The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Predictive Constrained Gain Scheduling For UGV Path Tracking in a Networked Control System

Bryan R. Klingenberg, Unnati Ojha, and Mo-Yuen Chow

Advanced Diagnosis Automation and Control (ADAC) Laboratory
North Carolina State University
Raleigh, North Carolina-27695

bryanklingenberg@gmail.com, uojha,chow@ncsu.edu

Abstract– **This paper presents a predictive gain scheduler for path tracking control in a networked control system with variable delay. The controller uses the plant model to predict future position and find the amount of travel possible with the global path as a constraint. Based on variable network conditions and vehicle trajectory's curvature the vehicle is allowed to travel farther on the current control signal while the vehicle trajectory matches the path constraint. This method uses path specific characteristics to evaluate the effectiveness of each generated control signal. By scheduling the gain on the control signal the vehicle tracking performance is maintained with an increase in network delay. The tracking time is decreased compared to other methods since the proposed control method allows the controller to look ahead and thus evaluate predicted effect of each control signal before scaling it. The proposed method is compared with existing delay compensation methods through simulation.**

I. INTRODUCTION

Path tracking control of unmanned ground vehicles (UGVs) is necessary to constrain UGV motion to a desired path. Many UGVs today communicate wirelessly with a remote controller that makes control decisions and shares global data with other controllers. These networked control systems are susceptible to delay which can adversely affect the path tracking performance of the UGV.

Most approaches to path tracking control of differential drive UGVs rely on the choice of an incremental reference position that the UGV tracks as it moves along its path. The UGV position is found either through dead reckoning, a vision system, or another locating system. The ubiquitous pure pursuit algorithm[1] uses a reference point on the path at a fixed distance ahead of the UGV and takes a circular trajectory to that point. The quadratic curve path tracking algorithm presented by Yoshizawa et al. in [2] provides a method for finding a variable reference point while approaching the path with a quadratic curve and variable velocity.

Another approach is to match the UGV with a desired pose by linearizing the system's differential equation, as Kanayama did in[3]. This resembles the classic Zhang tracker[4] which implements position and orientation tracking control. Finally a model predictive controller has been presented in [5] which implements predictive trajectory tracking. An overview of the stability of some path tracking controllers was completed by Ollero and Heredia in [6]. These path tracking algorithms are effective for non-holonomic, locally controlled UGVs.

The effect of curvature on path tracking has been approached in several ways. Peters and Iagnemma[7] present a model predictive controller to compensate for aggressive maneuvers and prevent wheel slippage by previewing the upcoming terrain and curvature. In [8] the curvature of the path controls the frequency of control signals sent to the UGV: higher curvature requires more updates. This heuristic approach mimics the way humans drive vehicles: curvy roads demand more attentive drivers In [9] the curvature ahead of the vehicle is detected and used to limit vehicle speed, thus limiting lateral motion. Glaser *et al.* use long range road data to prevent drivers from taking turns at unsafe speeds [10].

Finally others have studied the effect of network delay on network based path tracking controllers. A feedback preprocessor, similar to a smith predictor, is presented in [11] where network delayed feedback is compensated for using model prediction and network delay estimation. An elaboration on this approach is found in [12] which adds gain scheduling to prevent a control signal from causing path deviation based on network delay and trajectory curvature.

In this work the controller compensates for time varying network delay by predicting the future UGV positions. This prediction, with the constraint of the reference path, determines how far the UGV will accurately track the path given a current state and control signal. The allowable travel indexes a gain scheduling lookup table which, using network delay and trajectory curvature, will scale the control signal preventing future path deviation caused by network delay.

The remaining sections are organized as follows. Section II provides the system description of the UGV model, path tracking algorithm and network delay. Section III introduces the concepts of feedback preprocessing, gain scheduling and the new concept of predicted constrained gain scheduling. Section IV describes the testbed and how the simulations are set up. Finally simulation results are presented in Section V where the effectiveness of the proposed method is compared with that of existing techniques.

II. SYSTEM DESCRIPTION

A differential drive UGV with two driving wheels and one caster controlled using quadratic curve path tracking can be described using the kinematic model in (1)

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{bmatrix} \frac{\rho}{2} & \frac{\rho}{2} \\ \frac{\rho}{W} & -\frac{\rho}{W} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix}, \qquad (1)$$

where v_{ref} is the linear velocity of the UGV, ω_{ref} is the angular velocity, ρ is the radius of the drive wheels and W is the distance between the drive wheels. The ω_r and ω_l variables above represent the angular velocities of the right and left drive wheel respectively.

A. *Quadratic Curve Path Tracking Control*

The controller is given a path, ***p***, defined by a set of consecutive waypoints on the ground for the UGV to follow. The path is stored in the controller as a set of x and y coordinates and defined as $\boldsymbol{p}=[\boldsymbol{p}_x\ \boldsymbol{p}_y]$. This path is calculated prior to the execution of the path tracking algorithm.

To calculate the control signal for the UGV, the controller implements the quadratic curve path tracking algorithm which is a method for determining a variable look-ahead distance, based on the previous UGV control signal, as well as variable reference speed. As a reference point moves along the path and the UGV tracks the reference point, the UGV tracks the path. The reference point is located by finding the nearest point on the path using

$$D(\mathbf{a}(t_i), \boldsymbol{p}) = \min\left(\sqrt{\left(x(t_i)-\boldsymbol{p}_x\right)^2 + \left(y(t_i)-\boldsymbol{p}_y\right)^2}\right), \qquad (2)$$

where the UGV position $\boldsymbol{a}(t_i)$ is defined as:

$$\mathbf{a}(t_i) = \left[x(t_i) \quad y(t_i) \quad \phi(t_i)\right]. \qquad (3)$$

and looking ahead in the path a look-ahead distance d_0.

$$d_0 = \frac{d_{\max}}{1+\beta\left|A_{i-1}\right|}, \qquad (4)$$

where d_{max} is the maximum allowable lookahead distance, A_i is the quadratic coefficient defined in (5), and β is a tuning parameter used to prevent the UGV from cutting corners. The path index is increased and distance along the path is accumulated until d_0 has been travelled at which point the index is the reference point.

To track the reference point the controller finds a quadratic curve that passes through the reference point and has its vertex located at the UGV as seen below in Fig 1.

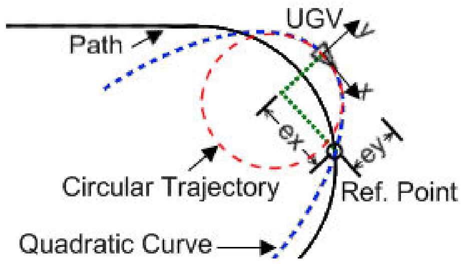


Fig 1: Quadratic curve path tracking showing the UGV and how the circular trajectory relates to the quadratic curve

A velocity and turn rate can then be calculated such that the UGV will move along the quadratic curve. As shown in [2] the velocity and turn rate can be simplified to the values shown below since the sampling period is sufficiently small.

$$A = \frac{e_y}{e_x^2}, \qquad (5)$$

$$\kappa = \operatorname{sgn}(e_x)\frac{\alpha}{1+|A|}, \qquad (6)$$

$$v_{ref} = \kappa, \qquad (7)$$

$$\omega_{ref} = 2A\kappa. \qquad (8)$$

$$A = \frac{\omega_{ref}}{2v_{ref}} \qquad (9)$$

The reference values are calculated by replacing the quadratic curve with a circle centered at the focus of the parabola with a radius defined by the distance from the focus to the vertex. The control mechanism is similar to pure pursuit[1] in that a circular control trajectory is used. The difference is a circular arc, defined by a quadratic curve, is fit to a varying rather than a constant reference point

The curvature is then the inverse of the radius of the circle that the UGV is being commanded to follow. Since the reference command is a circular trajectory, defined by (7)-(9) , the controller must frequently generate new quadratic curves and circular arcs. The UGV will follow the reference circular trajectory for a short period of time such that when all of the circular arcs are aggregated they approximate the quadratic curve which in turn approximates the desired path.

B. *Network Delay*

The UGV in consideration is controlled over a wireless network that causes the control signals and feedback signals to be delayed. The delay can be represented as the time that a signal takes to be transmitted from the controller to the robot (UGV), τ_{CR}, and the time that a signal takes to be transmitted from the robot to the controller, τ_{RC}. The sum of these delay times is the round trip delay, $\tau_{RTT} = \tau_{CR} + \tau_{RC}$. The nature of the delay is dependent on the type of network used, for example an IP network as studied in [13] or an IEEE 802.15.4 Zigbee network as studied in this work and in [14].

III. CONTROL PREPROCESSING BASED ON PREDICTED ERROR

A network based control scheme that requires frequent and timely updates to the control signal, such as quadratic curve path tracking will be affected by the network delay. In this section several methods for compensating for the delay are proposed including feedback preprocessing, gain scheduling middleware and the proposed predictive control scheme.

A. *Feedback Preprocessor*

A feedback preprocessor (FP) is used to predict the position of the UGV when the control value will be executed. The pertinent future position of the mobile robot is the position at

$t=t_i+\tau_{CR}(i)$ where t_i is the time when the control signal is calculated and $\tau_{CR}(i)$ is the estimated time that it will take for it to arrive at the UGV. The position predicted by the FP is used by the path tracking controller and is defined as $\hat{\mathbf{a}}(t_i+\tau_{CR})$. The position input to the FP, $\boldsymbol{a}(t_i)$, however, is the position of the UGV sampled before transmission to the controller at $t=t_i-\tau_{RC}(i)$ where $\tau_{RC}(i)$ is the time that it took for the signal to be communicated from the UGV to the controller.

Given that the input to the FP is $\boldsymbol{a}(t_i-\tau_{RC})$ and the desired output is $\hat{\mathbf{a}}(t_i+\tau_{CR})$ the UGV movement during this period can be approximated using the previous control value. The UGV is assumed to move with constant linear and angular velocities defined by the previous control value $\boldsymbol{u}(t_{i-1})$. Using the system dynamics given in (1) the future position is calculated by adding the change in position during $[t_i-\tau_{RC}, t_i+\tau_{CR}]$ with input

$$\mathbf{a}(t_i-\tau_{RC})=\left[x(t_i-\tau_{RC}) \quad y(t_i-\tau_{RC}) \quad \phi(t_i-\tau_{RC})\right], \tag{10}$$

and control value

$$\boldsymbol{u}(t_{i-1})=[v_{ref}(t_{i-1}) \quad \omega_{ref}(t_{i-1})]^T, \tag{11}$$

then, using the following estimations:

$$\tau=\tau_{RC}+\tau_{CR}, \tag{12}$$

$$\phi=\phi(t_i-\tau_{RC}), \tag{13}$$

the change in position over τ is:

$$\Delta_\tau\phi\cong\omega_{ref}(t_{i-1})\tau, \tag{14}$$

$$\Delta_\tau x\cong v_{ref}(t_{i-1})\tau[\cos(\Delta_\tau\phi)\cos(\phi)-\sin(\Delta_\tau\phi)\sin(\phi)], \tag{15}$$

$$\Delta_\tau y\cong v_{ref}(t_{i-1})\tau[\cos(\Delta_\tau\phi)\sin(\phi)+\sin(\Delta_\tau\phi)\cos(\phi)], \tag{16}$$

and the predicted position $\hat{\mathbf{a}}(t_i+\tau_{CR})$ is:

$$\begin{aligned}\hat{\mathbf{a}}(t_i+\tau_{CR})&=[x(t_i-\tau_{RC})+\Delta x,\\ &y(t_i-\tau_{RC})+\Delta y \quad \phi(t_i-\tau_{RC})+\Delta\phi],\end{aligned} \tag{17}$$

This value, $\hat{\mathbf{a}}(t_i+\tau_{CR})$, is forwarded to the controller for path tracking thus insuring appropriate control values upon arrival and execution. More discussion on FP is found in [11].

B. Gain Scheduler

A gain scheduler (GS) is implemented on the control signal after feedback preprocessing [15]. The GS adjusts the control signal using to network conditions and trajectory curvature preventing path deviation. Delayed packets may cause deviation and may cause the UGV to travel too far before a corrective signal arrives. The UGV is more open to error when not travelling in a straight line, or when that delay is large.

To evaluate the current control signal and determine what gain scheduling is needed the following cost function is used:

$$g(i+1)=\left\|\Delta_\tau\hat{\mathbf{a}}(i+1)\right\|_2. \tag{18}$$

This is the Euclidian norm of the change in UGV state caused by time τ. The state vector $\hat{\mathbf{a}}(i+1)$ can be expanded to its components and g becomes:

$$g(i+1)=\sqrt{\Delta_\tau\hat{x}^2(i+1)+\Delta_\tau\hat{y}^2(i+1)+\Delta_\tau\hat{\phi}^2(i+1)}, \tag{19}$$

We also define a cost function based on the absolute value of the UGV speed,

$$c(i+1)=\left|v_{ref}(i+1)\right|, \tag{20}$$

optimizing the control value $\boldsymbol{u}(i+1)$ by scaling it by K so that the new control value, $K\boldsymbol{u}(i+1)$, prevents UGV deviation. The optimal control signal requires maximizing the speed while avoiding deviation. To prevent deviation we will constrain $g\le\varepsilon$ limiting the change in UGV state caused before another controller update. The optimization problem then becomes:

$$g(i+1)\le\varepsilon, \tag{21}$$

$$\max c(i+1), \tag{22}$$

with the new control signal being $K\boldsymbol{u}(i+1)$ and since $\boldsymbol{u}(i+1)=[v_{ref} \ \ \omega_{ref}]$ we can scale (9) with K as:

$$A=K\frac{\omega_{ref}}{2v_{ref}}, \tag{23}$$

and using (14)-(16) with (23) we can substitute these into (18) and rewrite g as a function of A, K, and τ:

$$g(i+1)=\sqrt{\frac{2-2\cos(AK\tau)}{A^2+(AK\tau)^2}}. \tag{24}$$

In operation, after the control signal is generated the value of A is known and τ_{RTT} can be predicted then optimization can be carried out using the bisection algorithm to find the ideal value of K to maximize speed with a limit on UGV deviation. To increase performance this can be computed offline for known ranges of A, τ_{RTT} and stored in a lookup table. If we set the allowed deviation ε to 0.12 and if A is known to be in the range of [0,14] in normal operation and τ_{RTT}, is known to be in the range of [0,1] then K can be calculated in this range.

The optimal gain surface, an example of which can be seen in Fig 2, will reduce the control signal as delay increases and as the trajectory curvature increases. The optimal gain selected from the gain table will limit the deviation of the UGV to ε while operating at the highest speed possible. More discussion on the gain scheduler can be found in [15].

C. Predictive Constrained Gain Scheduling

The purpose of GS is to prevent a control signal from causing UGV deviation before an updated control signal is given. The GS method above is effective since it limits all UGV motion to be less than ε. Setting ε low allows accurate tracking for any type of path [15]. Predictive constrained gain scheduling(PCGS) allows ε to adapt until the point that deviation begins. The controller does this using data about the future path from either the supervisor or onboard sensors. Data regarding network conditions and the next control are also used. The safety region is determined by the required tracking accuracy and is analogous to the width of a road.

The gain table calculated for GS in [15] is overly conservative since it must work for all path conditions. Since we will choose an ε dynamically we can then calculate the gain table for different values of ε in the range of [0.1,1]. If ε is allowed to go less than 0.1 the UGV will stall and halt progress down the path. The upper limit of ε was chosen to be 1 as it allows the value of K to be 1 for all values of A when τ_{RTT} is close to 0 and for K to be 1 for all values of τ_{RTT} when A is close to 0. These gain tables, some shown below in Fig 2, are stored and used as lookup tables during path tracking control. Using a larger ε will allow the UGV to travel farther before it requires another control update.

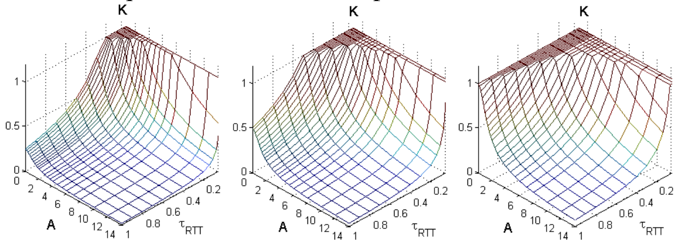
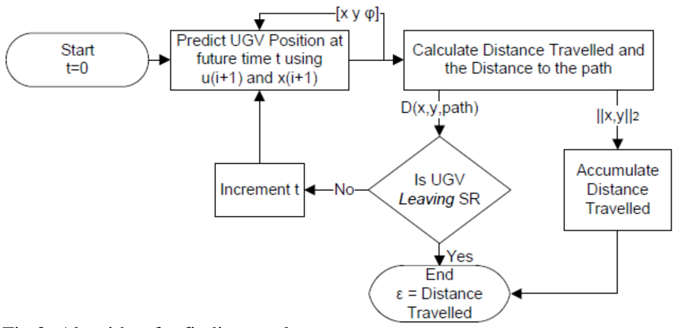


Fig 2: Optimal gain tables for $\mathcal{E}$ values of 0.25, 0.75 and 1

The issue at hand is to dynamically find an optimal value for ε based on the current UGV position and the future track of the path. To do this the future position is predicted using the next control value as a constraint.


Start t=0
Predict UGV Position at future time t using u(i+1) and x(i+1)
[x y φ]
Calculate Distance Travelled and the Distance to the path
D(x,y,path)
||x,y||2
Is UGV Leaving SR
No
Increment t
Accumulate Distance Travelled
Yes
End ε = Distance Travelled
Fig 3: Algorithm for finding ε value to use

The control signal, current state, and UGV model are used to predict UGV position incrementally until the UGV exits the safety region. The time is incremented from the current time t_i by $j\Delta\tau$ where j is the step number and $\Delta\tau$ is the step resolution. The point where the UGV exits the safety region represents the point where the control signal is no longer effective at tracking the path. Choosing ε to be the distance travelled to reach the exit point allows the UGV to travel farther in one time step provided that the travel follows the path.

$$\varepsilon = \sum_{j=1}^{n} D(\hat{a}_u(j), \hat{a}_u(j-1)), \qquad (25)$$

$$\hat{\mathbf{a}}_u(j) = \mathbf{a}_u(t_i + j\Delta\tau). \qquad (26)$$

As shown in Fig 3, the ε value can be calculated using (25) where n is the point the UGV leaves the safety region. The future position is predicted using (14)-(17) in a similar fashion to feedback preprocessing except that the value of τ is replaced with $j\Delta\tau$. This is summarized in Fig 3.

For example, in Fig 4 two UGVs are shown in different paths. The bottom UGV is travelling in a straight line approaching a curve. If ε is calculated for this case using the method shown in Fig 3, ε can be increased to 0.6. This value is the predicted travel of the UGV that is constrained by the control signal $\boldsymbol{u}(i+1)$. The top UGV has a control trajectory that matches the path; previous methods would limit the UGV travel through this curve, but since we can predict that the UGV will travel within the safety region, ε is increased to 0.9

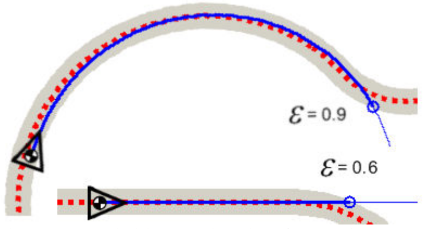


Fig 4: Predicted UGV travel (blue) and path safety region (grey) showing the effect of different epsilon values on different types of paths.

IV. TESTBED

The control systems described above is tested on a simulated UGV. The UGV to be simulated has been built using the LEGO mindstorms NXT TriBot as a base with the controller replaced with a SPOT controller from Sun Microsystems. Each UGV has a PI controller to control left and right wheel speed using encoder feedback and an 802.15.4 radio to communicate to a supervisory controller with communication delays as described in [14].

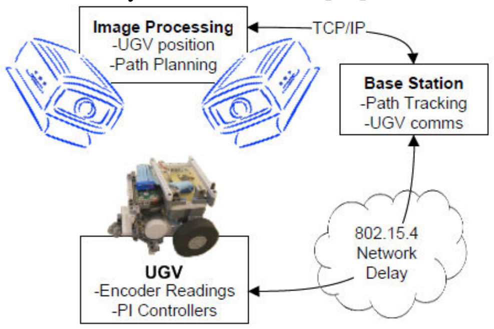


Fig 5: System configuration and communications diagram

The supervisory controller implements the quadratic curve path tracking algorithm, FP, GS, and PCGS. The path is generated before operation and control signals are sent to the UGV with a simulated variable exponential delay distribution with a given minimum and average delay.

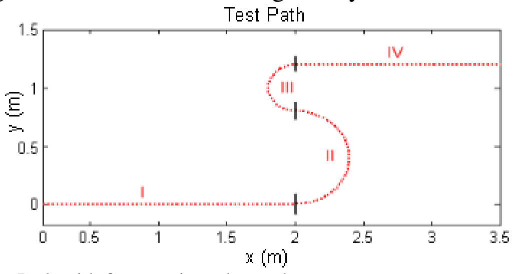


Fig 6: Test Path with four sections denoted

The test path is composed of four segments: an initial straight path (I) 0.5m long, a 0.4m radius curved section (II), an inflection point into a 0.2m curved section followed by a 0.5m straight section (IV). This is a representative path that contains varying path curvature conditions to test the UGV.

V. RESULTS FROM SIMULATION

To validate the methods described above the system has been modeled and simulated using MATLAB/Simulink. All three methods described in section III have been simulated to evaluate their performance. Using the test path shown in Fig 6 with the UGV starting at $\mathbf{a}(t_0)$=[0 0 0]. Each method is simulated with average round trip times of 70, 200, 400, and 600ms to represent a wide range of operating conditions.

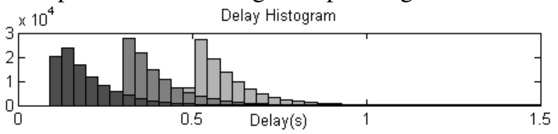


Fig 7: Generalized exponential delay distributions for average round trip times of 200, 400 and 600ms

The delay distributions shown in Fig 7 represent the three test cases with additional delay. The 70ms average round trip time represents the minimum communication delay achievable in this system with 802.15.4 protocol.

The base case simulation with no additional delay and with RTT of 600ms are shown below in Fig 8

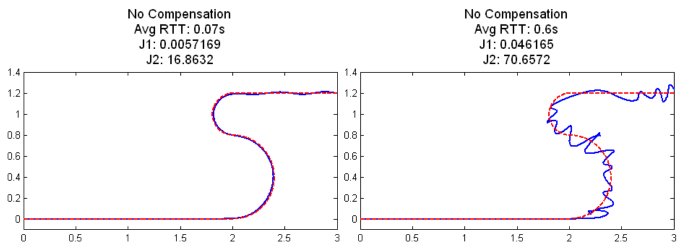


Fig 8: UGV results with no compensation. No additional delay on left and 600ms of delay on right.

The UGV tracks the path quickly and with little error when there is no delay. When delay is added the UGV tracks segment I with ease. This is because the control signal is following the straight path and the delay has no effect. PCGS takes advantage of this. After segment I the UGV begins to oscillate severely around the path as it tracks resulting in high tracking error as the control signal is causing the UGV to overshoot and travel too far before a new signal arrives.

To evaluate the performance of each simulation and to compare the results between methods, two performance metrics have been calculated: the accumulated tracking error and the total time. The metrics, J_1 and J_2, are shown below in (27) and (28).

$$J_1 = \frac{\int_{t_0}^{t_f} (\boldsymbol{y}(t) - \boldsymbol{p})dt}{\sum D(\boldsymbol{y}(j), \boldsymbol{y}(j-1))}, \tag{27}$$

$$J_2 = t_f - t_0, \tag{28}$$

where J_1 is the accumulated error, the numerator is the area between the UGV travel and the path $\boldsymbol{p}$. The denominator is the total distance travelled by the UGV. The metric J_2 is the total time that the UGV required to reach the destination.

A. Results

The methods described in section III above have been simulated to evaluate their effectiveness at mediating the problems illustrated in Fig 8. Using the performance metrics J_1 and J_2 the results can be compared in Fig 9.

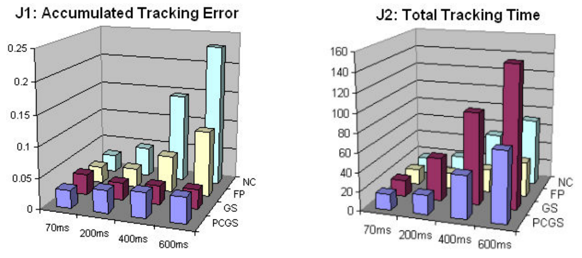


Fig 9: Accumulated error and total time for all test case simulations

Fig 9 shows that feedback preprocessing reduced tracking error and increased tracking time over the base case by eliminating much of the oscillation after segment I. We can also see that both GS and PCGS, which both use feedback preprocessing as well, reduce the tracking error when delay is increased. The tracking error with delay is relatively close to the tracking error without delay for both of these methods. The key difference between GS and PCGS can be seen in the J_2 results from Fig 9. GS and PCGS both achieve reduced tracking error at the expense of tracking time however PCGS is shown to produce accurate results, as seen below in Fig 10, in about half the time as GS.

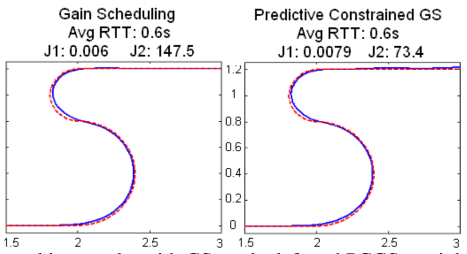


Fig 10: Path tracking results with GS on the left and PCGS on right.

B. Discussion

To illustrate how PCGS is able to reduce the tracking time while maintaining low tracking error we can compare accumulated error and the K output of the gain scheduler.

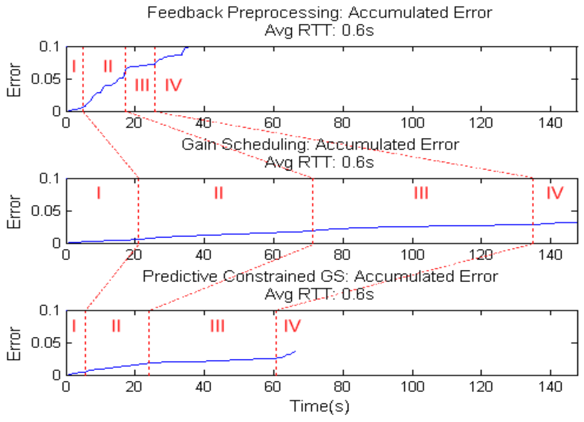


Fig 11: Accumulated error for FP, GS, and PCGS.

In Fig 11 the accumulated error is shown for the three methods when simulated with an average τ_{RTT}=0.6s.The vertical lines in the figure denote the transition of path segments with the segment numbers indicated. The feedback preprocessing results shown above are completed in the least time however the total error is more than 2.5 times greater than either GS or PCGS results. Both GS and PCGS provided significant accumulated error improvement over FP but PCGS is able to complete the trajectory in less than half the time of GS. Fig 11 demonstrates that PCGS completes segment I in 5.5 seconds versus 21 seconds using GS. This is because of the effect illustrated above in Fig 4 where ε is allowed to increase. The next segment is also completed in much less time with similar tracking error when using PCGS. This is because of the effect illustrated in where ε is allowed to increase because the control signal matches the path curvature.

These same effects and the cause of the increased speed are evident in Fig 12 when looking at the K gain value used in each method. The vertical lines indicate the transitions between different path segments. For the straight segment at the beginning, even though delay is high, PCGS does not scale the control at all since the path is straight whereas GS does. The same effect is evident for segment II where PCGS scales the control signal by about 0.3 and GS scales it by 0.1.

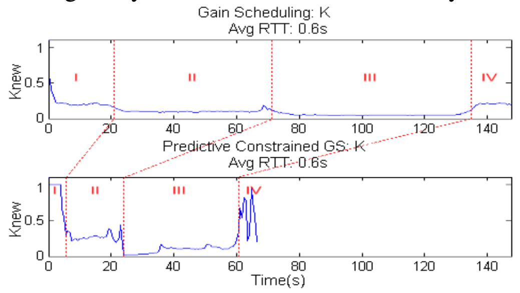


Fig 12: K for GS and PCGS.

Two notables regarding PCGS: the response after entering segment III and the response after entering segment IV. PCGS cannot predict around an inflection point. Thus the gain is low since the predicted position moves away from the path rapidly. In future work behavioral control could be used as a special case to mitigate this. In segment IV the UGV experiences a slight overshoot when the path transitions from curved back to straight. This can be seen as a steep jump in accumulated error in Fig 11 and as a dip in K in segment IV of Fig 12.

VI. CONCLUSION

Path tracking performance, in terms of tracking error and tracking time, for three control methods has been presented in this research. The GS method with FP is effective at reducing tracking error however, it is overly conservative. When the PCGS method is implemented the controller can predict the system response and allow more travel providing that the travel is constrained to the path. To do this the predicted UGV trajectory is compared with future path data. The data and predicted trajectory allow increased UGV travel in less time.

ACKNOWLEDGMENT

This research was supported by NSF-ECS-0823952 “Impaired Driver Electronic Assistant (IDEA)” and Department of Transportation Massive Sensor Based Congestion Management System for Transportation Systems .

REFERENCES

[1] C. C. Coulter, "Implementation of the pure pursuitpath tracking algorithm," Robotics Institute, Pittsburgh, PH, Tech-Report January 1992.

[2] K. Yoshizawa, H. Hashimoto, M. Wada, and S. Mori, "Path tracking control of mobile robots using a quadratic curve," in *Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE*, 1996, pp. 58-63.

[3] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 1990, pp. 384-389 vol.1.

[4] Y. L. Zhang, S. A. Velinsky, and X. Feng, "On the Tracking Control of Differentially Steered Wheeled Mobile Robots," *Journal of Dynamic Systems, Measurement, and Control*, vol. 119, pp. 455-461, 1997.

[5] G. Klancar and I. Skrjanc, "Predictive Trajectory Tracking Control for Mobile Robots," in *Power Electronics and Motion Control Conference, 2006. EPE-PEMC 2006. 12th International*, 2006, pp. 373-378.

[6] A. Ollero and G. Heredia, "Stability analysis of mobile robot path tracking," in *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, 1995, pp. 461-466 vol.3.

[7] S. C. Peters and K. Iagnemma, "Mobile robot path tracking of aggressive maneuvers on sloped terrain," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, 2008, pp. 242-247.

[8] D. Marshall, D. Roberts, D. Delaney, S. McLoone, and T. Ward, "Dealing with the Effect of Path Curvature on Consistency of Dead Reckoned Paths in Networked Virtual Environments," in *Virtual Reality Conference, 2006*, 2006, pp. 293-294.

[9] M. Netto, J. M. Blosseville, B. Lusetti, and S. Mammar, "A new robust control system with optimized use of the lane detection data for vehicle full lateral control under strong curvatures," in *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, 2006, pp. 1382-1387.

[10] S. Glaser, L. Nouveliere, and B. Lusetti, "Speed Limitation Based on an Advanced Curve Warning System," in *Intelligent Vehicles Symposium, 2007 IEEE*, 2007, pp. 686-691.

[11] R. Vanijjirattikhan, M. Y. Chow, and T. Yodyium, "Feedback preprocessed unmanned ground vehicle network-based controller characterization," in *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*, 2004, pp. 2333-2338 Vol. 3.

[12] R. Vanijjirattikhan, M.-Y. Chow, P. Szemes, and H. Hashimoto, "Mobile Agent Gain Scheduler Control in Inter-Continental Intelligent Space," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 1115-1120.

[13] Y. Tipsuwan and M.-Y. Chow, "Model predictive path tracking via middleware for networked mobile robot over IP network," in *American Control Conference, 2004. Proceedings of the 2004*, 2004, pp. 4307-4312 vol.5.

[14] M. Petrova, J. Riihijarvi, P. Mahonen, and S. Labella, "Performance study of IEEE 802.15.4 using measurements and simulations," in *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, 2006, pp. 487-492.

[15] Y. Tipsuwan and M.-Y. Chow, "Gain scheduling middleware for networked mobile robot control," in *American Control Conference, 2004. Proceedings of the 2004*, 2004, pp. 4313-4318 vol.5.