2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Slip-aware Model Predictive Optimal Control for Path Following*

Venkataramanan Rajagopalan[2], Çetin Meriçli[1,2] and Alonzo Kelly[1,2]

***Abstract*— Traditional control and planning algorithms for wheeled mobile robots (WMR) either totally ignore or make simplifying assumptions about the effects of wheel slip on the motion. While this approach works reasonably well in practice on benign terrain, it fails very quickly when the WMR is deployed in terrain that induces significant wheel slip. We contribute a novel control framework that predictively corrects for the wheel slip to effectively minimize path following errors. Our framework, the Receding Horizon Model Predictive Path Follower (RHMPPF), specifically addresses the problem of path following in challenging environments where the wheel slip substantially affects the vehicle mobility. We formulate the solution to the problem as an optimal controller that utilizes a slip-aware model predictive component to effectively correct the controls generated by a strictly geometric pure-pursuit path follower. We present extensive experimental validation of our approach using a simulated 6-wheel skid-steered robot in a high-fidelity data-driven simulator, and on a real 4-wheel skid-steered robot. Our results show substantial improvement in the path following performance in both simulation and real world experiments.**

I. INTRODUCTION

The ability to effectively navigate in unstructured rough terrain is very important for wheeled robots. Generating optimal, navigable paths is mission critical, and has been a widely popular research subject for many decades. However, following a given path is equally important and challenging as the navigation performance ultimately depends on how well the generated paths can be followed. For a WMR, the path following problem is the challenge of selecting a sequence of control actions that minimizes the error in pose of the robot with respect to a reference path, given the actuation constraints, and local terra-mechanical properties. This work, to the best of our knowledge, is the first time that a learned slip model has been incorporated in an optimal control framework for WMRs.

A. Motivation

Consider a robot trying to maneuver into a tight spot as shown in Fig. 2. It is duly noted that the terrain is slippery, which makes the robot to understeer. Unfortunately the robot is unaware of this condition and it continues to turn harder and harder. At some point, it reaches its minimum turning radius and hence cannot avoid hitting the obstacle on the right if it continues on that path. Feedback control mechanisms alone cannot recover from this failure as the error is persistent. Many such situations as described above can be avoided if we can estimate future errors and address them early.

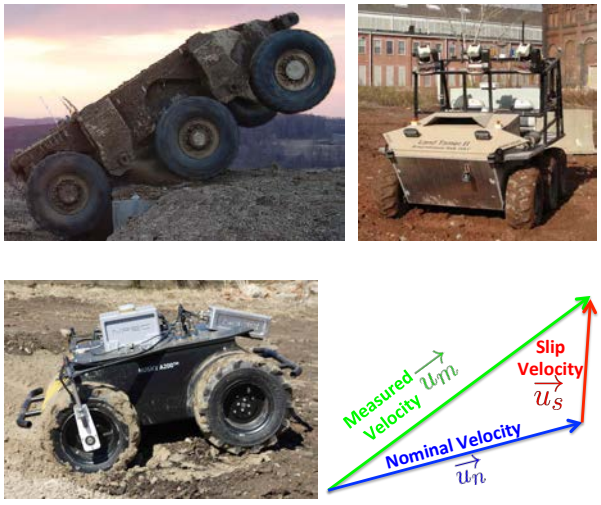

Fig. 1: Efficient slip modeling and compensation is critically important especially for the skid-steer robots on challenging rough terrain such as CMU's Crusher (top left), and our experimental validation platforms LandTamer (top right) and Clearpath Husky (bottom left). We define slip as the difference between the nominal and measured velocity vectors (bottom right).

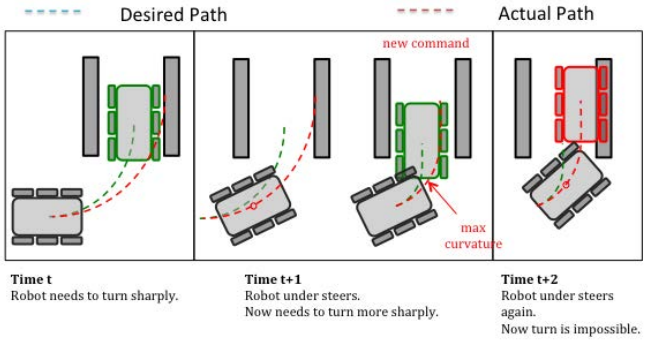

Fig. 2: A skid-steer robot trying to plan a challenging maneuver and failing due to the lack of slip compensation.

II. RELATED WORK

There has been substantial research into the problem of developing efficient and robust path following controllers for WMRs. Due to space constraints, here we only briefly mention some relevant studies. An early implementation

*This work was supported by the U.S. Army Research Office under the Vehicle-Ground Model Identification program Grant Number: W911NF-09-1-0557 and the Robotics Collaborative Technology Alliance program

[1]NREC, [2]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-2980, USA venkat@cs.cmu.edu, cetin@cmu.edu, alonzo@cmu.edu

This paper is best viewed in color when printed on paper

by Coulter used a pure-pursuit approach where a single look-ahead point was tracked [1]. This approach has had significant success in outdoor robots and is still being used today. Kelly and Nagy described a model predictive trajectory generation algorithm that generated a set of parameterized controls that could be directly executed by a vehicle controller [2]. Howard and Kelly described a Receding Horizon Model Predictive Control (RHMPC) that followed paths and avoided obstacles through geometric singularities and discontinuities [3]. Lacaze et al. used Ego Graphs to generate a set of layered trajectories that could be directly executed by the vehicle controller [4]. Other techniques such as Rapidly Exploring Random Trees (RRTs) [5] have also been successfully used to generate controls that can directly be executed by the vehicle controller. Ishigami et al. proposed a terra-mechanics based approach to compensate for lateral and longitudinal wheel slip for a WMR operating in loose terrain [6]. Hemlick et al. describe a path following approach that included a reactive slip compensation component [10]. In their approach, they use two separate control loops. The first one is a carrot heading compensation controller and the second one is a controller that generates a steering angle offset in the same direction for all wheels (crabbing controller). While this is closest to our body of work, their system is myopic as it only considers what is effectively instantaneous slip. Our approach, on the other hand, reasons about slip on a much longer time horizon, thereby effectively mitigating its effects by acting early.

III. Technical Approach

A. Wheel Slip Model

We use a velocity driven model to model slip [7], given by (1).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} c\theta & c\theta s\beta s\gamma - s\theta s\gamma & 0 \\ s\theta c\beta & s\theta s\beta s\gamma - c\theta c\gamma & 0 \\ 0 & 0 & \frac{c\gamma}{c\beta} \end{bmatrix} \left(\begin{bmatrix} V_x \\ V_y \\ V_\theta \end{bmatrix} + \begin{bmatrix} \delta V_x \\ \delta V_y \\ \delta V_\theta \end{bmatrix} \right) \quad (1)$$

- $\dot{x}$, $\dot{y}$ and $\dot{\theta}$ are the pose rates of the system in a ground-fixed frame
- γ, β and θ are the roll, pitch and yaw angles respectively
- c and s are the cos() and sin() of the above mentioned angles
  V_x, V_y and V_θ are the velocities of the WMR in body-fixed frame
- δV_x, δV_y and δV_θ are the wheel slip components.

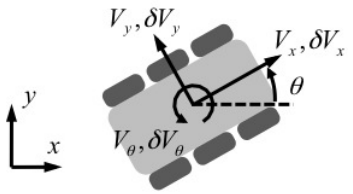

Fig. 3: Velocity driven vehicle model with slip.

Barring ballistic motion (which is not modeled here), (1) is a constrained differential equation - out of the six degrees of freedom, the constraint that the vehicle should be on the terrain fixes the pitch, roll and the elevation degrees of freedom. The only remaining degrees of freedom of the platform are the x, y locations and the yaw.

B. Identifying Wheel Slip

We follow the approach described in [7] to estimate the components of wheel slip. In that approach, the system dynamics is linearized to capture, to the first order, the error in the predicted 6 DOF pose of the WMR given a disturbance to the input (which is assumed to be caused by wheel slip). This linearized dynamics as defined by (2) is integrated over the input paths and compared against ground truth measured using a post-processed RTK-GPS Enabled INS solution in an Extended Kalman Filter framework to calibrate the systematic and stochastic errors attributed to wheel slip.

$$\delta \dot{\underline{x}} = F(t)\delta \underline{x}_t + G(t)\delta \underline{u}_t \quad (2)$$

$F(t)$ and $G(t)$ are the path dependent pose and control Jacobians, and are taken with respect to the pose vector $\underline{\rho} = [x, y, \theta]^T$ and the input vector $\underline{u} = [V_x, V_y, V_\theta]^T$.

In addition to learning the wheel slip characteristics of a WMR for a specific terrain, we also learn the first order response of its power train (the delay and the rise time for a discrete step input in control to take effect at the wheels).

C. Problem Formulation

In our formulation, the desired path is an ordered sequence of (x_d,y_d) arbitrarily spaced locations that the WMR is expected to traverse at speeds specified along the line segment defined by two successive locations. While there are numerous approaches to generate feasible paths ([4], [5], [10]), we use a variation of [1] in our framework. We define feasible path as an ordered set of tuples (x^*, y^*) that are traversed by the WMR in response to the set of commanded curvature and forward velocity actions (k^*, ${V_x}^*$) generated by the pure pursuit follower.

1) Pure Pursuit Path Follower: A pure pursuit path follower is a technique that finds a sequence of constant curvature arcs from a family of infinite such arcs, which, when executed, will make the WMR's executed path converge to the desired path asymptotically. A basic version of the pure pursuit path follower is described in Algorithm 1.

2) LQR Tracking: We pose the problem of predictively removing the effects of slip as a constrained optimization problem under the LQR tracking framework. We use the aforementioned pure pursuit follower algorithm to generate the reference trajectory which is a tuple with the following elements

- Planar location: x, y (meters)
- Yaw: θ (radians)
- Linear Velocity: $\dot{x}$, $\dot{y}$ (meters per second)
- Angular Velocity: $\dot{\theta}$ (radians per second)
- Desired linear velocity: $\dot{x_d}$ (meters per second)
- Desired angular velocity: $\dot{\theta_d}$ (radians per second)

Algorithm 1: Pure Pursuit (path, min turn radius, look-ahead distance)

while *not done* **do**

- Get the location and orientation of the WMR
- Find the closest point on the desired path from the WMR
  - Compute the normal projection of the WMR's location to the set [a] of line segments
  - Closest point is defined as the normal projection with the smallest absolute distance
- Starting from the closest point, find the point along the desired path that is look-ahead distance further along the path
- Compute the radius [b] of the circle subject to the constraint that the locations of the WMR and the look-ahead point lie on its circumference exactly

end

Starting at the reference trajectory's origin, for every time step in the desired (feasible) trajectory, we forward simulate the control generated by the pure pursuit path follower, this time, taking into consideration, the effects of slip and first order power train dynamics. This process generates the $\underline{x}_k$ which is generally different from the feasible state trajectory in (3). More formally, for a sequence of desired states $\underline{x}^*_{1..n}$ and controls $\underline{u}^*_{1..n}$, the system is linearized about that operating point (the current state and desired control) and the loss functional given by (3) is optimized subject to (4)

$$J = arg\min_{u} \sum_{k=1}^{n} (\underline{x}^*_k - \underline{x}_k)^T Q (\underline{x}^*_k - \underline{x}_k) + (\underline{u}^*_k - \underline{u}_k)^T R (\underline{u}^*_k - \underline{u}_k) \quad (3)$$

$$\underline{x}^*_{k+1} - \underline{x}_k = A_k(\underline{x}^*_k - \underline{x}_k) + B_k(\underline{u}^*_k - \underline{u}_k) \quad (4)$$

where $Q \succeq 0$ and $R \succ 0$ are the state and control penalty matrices respectively and A and B are the first order approximations of the system dynamics and the controls (linearized about the current operating point).

The solution to (3) is given by:

$$K_k = (R + B_k^T P_{k-1} B)^{-1} B^T P_{k-1} A \quad (5)$$

$$P_k = Q + K_k^T R K + (A + BK_k)^T P_k - 1(A + BK_k) \quad (6)$$

(5) is also called Kalman Gain [8] and is solved concurrently with (6) which is known as Ricatti ODE.

[a] In the first pass all the line segments formed between successive way points are considered and in subsequence passes, only a restricted subset is searched

[b] This is computed in closed form. Computed radius values s.t $|Radius| \leq MaxTurnRadius$ are handled as a special case

The new control is linear in state and is given by:

$$(\underline{u}^*_k - \underline{u}_k) = -K_k(\underline{x}^*_k - \underline{x}_k) \quad (7)$$

Finally, the quantity given in (7), also called *delta control* is added to the desired control $\underline{u}^*_k$ to produce the final control.

3) Implementation: Algorithm 2 briefly enumerates the steps that are involved in the RHMPPF. The input to the algorithm is the desired path along with the slip parameters identified beforehand for the terrain under consideration and the time horizon. The output of the algorithm is a modified control signal for the current time step. This control signal is passed along to the vehicle controller at a deterministic rate (20 Hz in our case). The whole process repeats until the WMR is i) within a certain radius of the end of the path, or ii) is further than a specified normal distance away from the desired path.

Algorithm 2: RHMPPF (path, slip parameters, horizon)

while *not done* **do**

- Update vehicle state
- Generate the reference path and controls up to the desired time horizon
- Generate the predicted path under real slip with reference control
- Run the LQR tracker to compute delta control
- Pass along the modified control to the vehicle controller

end

IV. Experimental Setup and Evaluation

We conducted a series of trials in simulation as well as in the real world to validate our approach. In all our simulation and real-world trials, we used the cross track error between the desired and actual paths as the performance metric.

A. Data Driven Simulation

In order to make our simulation realistic enough, we chose a data driven approach to building and simulating the terrain and the vehicle of choice.

1) Model Building: A modified LandTamer 6x6 all-wheel drive skid-steered vehicle, manufactured by PFM Manufacturing, retrofitted with forward looking lidar and camera units, wheel encoders and a NovAtel SPAN INS solution (with real time RTK corrections) served as the platform of choice for data collection. 3D Terrain and mobility models for the simulation were generated from the data that was collected with this platform from a real world site of approximately 1 square kilometer in area. A small portion of the site, approximately 100 square meters in area was then chosen for simulation (Fig. 4a). The data for building the terrain geometry was collected using the SICK Lidars mounted on a nodding mechanism. A 2.5D height map was generated with the geo-registered lidar data. This map served as the

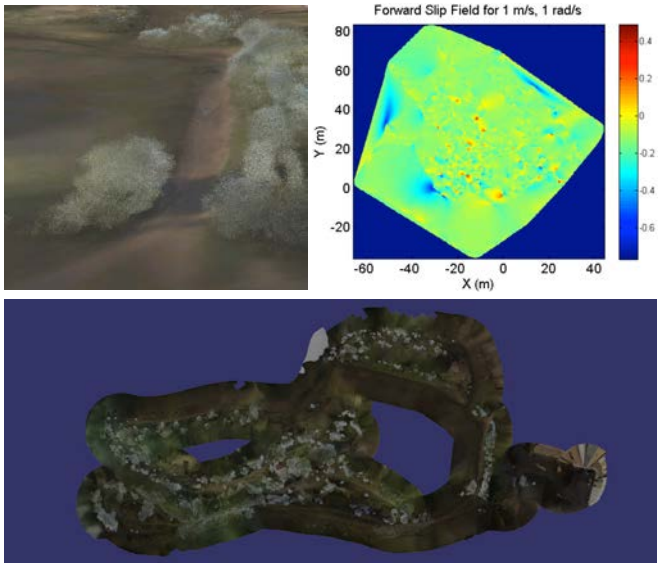

Fig. 4: A close up view of the test site (top left), the residual slip field (top right), and birds eye view of the test site (bottom).

"ground truth" for the vehicle mobility simulator. Fig. 4c is an overhead view of a colorized model of the real world test site.

The data collection effort for mobility modeling was different to the one used for terrain geometry modeling even though they both covered the same area of interest in general. For this effort, the WMR was driven in such a way that we had good coverage in the linear and angular velocity space. The result of the calibration is two fold.

- Systematic slip parameters - this captures the "average" slip and is globally applicable over the entire terrain. These parameters are stored in a configuration file much like the other parameters such as vehicle dimensions, min. turn radius etc.
- Residual slip parameters - this captures the "residual" slip, the slip that is not explained by the global slip parameters for any given location of the modeled terrain. Thus, for every parameter dimension in the systematic slip, there is a 2D geo-referenced grid of residual slip information. This is saved in a multi-layered GeoTIFF file. Fig. 4b is a visual representation of one such layer of data.

2) Implementation: We leveraged an in house developed high-fidelity, data-driven simulation framework for testing Unmanned Autonomy Systems (UAS) to develop and tune the pure pursuit path tracker and the RHMPPF. This framework mimics (at the network packet level) the data and the control interfaces found on the LandTamer. This is done deliberately to achieve compatibility with the real robot.

The vehicle mobility simulation is implemented as a plug-in module in C++, runs at 100Hz, and uses a Boost-IPC shared memory model to publish state information and obtain vehicle control. The RHMPPF is implemented as a separate module in C++ that shares the forward simulation components with the vehicle mobility simulator and runs as a separate process at 20Hz.

3) Parameter Tuning: For the pure pursuit path follower, we hand-tuned the two parameters described below.

- Fixed Look-Ahead Distance (m) - this is the constant distance that we "look-ahead" along the desired path from the closest point of the WMR on the path to determine the look-ahead point
- Speed Based Look-Ahead Distance(m) - this is a linear function of the desired speed along the path. This quantity is added to the fixed look-ahead distance to compute the final look-ahead point. This was implemented as a simple look-up table.

For the LQR tracking problem, we used Bryson's method [9] for tuning the the state(Q) and control(R) penalty matrices. We retained all the common parameters (such as parts of the speed-based look-ahead distance and the position penalty terms in the Q matrix) and tuned the other parameters separately for the real and simulated platforms.

4) Test Scenarios: Three test scenarios of varying slip characteristics are used in the trials as follows:

- Dirt (packed with occasional gravel). This was the terrain on which we collected mapping and mobility data. Among our three classes, this offers the best mobility characteristics.
- Adversarial situation. This made up case was setup to mimic a hardware failure in the system. In this case, it is always hard for the WMR to turn to one side (left). We effected this by modifying the systematic slip parameters.
- Sand Pit. This was the terrain where we ran the real world experiments with the Husky robot. In this case, we pretended that we were training in one region and carried over the systematic slip parameters to another region.

On all of the above mentioned scenarios, we tested the algorithm on a desired path that had twenty 90 degree turns (8 left and 12 right) spread out over a path 312.5 meters long. The WMR was commanded to traverse this entire path at at maximum possible speed of 5 m/s which also happened to be the maximum speed of the mapping platform.
The simulations were conducted in UTM coordinates and all paths are plotted relative to starting location of the WMR to reduce clutter. Also, the vehicle overlay on the plots is for reference only and not drawn to scale. Since the simulation framework provides ground truth pose information, a single run was enough to collect performance statistics.

Fig. 5 shows the comparative performance of RHMPPF and the pure pursuit tracker. The top row shows the entire traversed path for all three scenarios. The bottom row zooms into a smaller section to drive home the point.

5) Results: Table I summarizes the maximum and the mean error (m) for the pure pursuit and the RHMPPF algorithms. Colors red and green indicate a higher (worse) or a lower (better) cross track error among the competing algorithms within the same scenario/terrain.

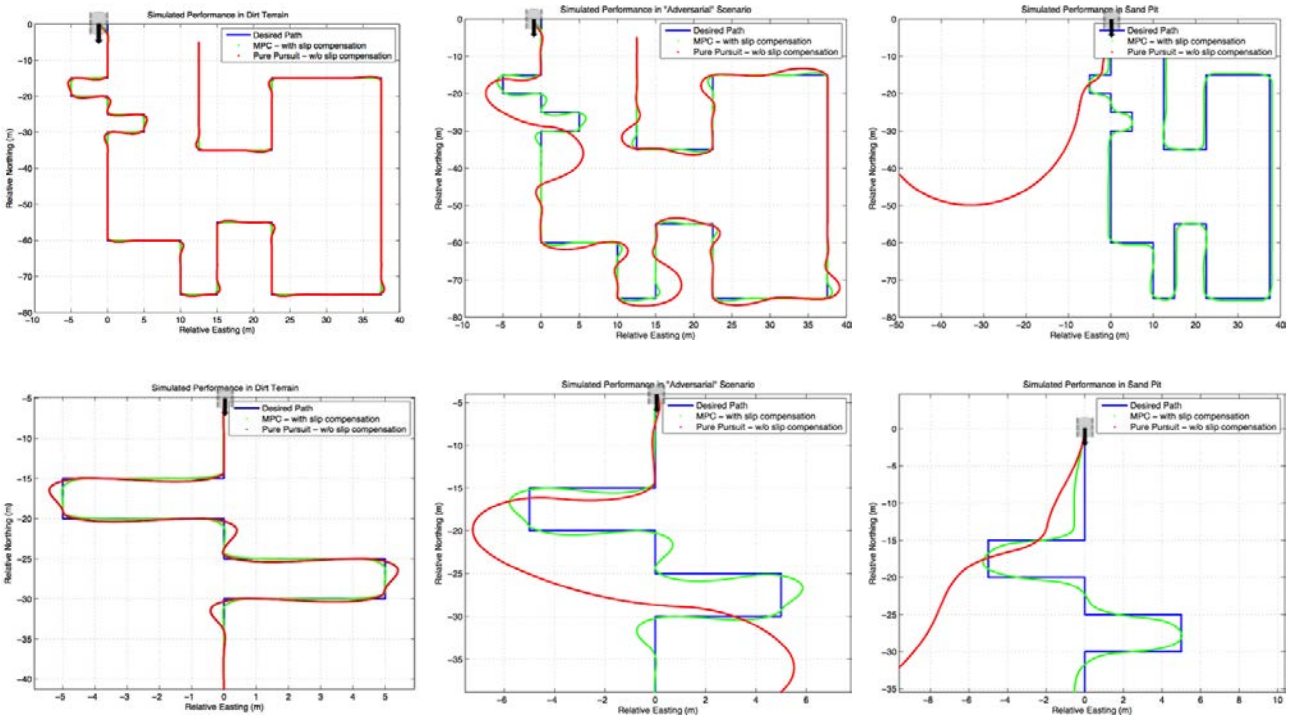

Fig. 5: From left to right: dirt terrain, adversarial scenario, and sandpit experiment routes. The top row shows the entire path and the bottom row shows a close up view. The desired path is in blue, the path traversed using the pure pursuit algorithm is in red while the one traversed using RHMPPF is in green.

TABLE I: Max/Mean Cross Track Error (m) in Simulation

	Dirt	Adversarial	Sand Pit
Pure Pursuit	0.41/0.08	5.03/0.54	50.00/N/A
RHMPPF	**0.29/0.02**	**0.73/0.14**	**0.38/0.14**

B. Real Robot Trials

In this section, we describe our efforts to validate the performance of the RHMPPF in the real world. A 4x4, all-wheel drive, skid-steered vehicle, manufactured by Clear Path Robotics called Husky retrofitted with wheel encoders and a pose system similar to the one on the Land Tamer served as the platform of choice for our field trials.The Husky measures 39 x 14 x 19.6 inches in its exterior dimensions and had a wheel diameter of 13 inches and 5 inches of ground clearance. We used the ROS based command and control API that was shipped with the platform. The RHMPPF was implemented as a ROS node and produced control commands at 20 Hz. The joystick controller was re-programmed to execute the commands generated by the RHMPPF on demand. Similar to the data collection efforts mentioned above, slip parameters of the terrain and the power train dynamics were learned by driving the Husky on the representative terrains. We evaluated the performance of the RHMPPF against a pure pursuit path tracker using cross track error as the guiding metric. In all our trials, we set the mission speed to 1.0 m/s, the top speed of the platform. We chose two terrains:

1) Sand Pit: We created a sand pit that was roughly 10 meters long, 4 meters wide, 18 inches deep and filled it with dry, fine river sand. We conducted a total of 10 trials each with the pure pursuit path tracker and RHMPPF and recorded the vehicle state information on each of those trials. We also laid small traffic cones (see Fig. 6d) in the sand pit in order to visually see the performance difference between the two approaches. Trials were conducted on a path slightly longer than the length of the sand pit with two 45 degree right turns and three 45 degree left turns placed approximately equidistant from each other along the path. The sand pit was raked after every trial so that subsequent ones started with similar initial conditions. Fig.6d shows the paths traversed by the pure pursuit path follower and RHMPPF during one such trial.

2) Trials in Dirt Terrain: We used the dirt terrain that was adjacent to the sand pit described above as the second terrain type for the real robot experiments. This terrain was roughly 15 meters wide and 25 meters long and consisted primarily of packed dirt and occasional loose rocks and tufts of grass. Trials were conducted on a path with eight 90 degree turns (4 left and 4 right) spaced about 4 to 5 meters apart. We performed a total of 10 trials each with the pure pursuit path tracker nad the RHMPPF and Fig. 6c shows the relative performance of the two approaches.

3) Results: Table II summarizes the maximum and the mean cross track error (m) for the pure pursuit and the RHMPPF algorithms. Colors red and green indicate a higher (worse) or a lower (better) cross track error among the competing algorithms within the same scenario/terrain. The numbers that are shown are the average over all the trials for a specific terrain type for each approach.

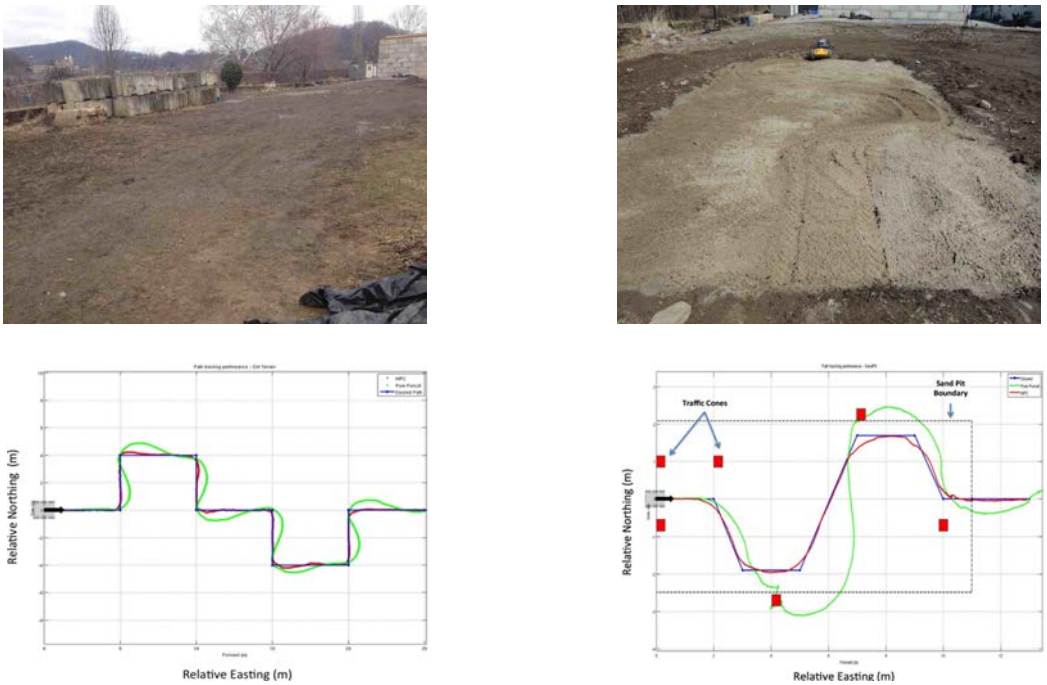

Fig. 6: Top: Terrains used in the real robot experiments; bottom: Paths traversed by the robot during the trials. Desired path in blue, RHMPPF paths in green and pure pursuit path in red.

TABLE II: Max/Mean Cross Track Error(m) in Field Trials

	Dirt	Sand Pit
Pure Pursuit	0.75/0.20	1.54/0.70
RHMPPF	**0.31/0.10**	**0.26/0.04**

V. CONCLUSIONS

We introduced the Receding Horizon Model Predictive Path Follower (RHMPPF) optimal control framework for improving path following performance on WMRs through predictive slip correction. Our framework shows a lot of promise especially in dealing with systematic errors which cannot be removed by feedback control alone. In our real robot experiments where we used a Husky skid steer mobile robot, we achieved six fold reduction in maximum cross track error and 17 fold reduction in mean cross track error in a sand pit, and 50% reduction in both mean and maximum cross track errors in the dirt terrain.

We have shown in high-fidelity data-driven simulation and on real robots that our framework adapts extremely well to terrain with varying wheel slip properties. Incorporating our slip aware prediction approach to path planning process for generating terrain-specific navigable paths and leveraging the potential correlation of visual appearance of a terrain patch and its slip characteristics to boost our slip model training process using past experiences can be listed among the things we would like to address in the future. We also hope to address the proof for stability and convergence of our approach along with techniques for learning the Q and the R matrices automatically in the near future.

ACKNOWLEDGMENT

This research was conducted at the National Robotics Engineering Center, Carnegie Mellon University under contract to the Army Research Office as a part of the Vehicle-Ground Model Identification program (Grant Number: W911NF-09-1-0557) and the Robotics Collaborative Technology Alliance program.

REFERENCES

[1] R.C. Coulter. Implementation of pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, 1992.

[2] Kelly, A., Nagy, B. "Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control", International Journal of Robotics Research, Vol (No) 22, 2003

[3] Howard, M. Thomas, Green, Collin and Kelly, Alonzo. "Receding Horizon Model-Predictive Control for Mobile Robot Navigation of Intricate Paths", In Proc. of FSR'09, July 2009

[4] A.Lacze,Y.Moscovitz,N.DeClaris,and K.Murphy. "Path Planning for Autonomous Vehicles Driving Over Rough Terrain". In Proc. of the ISIC/CIRA/ISAS'98, pages 5055, September 1998.

[5] Y. Kuwata, A. Fiore, J. Teo, E. Frazzoli, and J.P. How. "Motion Planning for Urban Driving using RRT". In Proc.of IROS'08, pages 16811686, September 2008.

[6] G. Ishigami, K. Nagatani, K. Yoshida, "Slope Traversal Controls for Planetary Exploration Rover in Sandy Terrain", Journal of Field Robotics, Vol. 26, Issue 3, pp. 264-286, March, 2009

[7] Seegmiller, N., Rogers-Marcovitz, F., Miller, G.A., and Kelly, A. "A Unified Perturbative Dynamics Approach to Online Vehicle Model Identification", ISRR, August, 2011

[8] Kalman, R. E., "Contributions to the Theory of Optimal Control", Bol. Soc. Mat. Mexicana, 1960, pp, 102-119.

[9] Johnson M.A., Grimble M.J. (1987). "Recent Trends in Linear Optimal Quadratic Multivariable Control System Design". IEE Proc. Contr. Theory and Appl., 134, 53-71.

[10] Helmick, D., Angelova, A. and Matthies, L. "Terrain Adaptive Navigation for Planetary Rovers". In Journal of Field Robotics , Issue 26(4), pp.391-410, 2009