In [ ]:
```
def pathsearch(s_pt,g_pt):

    t_list = [[2,15],[3,16],[3,17],[4,18],[4,19],[10,14],[11,13],[12,12],[13,11],[14,10],[5,4],[9,15],
              [6,4],[7,4],[8,4],[9,4],[5,5],[6,5],[7,5],[8,5],[9,5],[4,4],[4,5],[10,4],[10,5],[15,9]]

    #way_pt = [[17,15],[6,14],[3,4],[17,3],[9,9]]
    route = []

    #s_pt = [1,2]
    s_gt_pt = mapper.from_map(s_pt[0],s_pt[1],0)
    #loc.plotter.plot_point(s_gt_pt[0], s_gt_pt[1],ODOM)
    #g_pt = [17,17]
    g_gt_pt = mapper.from_map(g_pt[0],g_pt[1],0)
    #loc.plotter.plot_point(g_gt_pt[0], g_gt_pt[1],GT)
    r_flag = 0
    if((20- s_pt[0]) > (20 - g_pt[0]) or s_pt[1] > g_pt[1]):
        buff = g_pt
        g_pt = s_pt
        s_pt = buff
        r_flag = 1



    way_pt = [[17,8]] #sim_way_pt
    #way_pt = [[10,10]]
    way_pt.append(g_pt)
    route.append(s_pt)
    s_gt_pt = mapper.from_map(s_pt[0],s_pt[1],0)
    loc.plotter.plot_point(s_gt_pt[0], s_gt_pt[1],ODOM)
    for i in range (len(way_pt)):
        g_pt = way_pt[i]
        dis_x_f = g_pt[0]-s_pt[0]
        dis_y_f = g_pt[1]-s_pt[1]

        con = 0
        flag_x = 0
        flag_y = 0
        tmp_x = s_pt[0]
        tmp_y = s_pt[1]
        y_en = 0
        y_en2 = 0

        while(con == 0):
            while(flag_x < abs(dis_x_f)):
                dis_x = g_pt[0] - tmp_x
                dis_y = g_pt[1] - tmp_y
                if(y_en == 0):
                    if(dis_x <0):
                        tmp_x = tmp_x - 1
                        tmp_pt = [tmp_x, tmp_y]
                    else:
                        tmp_x = tmp_x + 1
                        tmp_pt = [tmp_x, tmp_y]
                else:
```

```
            tmp_pt = [tmp_x, tmp_y]
        flag = 0
        for i in range (len(t_list)):
            if(np.any(tmp_pt == t_list[i])):
                flag = flag + 1
            elif(tmp_pt[0] >= 21 or tmp_pt[0] <= -1):
                flag = flag + 1
            elif(tmp_pt[1] >= 21 or tmp_pt[1] <= -1):
                flag = flag + 1
            else:
                flag = flag + 0
        if(flag > 0 and y_en == 0): #go up
            if(dis_x <0):
                tmp_x = tmp_x + 1
            else:
                tmp_x = tmp_x - 1
            tmp_y = tmp_y + 1
            y_en = 1
        elif(flag > 0 and y_en == 1): #go down
            tmp_y = tmp_y - 2
            y_en2 = 1
        elif(flag ==0):
            route.append(tmp_pt)
            tmp_gt_pt = mapper.from_map(tmp_pt[0],tmp_pt[1],0)
            loc.plotter.plot_point(tmp_gt_pt[0], tmp_gt_pt[1],ODOM)
            if(y_en == 1 and y_en2 == 0):
                if(dis_y < 0):
                    flag_y = flag_y - 1
                else:
                    flag_y = flag_y + 1
            elif(y_en == 1 and y_en2 == 1):
                if(dis_y < 0):
                    flag_y = flag_y + 1
                else:
                    flag_y = flag_y - 1
            else:
                flag_x = flag_x + 1
            y_en = 0
            y_en2 = 0
    x_en = 0
    x_en2 = 0

    while(flag_y < abs(dis_y_f)):
        dis_x = g_pt[0] - tmp_x
        dis_y = g_pt[1] - tmp_y
        if(x_en ==0):
            if(dis_y <0):
                tmp_y = tmp_y - 1
                tmp_pt = [tmp_x, tmp_y]
            else:
                tmp_y = tmp_y + 1
                tmp_pt = [tmp_x, tmp_y]
        else:
            tmp_pt = [tmp_x, tmp_y]
        flag = 0
        for i in range (len(t_list)):
            if(np.any(tmp_pt == t_list[i])):
```

```
                    flag = flag + 1
                elif(tmp_pt[0] >= 21 or tmp_pt[0] <= -1):
                    flag = flag + 1
                elif(tmp_pt[1] >= 21 or tmp_pt[1] <= -1):
                    flag = flag + 1
                else:
                    flag = flag + 0
            if(flag > 0 and x_en == 0): #go right
                if(dis_y <0):
                    tmp_y = tmp_y + 1
                else:
                    tmp_y = tmp_y - 1
                tmp_x = tmp_x + 1
                print(tmp_x, tmp_y)
                x_en = 1
            elif(flag > 0 and x_en == 1): #go left
                tmp_x = tmp_x - 2
                print(tmp_x, tmp_y)
                x_en2 = 1
            elif(flag == 0):
                route.append(tmp_pt)
                tmp_gt_pt = mapper.from_map(tmp_pt[0],tmp_pt[1],0)
                loc.plotter.plot_point(tmp_gt_pt[0], tmp_gt_pt[1],ODOM)
                if(x_en == 1 and x_en2 == 0):
                    if(dis_x < 0):
                        flag_x = flag_x - 1
                    else:
                        flag_x = flag_x + 1
                elif(x_en == 1 and x_en2 == 1):
                    if(dis_y < 0):
                        flag_x = flag_x + 1
                    else:
                        flag_x = flag_x - 1
                else:
                    flag_y = flag_y + 1
                x_en = 0
                x_en2 = 0
        if(tmp_x == g_pt[0] and tmp_y == g_pt[1]):
            con = 1
    s_pt = g_pt

if(r_flag == 1):
    route = route[::-1]
return route
```

In [ ]:
```
def robotcontrol(route):
    #calculate the approximate time for velocity
    dist = []
    for i in range (len(route)-1):
        if(route[i+1][1] == route[i][1]): #x change
            if(route[i+1][0] == route[i][0] + 1):
                dist.append(1)
            else:
                dist.append(-1)
        else: #y change
            if(route[i+1][1] == route[i][1] + 1):
                dist.append(2)
            else:
                dist.append(-2)
    v_control = []
    t_control = []
    v_control.append(dist[0])
    t = 0
    for i in range (len(dist)):
        if(i > 0 and dist[i] != dist[i-1]):
            v_control.append(dist[i+1])
            t_control.append(t)
            t = 0
        t = t + 1
    t_control.append(t)

    bond_x = 4
    bond_y = 4
    spa_x = bond_x / 20
    spa_y = bond_y / 20
    for i in range (len(v_control)):
        if(v_control[i] == 1 or v_control[i] == -1):
            t_control[i] = t_control[i]*spa_x
        elif(v_control[i] == 2 or v_control[i] == -2):
            t_control[i] = t_control[i]*spa_y
    return v_control, t_control
```

In [ ]:
```
# Init Uniform Belief
loc.init_pose()

# Get Observation Data by executing a 360 degree rotation motion
loc.get_observation_data()

# Run Update Step
loc.update_step()
pose=loc.print_update_stats(plot_data=True) #get the start point

#assign start point and goal point
s_pt = [pose[0],pose[1]]
g_pt = [9,7]
#path planning
route = pathsearch(s_pt,g_pt)
[v_control,t_control]=robotcontrol(route)

#speed control
w = 0.2
v = 0.25

#plot inital point
a_pt = route[0]
a_gt_pt = mapper.from_map(a_pt[0],a_pt[1],0)
loc.plotter.plot_point(a_gt_pt[0], a_gt_pt[1],GT)

#start process
prev_odom = robot.get_gt_pose()
i_angle = pose[2] #inital angle

for i in range (len(v_control)):
    ang_flag = 0
    while(ang_flag == 0):
        if(v_control[i] == 1):
            angle = 9 #index 9 -- 0 degree
        elif(v_control[i] == -1):
            angle = 0  #index 0 ---180 degree
        elif(v_control[i] == 2):
            angle = 13 #index 13 --90 degree
        elif(v_control[i] == -2):
            angle = 4 #index 4 -- -90 degree

        diff_ang = i_angle - angle
        if(abs(diff_ang) >= 2):
            if(diff_ang < 0):
                diff_ang = abs(diff_ang)
                #w = abs(w)
            else:
                diff_ang = 18 - diff_ang
                #w = -abs(w)
            w_t = (diff_ang-0.25)*20/180*3.1415/w
            robot.set_vel(0,w)
            time.sleep(abs(w_t))
            robot.set_vel(0,0)
            i_angle = angle
        #check the pose angle
```

```
        # Prediction Step
        current_odom = robot.get_pose()
        loc.prediction_step(current_odom, prev_odom)
        loc.print_prediction_stats(plot_data=True)
        # Get Observation Data by executing a 360 degree rotation motion
        loc.get_observation_data()
        # Update Step
        loc.update_step()
        pose = loc.print_update_stats(plot_data=True)
        prev_odom = current_odom
        check_ang = pose[2] - angle
        if(abs(check_ang) <= 4):
            ang_flag = 1
        else:
            i_angle = pose[2]



    #i_angle = pose[2]
    v_t = t_control[i]/v
    robot.set_vel(0,0)
    time.sleep(0.1)
    robot.set_vel(v,0)
    time.sleep(v_t)
    robot.set_vel(0,0)
    time.sleep(0.1)


#identify whether the robot reach the last point
# Prediction Step
loc.prediction_step(current_odom, prev_odom)
loc.print_prediction_stats(plot_data=True)
# Get Observation Data by executing a 360 degree rotation motion
loc.get_observation_data()
# Update Step
loc.update_step()
pose = loc.print_update_stats(plot_data=True)

close_enough = 0.5

end_pt = route[-1:][0]
real_loc = mapper.from_map(end_pt[0],end_pt[0],pose[2])
exp_loc = mapper.from_map(pose[0],pose[1],pose[2])
dis_err = (real_loc[0] - exp_loc[0]) ** 2 + (real_loc[1] - exp_loc[1]) ** 2

if(dis_err <= (close_enough **2)):
    print("reach the point")
```