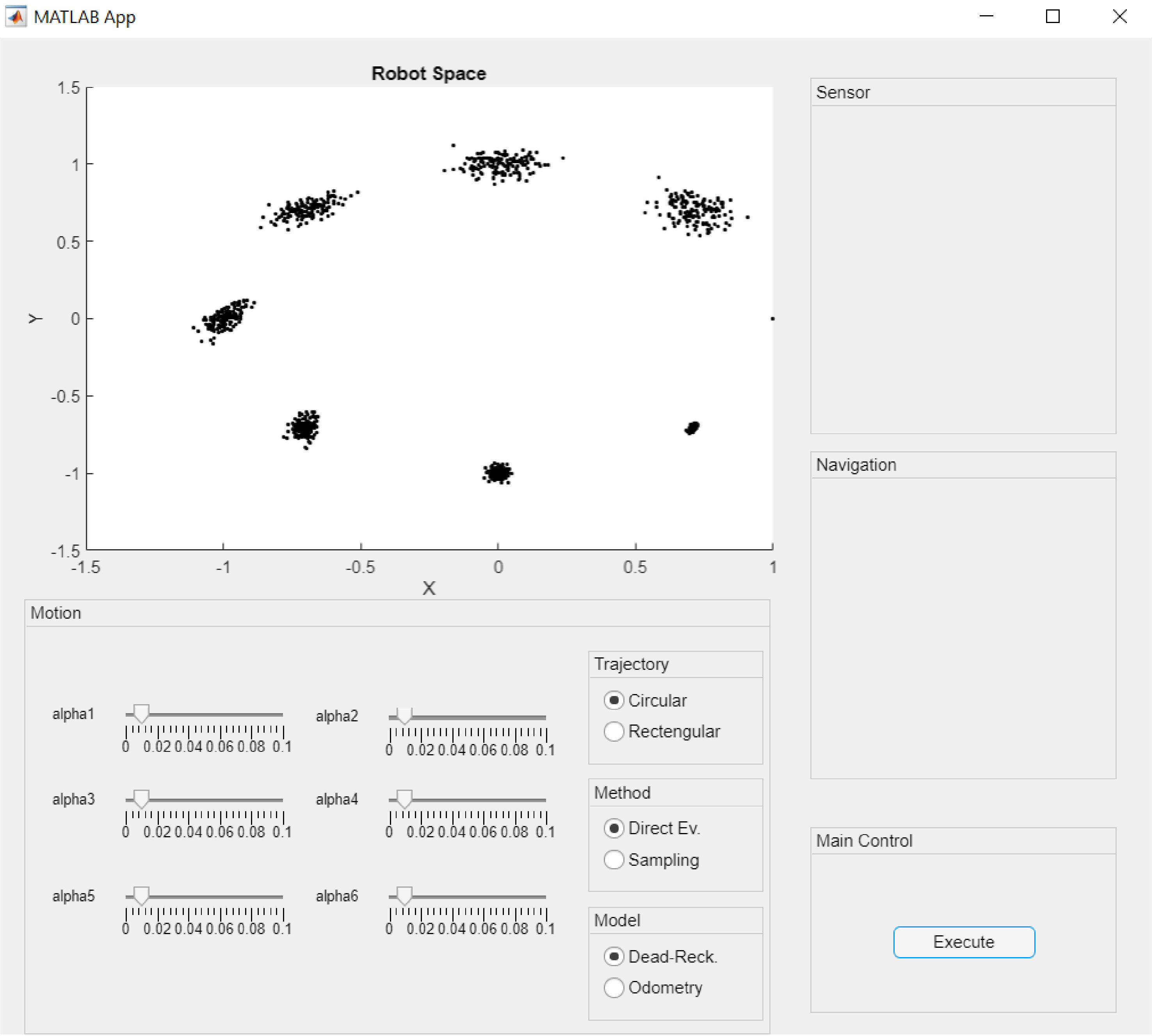

Figure 2 Deadreckoning model with sampling algorithm for alphaN = 0.01

Increasing alpha values causes more uncertainty in the model. This can be seen in Figure 3.

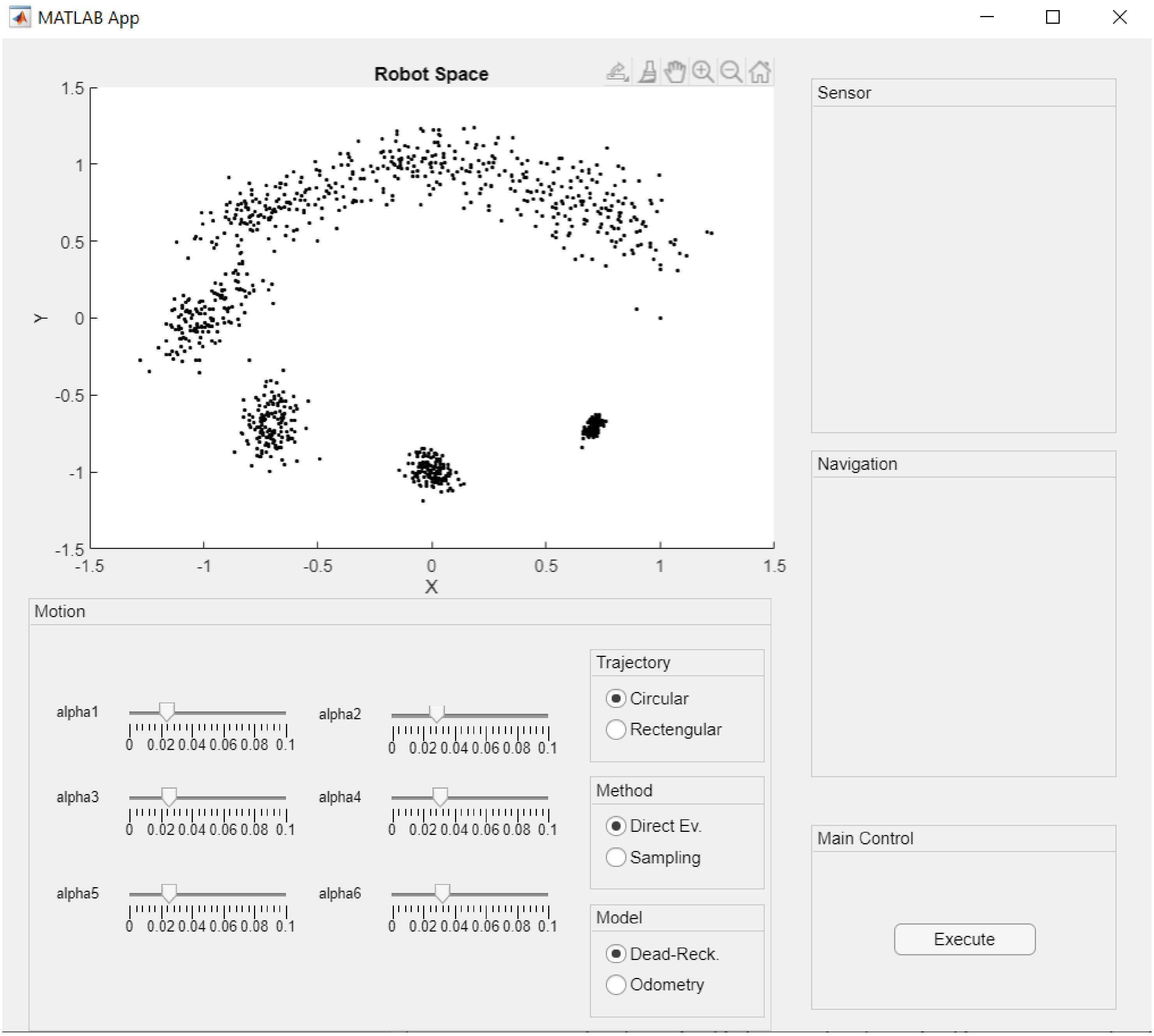

Figure 3 Deadreckoning model with sampling algorithm for alphaN > 0.01

c) Do the same for (a) & (b) using “odometry model”.

The MATLAB code of sampling algorithm for Odometry Circular Trajectory model has been given below.

```
function Odometry(app)
    a1 = app.alpha1Slider.Value;  a2 = app.alpha2Slider.Value;
    a3 = app.alpha3Slider.Value;  a4 = app.alpha4Slider.Value;
    a5 = app.alpha5Slider.Value;  a6 = app.alpha6Slider.Value;
    if(app.CircularButton.Value == 1)
        sample_n = 150;
        xc = 0; yc = 0;
        x=zeros(sample_n,8); y=zeros(sample_n,8); tet = zeros(sample_n,8);
        xp=zeros(sample_n,8); yp=zeros(sample_n,8); tetp = zeros(sample_n,8);
        xp(:,1) = 1; yp(:,1) = 0; tetp(:,1) = -pi/2;
        tet(:,1) = -pi/2;
        r = 1;
        t = 1; %time of traviling for 45 degrees
```

```
        w = -pi/4;
        for i=1:8
              x(:,i) = xc - r*sin(tet(1,i));
              y(:,i) = yc + r*cos(tet(1,i));
              tet(1,i+1) = tet(1,i) + w*t;
        end
            for i=1:7
                for j=1:sample_n
                    delta_rot1 = atan2((y(j,i+1)-y(j,i)),(x(j,i+1)-x(j,i))) - tet(j,i);
                    delta_trans = sqrt((y(j,i+1)-y(j,i))^2 + (x(j,i+1)-x(j,i))^2);
                    delta_rot2 = tet(j,i+1) - tet(j,i) - delta_rot1;

                    delta_rot1_hat = delta_rot1 -
normal_dist(app,a1,a2,delta_rot1,delta_trans);
                    delta_trans_hat = delta_trans -
normal_dist(app,a3,a4,delta_trans,delta_rot1+delta_rot2);
                    delta_rot2_hat = delta_rot2 -
normal_dist(app,a1,a2,delta_rot2,delta_trans);

                    xp(j,i+1) = xp(j,i) + delta_trans_hat * cos(tetp(j,i)+delta_rot1_hat);
                    yp(j,i+1) = yp(j,i) + delta_trans_hat * sin(tetp(j,i)+delta_rot1_hat);
                    tetp(j,i+1) = tetp(j,i) + delta_rot1_hat + delta_rot2_hat;
                end
            end
        plot(app.UIAxes,xp,yp,'k.');
    end
```

The results for this algorithm for value of 0.01 for alpha1 to alpha6 and values greater then 0.001 have been given in Figure 4.

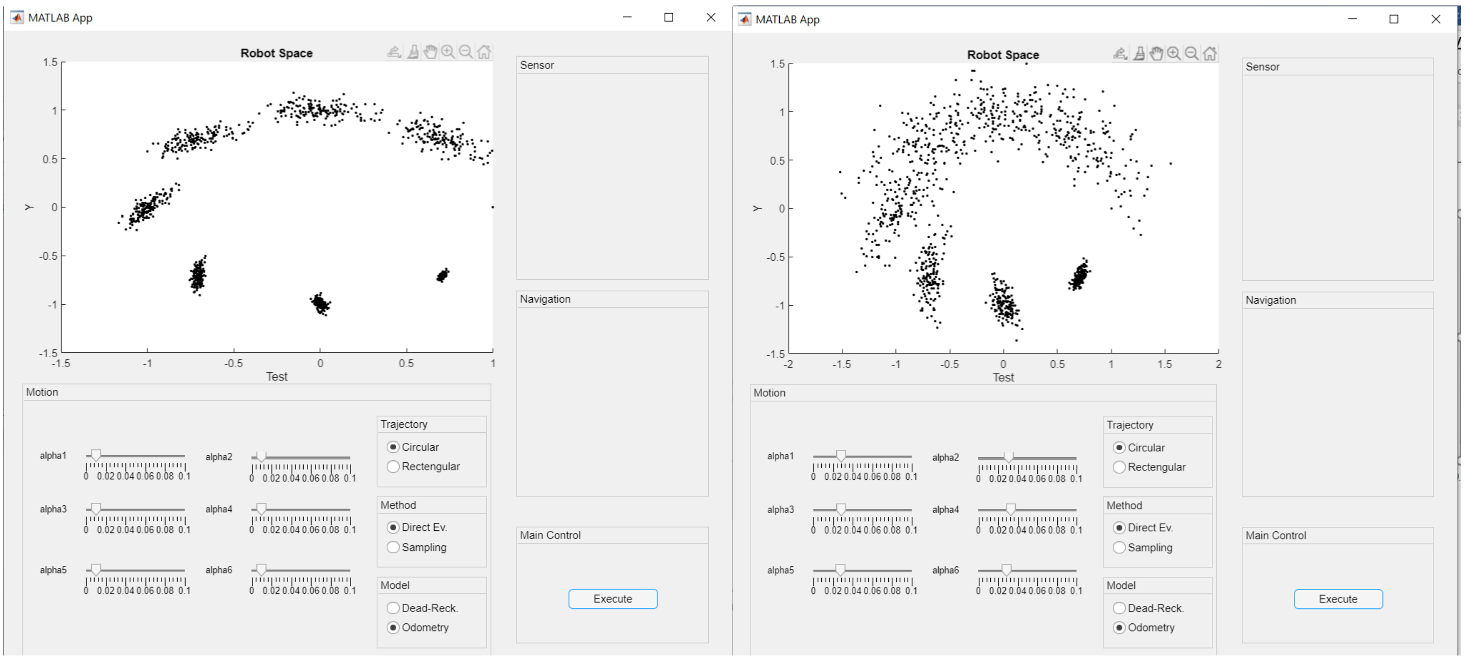

Figure 4 Odometry model with sampling algorithm for alphaN = 0.01 and alphaN > 0.01

II) Motion model for a Rectangular trajectory

a) Use odometry model and plot the uncertainty using sampling method.

MATLAB code for odometry model and rectangular trajectory is as below.

```
if(app.RectengularButton.Value == 1)
        sample_n = 150;
        xc = 0; yc = 0;
        x=zeros(sample_n,13); y=zeros(sample_n,13); tet = zeros(sample_n,13);
        xp=zeros(sample_n,13); yp=zeros(sample_n,13); tetp = zeros(sample_n,13);
        x(:,1) = -4; y(:,1) = -4; tet(:,1) = 0;
        xp(:,1) = -4; yp(:,1) = -4; tetp(:,1) = 0;
        tet(:,1) = 0;
        r = 1;
        t = 1; %time of traviling for 45 degrees
        d2r = pi/180;
        w = 0.001; %Assumed 45 degrees travel in one sec.
        v = 2;
        for i=2:5
              x(:,i) = x(:,i-1) + 2;
              y(:,i) = y(1,1);
              tet(:,i) = 0;
        end

        for i=6:9
              x(:,i) = x(1,5);
              y(:,i) = y(:,i-1) + 2;
              tet(:,i) = pi/2;
        end

        for i=10:13
              x(:,i) = x(:,i-1) - 2;
              y(:,i) = 4;
              tet(:,i) = pi;
        end

            for i=1:12
                for j=1:sample_n
                    delta_rot1 = atan2((y(j,i+1)-y(j,i)),(x(j,i+1)-x(j,i))) - tet(j,i);
                    delta_trans = sqrt((y(j,i+1)-y(j,i))^2 + (x(j,i+1)-x(j,i))^2);
                    delta_rot2 = tet(j,i+1) - tet(j,i) - delta_rot1;

                    delta_rot1_hat = delta_rot1 -
normal_dist(app,a1,a2,delta_rot1,delta_trans);
                    delta_trans_hat = delta_trans -
normal_dist(app,a3,a4,delta_trans,delta_rot1+delta_rot2);
                    delta_rot2_hat = delta_rot2 -
normal_dist(app,a1,a2,delta_rot2,delta_trans);

                    xp(j,i+1) = xp(j,i) + delta_trans_hat * cos(tetp(j,i)+delta_rot1_hat);
                    yp(j,i+1) = yp(j,i) + delta_trans_hat * sin(tetp(j,i)+delta_rot1_hat);
                    tetp(j,i+1) = tetp(j,i) + delta_rot1_hat + delta_rot2_hat;
                end
            end
        plot(app.UIAxes,xp,yp,'k.');
    end
    end
```

Resuls can be seen in Figure 5 for both alphaN = 0.01 and alphaN > 0.01.

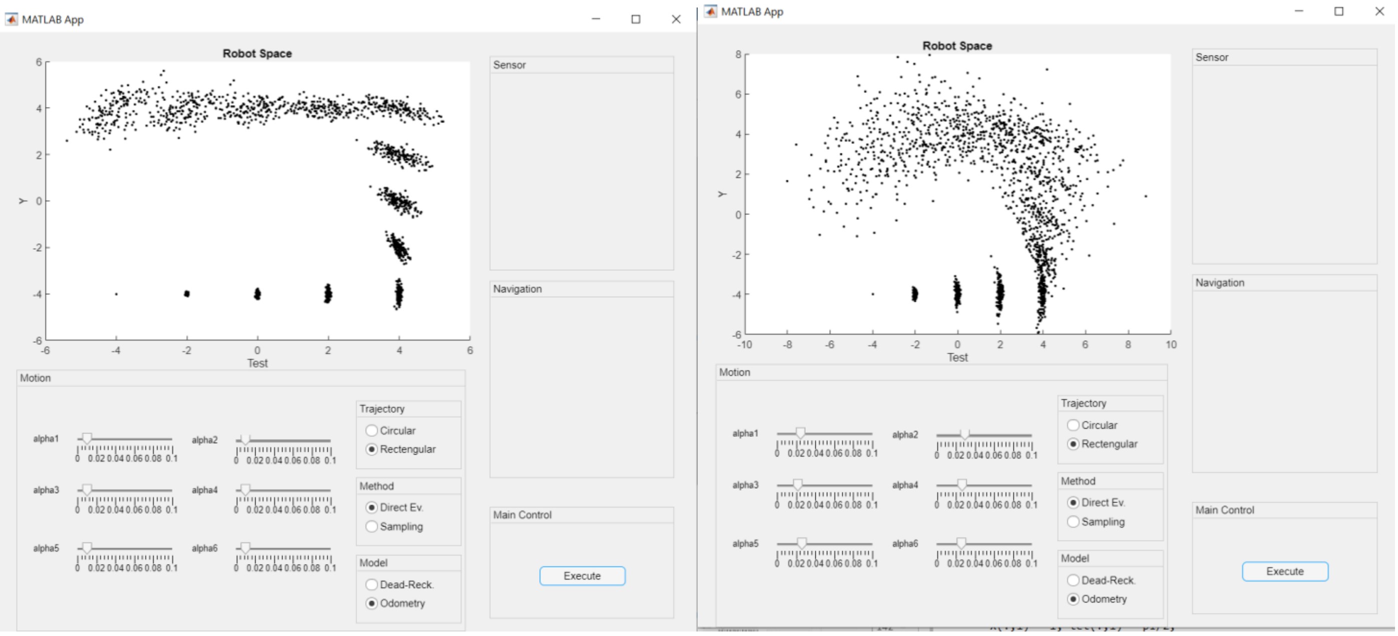

Figure 5 Odometry Rectenguler Trajectory with Sampling Algorithm

The results for Deadreckoning algorithm is in Figure 6.

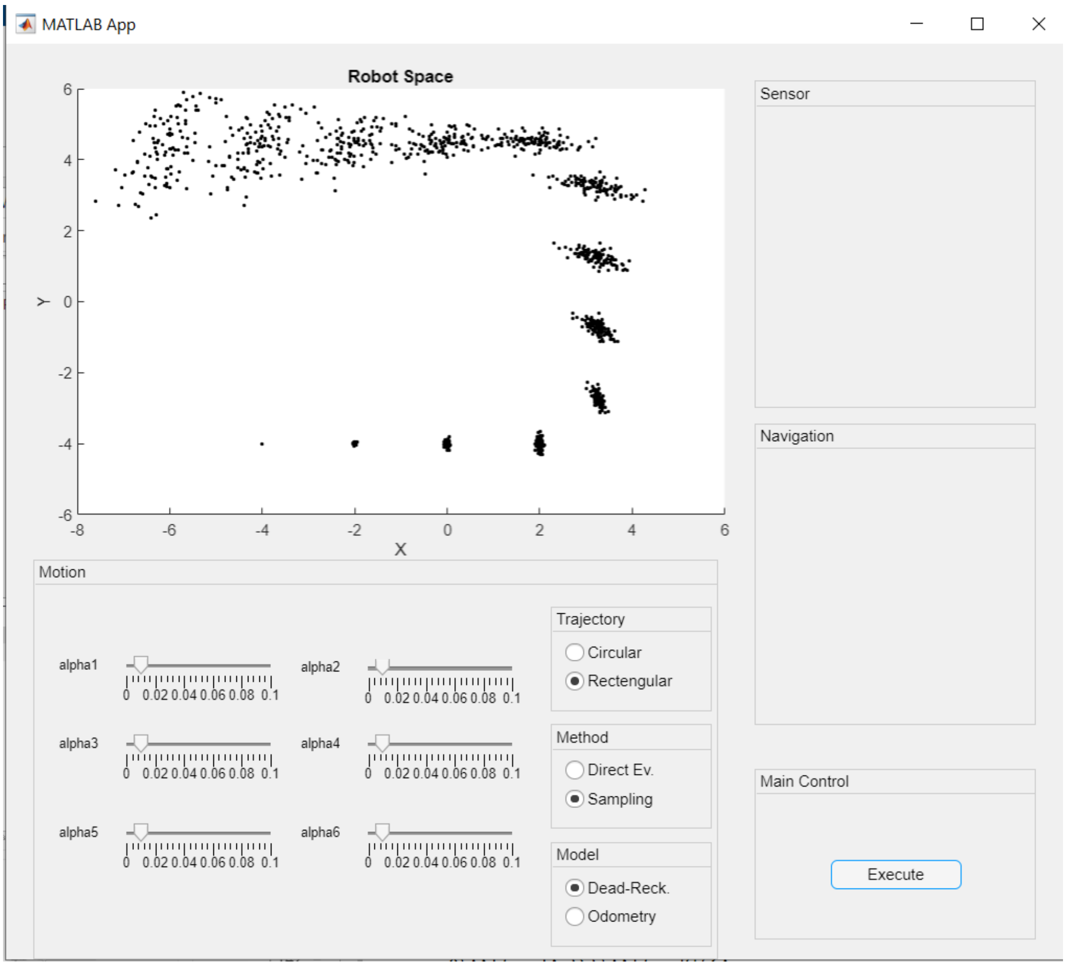

Figure 6 Deadreckoning Rectenguler Trajectory with Sampling Algorithm