Applied Control Assignment 2

Yi Zhou 9971603
Group number:26 Teammate: Weilin Zhong

Task 1

The aim of the assignment is to implement sliding mode control for trajectory tracking with DaNI nonholonomic mobile robot using LabVIEW. The cascade control structure with PID inner loop and SMC outer loop is used for this system.

The cascade control is always used for the system which can be divided into two parts, a fast system as inner loop and a slow system as a outer loop. For this robot system, the sliding mode control system takes account of the trajectory and provide a sliding surface where the error in state may converge to zero. Thus, it is used as outer loop, which provided the designed angular velocity as the reference for the PID controller. The PID controller should be used as inner loop, which enables the tracking of the angular velocity designed by SMC. The PID controller should be faster than the SMC, in other words, its sampling time should be smaller than that of SMC. Otherwise, if the PID is too slow, its control effect will be weak. Then the previous designed velocity may not be reached in the next loop of SMC, which means the SMC will not take the effect in time. In this assignment, the sampling time of the SMC loop is fixed as 150 ms due to the interval time in the trajectory files. Then the sampling time of PID is designed as 65 ms.

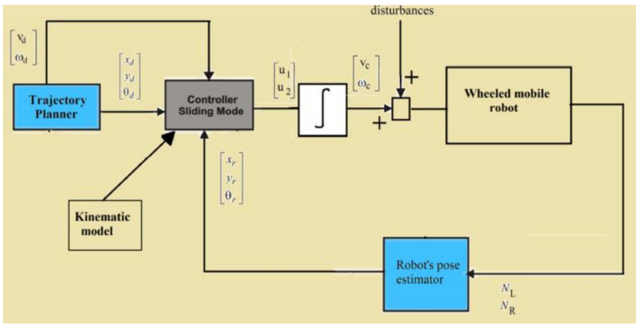

disturbances
$\begin{bmatrix} v_d \\ \omega_d \end{bmatrix}$
$\begin{bmatrix} x_d \\ y_d \\ \theta_d \end{bmatrix}$
Trajectory Planner
Controller Sliding Mode
$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$
$\begin{bmatrix} v_c \\ \omega_c \end{bmatrix}$
+
+
Wheeled mobile robot
$\begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}$
Kinematic model
Robot's pose estimator
N_L
N_R

Figure 1.1 The structure of controller[1]

In the outer loop there are mainly four parts. The first part is robot's pose estimator. It read the past data from shift registers (for pose) and local variables(for angular velocity), then update the pose and velocity information as the input of the SMC controller.

$$v_r = \frac{r}{2}\omega_R + \frac{r}{2}\omega_L$$

$$w_r = \frac{r}{l}\omega_R - \frac{r}{l}\omega_L$$

$$x := x + dt\left(\frac{\omega_r + \omega_l}{2} r\right)\cos(\theta)$$

$$y := y + dt\left(\frac{\omega_r + \omega_l}{2} r\right)\sin(\theta)$$

$$\theta := \theta + dt\left(\frac{\omega_r - \omega_l}{l} r\right)$$

The second part is trajectory planner. Since the trajectory is already made, only we should do is to extract it from the text files. A 'scan from file' function is used to read the trajectory from the text files. The trajectory have eight columns which represents desired pose x,y,theta, desired velocity v,w, desired acceleration vdot, wdot and time. Then the necessary information are connected to the required block.

After getting the desired trajectory and current pose information, the error calculation block can be executed.

The errors in pose are

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x - x_d \\ y - y_d \\ \theta - \theta_d \end{bmatrix}$$

The errors in velocity are

$$\dot{x}_e = -v_d + v \cdot \cos\theta_e + y_e w_d$$
$$\dot{y}_e = v \cdot \sin\theta_e - x_e w_d$$
$$\dot{\theta}_e = \omega - \omega_d$$

Finally, after getting all necessary information, the SMC can be ran to regulate the desired velocity. The sliding law is designed as

$$s_1 = \dot{x}_e + k_1 x_e$$
$$s_2 = \dot{\theta}_e + k_2\theta_e + k_0 y_e$$

Then s1=s2=0 is the sliding surface. All of the trajectories should finally converge to this surface in order to make the errors asymptotically converge to zero . Moreover, the sliding mode is robust to the modeling errors and disturbances.

When s_1 is zero, the x_e converges to zero with a time constant determined by k_1. When s_2 is zero, the convergence of y and θ are coupled. However, it can be easily proved that the y and θ will both converges to zero. To tune these parameters, the larger k_1 the faster x_e converges. The larger k_2 the faster θ_e converges. The larger k_0 will slow the convergence of y_e and θ_e. But it acts like a weighting number of y_e which can reduced the error in y. After several test, we set k_0, k_1, k_2 as 2,4,1.1 respectively.

The reaching law is designed as

$$\dot{s}_1 = -Q_1 s_1 - P_1 \operatorname{sgn}(s_1)$$
$$\dot{s}_2 = -Q_2 s_2 - P_2 gn(s_2)$$

The larger the Q is, the faster s converges to zero. The larger the P is, the lager oscillation, which is called chattering, will occur around the sliding surface. In the meanwhile, too small value of these two parameter will both lead to slow converging. After some test, the final value for our controller are set as P_1=P_2=0.2 ,Q_1=Q_2=0.4.

In theory, when the trajectory reached the sliding surface, it will follow the surface, i.e. s and sdot should all equal to zero. However, in reality, the s dot can not be constant zero and s should be bounded in a slight deviation of the zero surface. The reason is that the discrete controller may constantly change the direction of the control signal along the sliding surface , which called chattering. The phenomenon of chattering may lead to the undesired high frequency dynamics and noise of the system. In our SMC method ,the asymptotic sliding mode is used to reduce the chattering. The SMC is implemented on the control function derivative,i.e. vdot and wdot. Thus the actual controller is continuous through integral.

One simple way to eliminate chattering is to make a threshold below which the SMC will not work. Furthermore, the continuous sigmoid function can be used instead of the discrete sign function. The above methods can be summarised as boundary layer approach which leads to Quasi sliding mode controller. However this method will sacrifice the robustness of the controller.

Finally the velocity are computed and updated through its derivative.

$$\dot{v}_c = \frac{1}{\cos(\theta_e)}\left[-Q_1 s_1 - P_1 \operatorname{sgn}(s_1) - k_1 \dot{x}_e + \dot{V}_d \right.$$
$$\left. + v \cdot \sin(\theta_e) \cdot \dot{\theta}_e - \dot{y}_e \omega_d - \dot{\omega}_d y_e \right]$$
$$\dot{\omega}_c = -Q_2 s_2 - P_2 \operatorname{sgn}(s_2) - k_0 \dot{y}_e + \dot{\omega}_d - k_2 \dot{\theta}_e$$
$$v_c = v_c + \dot{v}_c \cdot dt$$
$$\omega_c = \omega_c + \dot{\omega}_c \cdot dt$$

After that, the designed angular velocities of two wheels then will be computed and transferred into the PID controller by using two local variables.

$$\omega_R = \frac{l}{2r}\left(\omega_c + \frac{2}{l} v_c\right)$$
$$\omega_L = \frac{l}{2r}\left(-\omega_c + \frac{2}{l} v_c\right)$$

The PID loop is designed just like assignment 1. It takes the form of digital PID

$$u_n = K_p\left[e_n + \frac{1}{T_i}\left(I_{n-1} + e_n T\right) + T_d\left(\frac{e_n - e_{n-1}}{T}\right)\right]$$
$$= K_p\left[e_n\left(1 + \frac{T_d}{T} + \frac{T}{T_i}\right) - e_{n-1}\left(\frac{T_d}{T}\right) + \frac{I_{n-1}}{T_i}\right]$$

Where Kp, Ti, Td is proportional gain, integral time and derivative time respectively. The T is the sampling time of PID.

Since the PID controller is essential for this whole control loop, we re-tune the parameters. The sampling time T is 0.065 s. The final Kp, Ti ,Td are 0.035, 0.8 and 0.0001 respectively.

Furthermore, to stop the robot when the job is done, a automatic stop condition is made. In the PID loop, a stop button is added to manually stop the program at any time. A local variable of this stop button is used in SMC loop to make this two loops stop together. In the SMC loop, when the scan function detects no file, the program will also stop. At the same time, the PID loop will stop due to the local variable of this error.

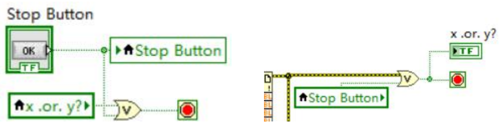


Figure 1.2 The stop function in PID and SMC loop

Firstly, a line trajectory is tested. From the simulation and experiment results below. Thus the error in pose will be little bit larger. The overall performance of pose tracking is acceptable, but the small error in y direction should be improved. The rise time of linear velocity is short while some overshoot occurs. The angular velocity is oscillatory. Compared to the simulation results, we can find that in reality, the velocity is much more oscillatory. The s1 and s2 chatter around the sliding surface of s=0. The deviation of realistic s is again larger than simulation.

If time permitted, the performance can be improved by tuning the parameters.

Figure 1.3 Robot pose

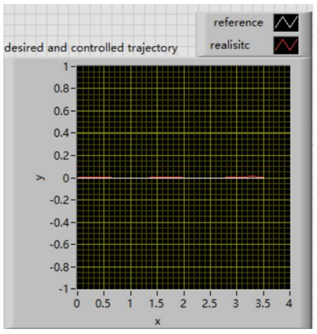


Figure 1.4 Linear velocity v

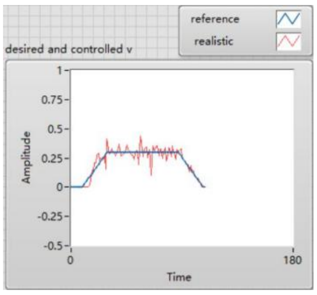


Figure 1.5 Angular velocity ω

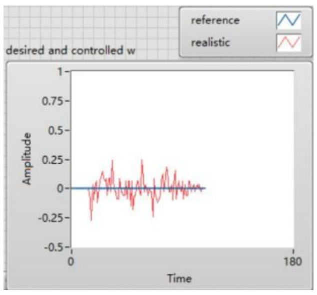


Figure 1.6 Error in x, y, θ

(ignore the final peak which caused by reset the state)

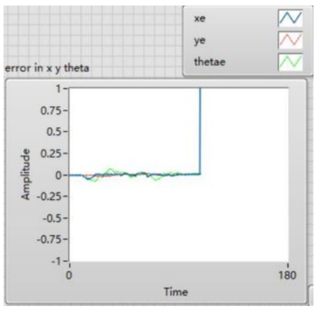
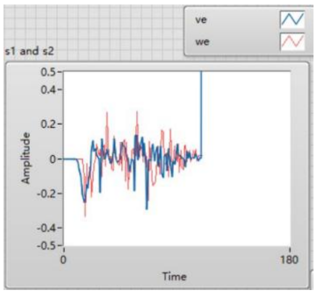


Figure 1.7 S_1 S_2



Task 2

A circle trajectory with diameter of 1.5 m is tested. The overall tracking performance of both pose and velocity are acceptable. The diameter of the real circle is larger than desired. The velocity curves are fast but still oscillatory. The error in theta seems a little bit large. A further tuning of k_2 may be necessary. The s1 and s2 chatter around the sliding surface of s=0. The deviation of realistic s is larger than simulation.

Figure 2.1 Robot pose

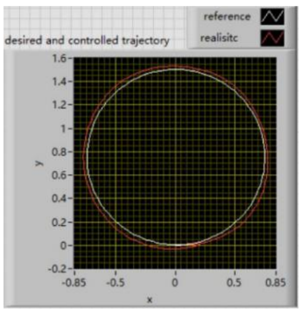


Figure 2.2 Linear velocity v

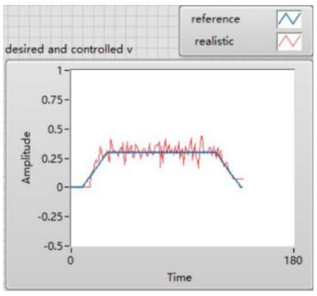


Figure 2.3 Angular velocity ω

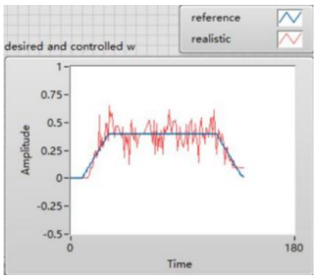


Figure 2.4 Error in x, y, θ

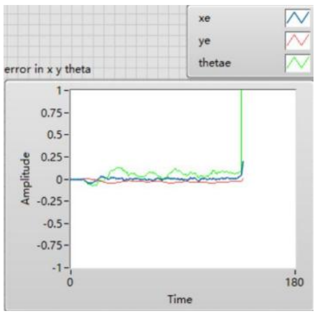
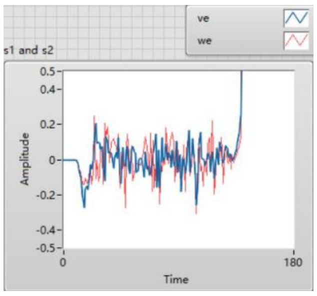


Figure 2.5 S_1 S_2



A larger circle trajectory with diameter of 2 m is tested. The test results are similar to the previous while the deviations are larger in every plot.

Figure 2.6 Robot pose

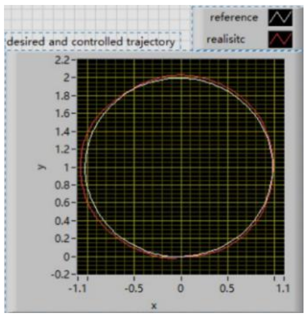


Figure 2.7 Linear velocity v

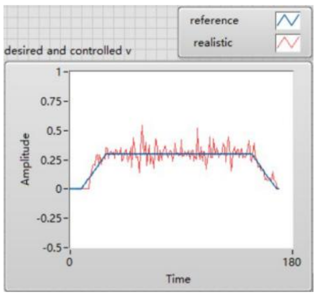


Figure 2.8 Angular velocity ω

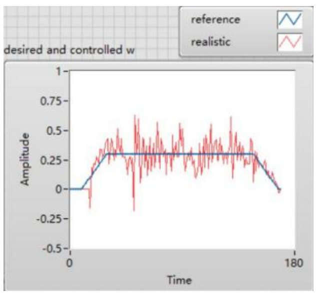


Figure 2.9 Error in x, y, θ

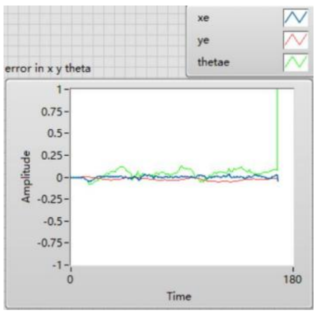
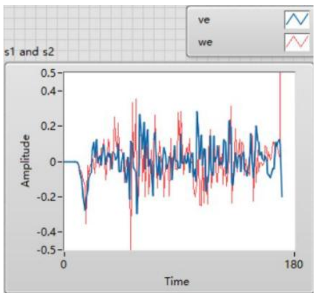


Figure 2.10 S_1 S_2



Task 3

A S-shape trajectory is tested for this robot. The overall tracking performance of both pose and velocity are acceptable. There are some visible error in the trajectory. The experiment results of velocity are much more oscillatory. The error in theta is relatively large. The s1 and s2 chatter around the sliding surface of s=0. The deviation of realistic s is larger than simulation.

Figure 3.1 Robot Pose

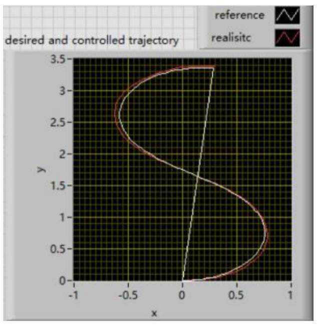


Figure 3.2 Linear velocity v

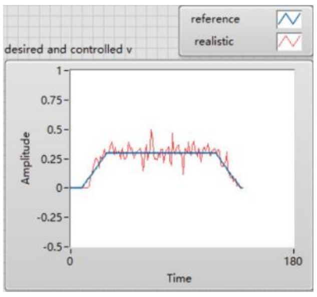


Figure 3.3 Angular velocity ω

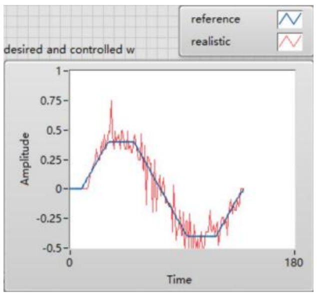


Figure 3.4 Error in x, y, θ

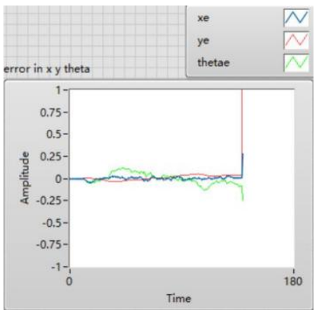
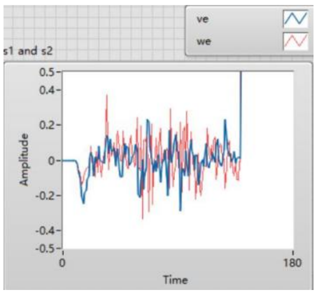


Figure 3.5 S_1 S_2



Task 4

Test the robot with some initial offset of 0.25m. The trajectory is circle with diameter of 1.5m. The robot reduce the offset little by little and finally follow the correct trajectory after almost a quarter's turn. The velocity has some overshoot and several oscillations at the beginning due to the large error. Correspondingly, the s curve is very negative than converges to zero. The large deviation of s gives system the force to pull back from offset.

After turning back to the trajectory, the behaviour is similar to the previous circle one. In other words, the initial error can be eliminated and does not affect the later trajectory.

Figure 4.1 Robot Pose

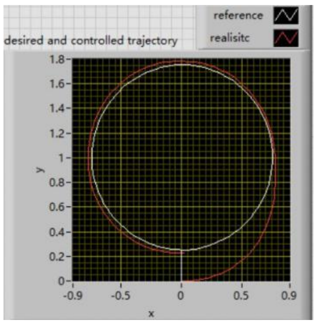


Figure 4.2 Linear velocity v

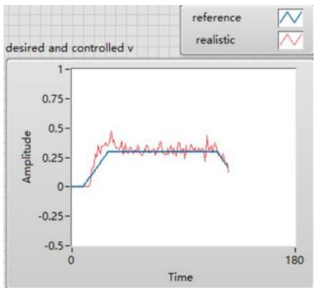


Figure 4.3 Angular velocity ω

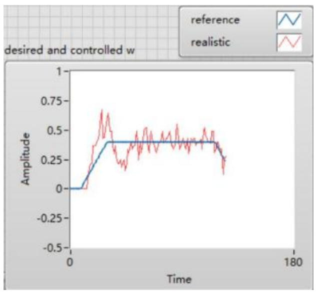


Figure 4.4 Error in x, y, θ

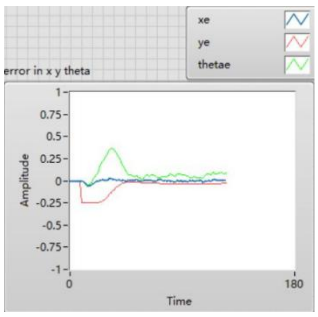
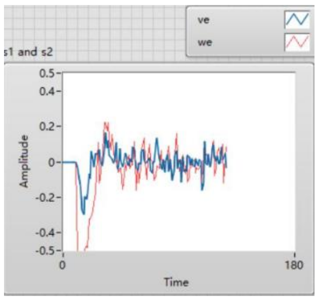


Figure 4.5 S_1 S_2



Change the offset to 0.5 meter. The deviation of s is larger due to this offset. However, the robot comes back to the desired trajectory in about the same time with the previous case. The larger deviation give the controller larger power to pull the robot back to the sliding surface. The exponential convergence property enables a similar performance with short converging time is obtained.

Figure 4.6 Robot Pose

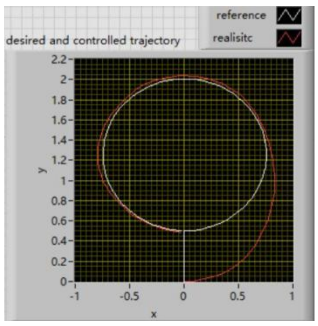


Figure 4.7 Linear velocity v

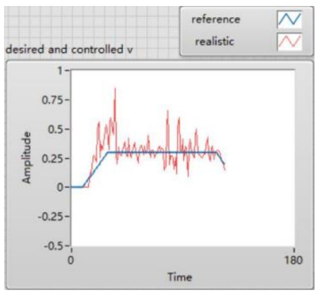


Figure 4.8 Angular velocity ω

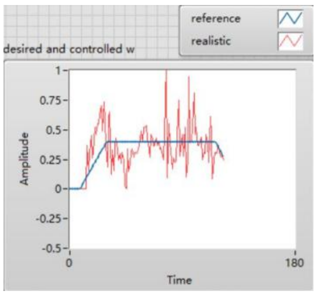


Figure 4.9 Error in x, y, θ

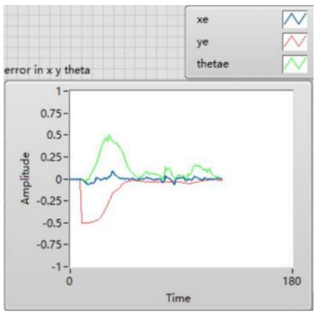
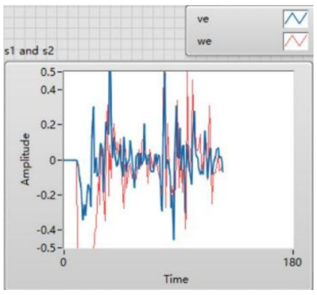


Figure 4.10 S_1 S_2



From the previous discussion, we can conclude that the SMC controller can deal with the large initial error.

Reference:

[1] Applied Control. Alexandru Stancu