2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Localization Uncertainty-driven Adaptive Framework for Controlling Ground Vehicle Robots

Daniel Kent[1], Philip K. McKinley[2] and Hayder Radha[1]
[1]Department of Electrical and Computer Engineering
[2]Department of Computer Science and Engineering
Michigan State University, East Lansing, Michigan, USA
(kentdan3@egr.msu.edu, mckinley@cse.msu.edu, radha@egr.msu.edu)

***Abstract*— Modern localization techniques allow ground vehicle robots to determine their position with centimeter-level accuracy under nominal conditions, enabling them to utilize fixed maps to navigate their environments. However, when localization measurements become unavailable, the position accuracy will drop and uncertainty will increase. While research and development on localization estimation seeks to reduce the severity of these outages, the question of what actions a robot should take under high localization uncertainty is still unresolved, and can vary on a platform-by-platform and mission-by-mission basis. In this paper, we exploit localization uncertainty measures to adapt system control parameters in real time. Offline, we optimize non-linear activation functions whose control parameters and relevant weights are trained and learned using Evolutionary Algorithm (EA). Subsequently, in real time, we apply the optimized adaptation functions to the controller look-ahead distance and intermediate linear and angular velocity commands, which we identify as the most sensitive to localization error. Evolutionary runs are conducted in which a simulated target vehicle is tasked with following a randomly generated path while minimizing cross-track error, with time varying localization uncertainty added. These runs produce situation-dependent weights for parameters to the adaptation functions, which are transferred to the physical platform, a 1:5-scale autonomous vehicle. In simulation, our system was able to reduce cross-track error, which in certain cases exceeds 250 centimeters on non-adapted systems, to below 15 centimeters on average using EA-derived weights and parameters applied to our proposed adaptation system. Evaluation on the physical platform demonstrates that without the adaptation module in place, the platform is unable to successfully follow the path; with the adaptation module, the platform automatically adjusts its velocity and look-ahead distance to compensate for localization uncertainty.**

***Index Terms* — autonomous vehicle, localization, path planning, uncertainty, evolutionary robotics, Robot Operating System**

I. Introduction

The capabilities of automated ground vehicle robots have advanced significantly over the past two decades, with many companies planning to release automated vehicles commercially in the near future. However, a question highly relevant to autonomous vehicles remains unanswered: What actions are appropriate for a vehicle to take when its localization uncertainty exceeds acceptable thresholds? For autonomous vehicles, centimeter-order uncertainty is required to safely utilize lane center information for path planning purposes [1]. However, the appropriate action under a given level of uncertainty will depend on the specific platform and environment.

Solutions to this problem attempt to maintain acceptable localization estimates when world-referenced positioning is reduced or unavailable. Localization results from Global Navigation Satellite System (GNSS) alone are generally not used for automated vehicle navigation, as commonly encountered environs such as tunnels and urban canyons can reduce the accuracy of or even outright block GNSS positioning capabilities. Inertial Navigation Systems (INS) fuse the global-frame localization results from GNSS with local odometry and sensor data, such as Inertial Measurement Unit (IMU) data, to produce a continuous localization estimate through the use of Kalman filters [2], factor graphs [3], or other techniques. Since local sensor and odometry data is subject to process and sensor noise, causing the estimate to lose accuracy over time, typically these errors are periodically corrected with global reference data such as that from GNSS. However, if GNSS is denied or otherwise unavailable, the localization errors will continue to grow unbounded. If localization uncertainty exceeds a certain point, successfully following paths defined in the global reference frame becomes difficult and potentially hazardous, if not impossible. While much research has focused on reducing localization uncertainty and error, and a limited body of work addresses localization uncertainty on platforms that are permitted to explore their environments [4], there is limited research on adapting to localization uncertainty into path-following platforms with limited capacity for error.

In this paper, we investigate the application of evolutionary search to determine how robot platforms should respond to high localization uncertainty, as well as whether a non-linear activation function can effectively adapt the system in such situations. In the process, we determine what adaptations, if any, reduce operational error during periods of high localization uncertainty, and at what level(s) of uncertainty each adaptation activates. The target platform for this study, shown in Figure 1, is a 1:5-scale vehicle based on an open platform developed for the study of autonomous driving [5]. Evolutionary runs conducted using a simulation model of this vehicle revealed sets of adaptation parameters that each performed better under intermittent uncertainty, than a non-adaptive system; indeed, in some cases the non-adaptive sys-

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

tem was unable to perform path following at all. Experiments on the physical platform confirmed these results under actual changes in localization uncertainty.

Fig. 1. MSU EvoRally, a 1:5-scale vehicle constructed at Michigan State University, based on AutoRally, an open platform developed by researchers at Georgia Tech [5].

This work contributes two specific ideas: (1) the use of non-linear adaptation functions as specific implementations for localization uncertainty that can be applied to system inputs, parameters, and outputs in real time, and (2) the use of EAs to evaluate these adaptations for applicability and determine their optimal values. The adaptation equations can be applied to a variety of different platforms and platform components, and the EA scales well to an increasing number of these adaptation parameters.

The remainder of this paper is organized as follows. Section II provides background on the localization problem and evolutionary robotics. Section III describes the experimental setup, including details of the MSU EvoRally platform, the evolutionary search framework, the localization uncertainty model, and the specific parameters exposed to evolutionary pressures. Section IV presents the results of the evolution runs, and Section V describes validation on the physical vehicle. Finally, Section VI draws conclusions and discusses possible future directions.

II. Background and Related Work

We begin by providing background on the localization problem and evolutionary search as applied to robotics.

A. Localization in Ground Vehicle Robots

Precision localization is an important function for ground vehicle robots; with centimeter-level accuracy, ground vehicle robots can navigate previously mapped lanes on roadways [1], which simplifies lane keeping down to a path following problem. Aqel et al. [6] point out that while several techniques fulfill the accuracy requirement, each has trade-offs. For example, Real-Time Kinematic (RTK) GNSS and other differential GNSS techniques can reduce standard satellite navigation uncertainty to centimeter or millimeter range, but require an unobstructed view of the sky. Inertial Navigation Systems (INSs) can provide short-term dead reckoning estimates using accelerometers in the absence of other data, but are prone to drift over long periods of time. Lidar can report precise distances to surrounding objects, which can be used for lidar map-based localization [7] or for odometry generation through Simultaneous Localization and Mapping (SLAM) [8]. However, these methods may not work well for areas with few vertical features [9]. Cameras can be used to generate odometry information, but substantial compute power is needed to process the images, and this approach is sensitive to environmental conditions such as weather. Moreover, monocular camera odometry could contain a non-static *scale uncertainty* that can cause errors in the visual odometry estimate [10]. Even if multiple techniques are fused, localization errors can arise if one or more technique is subject to high uncertainty, or if a technique becomes unavailable due to equipment failure.

Given the potential vulnerabilities of all localization systems, it is important that ground vehicle robots have the ability to detect increases in localization uncertainty and respond appropriately. While many techniques are able to report uncertainty, the appropriate response will vary by platform, application, and mission. For instance, probabilistic approaches have been shown to be effective in mapping uncertain environments, but primarily target robotic platforms that are permitted to explore their environments, such as interplanetary rovers [4]. In this paper, we explore the potential role of evolutionary algorithms (EAs) as a means to find parameter values for adapting to localization uncertainty on platforms required to follow specific paths with high precision.

B. Evolutionary Robotics

Modern robots are highly complex; a small change to a parameter of a single subsystem might produce significant changes in performance. For example, a small adjustment of gain in a controller on a robot could cause the entire system to exhibit undesirable behavior, such as oscillatory motion around the desired control signal. The optimal value for this gain could in turn be influenced by other factors in other subsystems. Tuning a robot's parameters can be a tedious, manual task, which has driven research on alternative ways to find optimal, or at least suitable, sets of parameters.

Researchers in the field of evolutionary robotics (ER) [11] have sought to *harness* the search capabilities of EAs in designing the control and morphology of robots. By operating in an open-ended manner, unconstrained by human preconception and bias, these algorithms can find unconventional solutions to problems as well as reveal situations that might cause the system to fail [12], [13]. In order to avoid damage to physical robots, evolution is usually conducted in simulation. A typical approach uses a genetic algorithm (GA) to optimize characteristics (e.g., controller parameters, sensor configurations) of the robot. Each individual in a population represents a possible software/hardware configuration of the target platform, whose performance is evaluated with respect to a fitness function. Those individuals with high fitness are selected to pass genes to the next generation. Over generations, performance of the task improves and eventually

plateaus, yielding one or more potential solutions, which can then be tested in physical robots. Differences between simulated performance and actual performance, referred to as the *reality gap* [14]–[17], are analyzed and used to modify the simulation models. The process is repeated until a satisfactory solution is produced.

EAs are not the only optimization method that can be used to optimize parameters for robotics applications. In addition to classical methods, machine learning-based techniques such as gradient descent and reinforcement learning (RL) have been used to develop and optimize control systems. However, gradient descent-based algorithms are difficult to parallelize [18], and RL techniques typically use gradient descent to optimize parameters, whereas EAs are much more conducive to decentralized evaluation. While RL can incorporate EAs to perform parameter search [19], we opted to use EAs directly to discover optimal parameters.

C. Evo-ROS

Evolutionary approaches have yielded effective designs for terrestrial, aquatic, and aerial robots [11]. However, ER robots and their environments tend to be relatively simple, due in part to the high computational cost of evolutionary runs. Moreover, using simplified models increases the reality gap. Recently, Silva et al. [20] addressed this issue and pointed out the benefits of using tools from the mainstream robotics community in ER simulations. For example, an increasing number of robotic systems and autonomous vehicles utilize software infrastructures based on the Robot Operating System (ROS) [21]; an advantage of ROS is that code developed and tested for a simulated robot can be deployed and executed directly on the corresponding physical robot.

Simon et al. [22] recently developed Evo-ROS, which extends evolutionary search to robots whose software infrastructure is based on ROS. Evo-ROS currently uses the Gazebo physics-based simulator, which is often coupled with ROS and provides tested models of many commercially-available hardware components, although other simulators could be used. Evo-ROS has previously been applied to increase robustness of ground vehicle robots by optimizing sensor placement and redundancy in the presence of component failure and damage [22]. In addition to optimizing physical characteristics of the platform, Evo-ROS can be used to optimize software components and parameters. For example, Langford et al. [23] combined Evo-ROS with novelty search [24] in order to optimize the throttle controller on EvoRally, the target platform of the current study.

III. Evolutionary Process and Design

Our experiments were conducted in two stages. The first stage constituted Evo-ROS runs with the simulated robot in order to find weights for adaptation parameters based on simulated localization uncertainty. In the second stage, we tested those parameters on a physical robot platform to determine how these adaptation parameters affect performance and whether the evolved adaptation produced a system that was more robust to localization uncertainty.

A. Ground Vehicle Platform

Although the proposed approach is applicable to many autonomous vehicles, the demonstration system in this study is the MSU EvoRally vehicle shown in Figure 1. EvoRally is based on the open-source AutoRally platform developed at Georgia Tech; Goldfain et al. have published a comprehensive description of their system's hardware and software [5]. The vehicle's chassis is borrowed from a gas-powered 1:5 scale remote-controlled truck; the engine is replaced with an electric motor, and many mechanical components are replaced with stronger versions to accommodate the weight of computing equipment and sensors. A custom compute box houses a quad-core processor, 32GB RAM, 2TB SSD, and a GPU for real-time image processing. Sensors include a high-precision IMU, RTK GNSS, Hall-effect rotation sensor on each wheel, and two front-facing machine vision cameras; the completed vehicle weighs 46 lbs and has a top speed of 60 mph. Our platform was built with similar hardware specifications as the one described in [5], with changes made to attach and integrate a different GNSS receiver, among other modifications. For the Evo-ROS simulation of the vehicle, we reused and expanded simulation code developed by the Georgia Tech group.

As noted above, the vehicle's software infrastructure is based on ROS, where individual functions are realized as separate executables, enabling us to easily exchange components such as controllers with off-the-shelf or experimental software. For this study, we replaced the default ROS navigation stack and implemented the platform's path following software as a two stage controller. The first controller stage utilizes a Pure-Pursuit based algorithm implemented previously by the Autoware project [25]. The algorithm takes an input path and converts it into a desired linear and angular velocity, or *twist*, based on a point on the path chosen in front of the platform; the distance to this point is known as the *look-ahead distance*, and the chosen point the *look-ahead point*. In the second stage, this *twist* is converted into platform-specific throttle, steering, and braking commands using a proportional-integral-derivative (PID) controller. The traditional Pure Pursuit algorithm computes the look-ahead point deterministically using either a fixed look-ahead distance, or a simple varying look-ahead distance that depends on a measure such as current vehicle speed [26]. Similarly, even though the twist command output from the Pure Pursuit algorithm is derived from the vehicle's current position and is implicitly sensitive to localization error, traditional PID controllers do not incorporate localization uncertainty as a parameter used to calculate platform-specific commands.

While PID controllers may not be as robust for path following as other types of controllers, such as Model Predictive Control (MPC) controllers, our PID-based controller is adaptable to different platforms, and only requires modification of the specific output commands and tuning the gains of the PID controller. In addition, MPC-based controllers for

ground vehicles incorporate similar parameters such as look-ahead distance [27] that could be adapted in real time using our proposed method and optimized using EAs. While this two-stage control method works well when the measured and/or estimated position is accurate, inaccurate localization data can result in poor path following performance, potentially putting at risk the vehicle, its surroundings, or bystanders.

B. Adaptation, Evolution, and Fitness Parameters

To address localization uncertainty, we first identified three specific parameters in our system that were the most sensitive to localization uncertainty: the configured look-ahead distance within the pure pursuit module, as well as the output linear and angular velocity commands. We developed a novel uncertainty adaptation module that could adjust the look-ahead distance and velocity commands in real time based on the estimated uncertainty into the two-stage control algorithm explained above. As shown in Figure 2, the uncertainty adaptation module reads in the current localization uncertainty, and uses it to adjust system parameters in real time. Without the uncertainty adaptation module, the direct output of the pure pursuit module is used as the input of the twist controller, with the look-ahead distance remaining fixed. With the module in place, the localization uncertainty as computed by the localization subsystem is used to adjust the look-ahead distance and reduce the commanded linear and angular velocities as localization uncertainty increases.

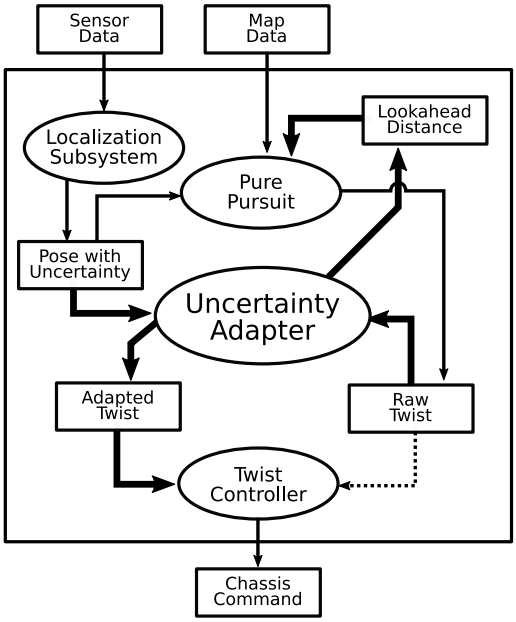

Fig. 2. Software flow diagram showing Uncertainty Adapter node adjusting data and parameters across multiple controller subsystems. Thick lines indicate flows that are active when the Uncertainty Adapter module is enabled, and dashed lines indicate flows that are active when the Uncertainty Adapter module is disabled.

The specific equations we developed for adapting system parameters are based on a Rectified Linear Unit (ReLU) activation function, and was partially inspired by formulations of non-linear activation in the field of machine learning [28][29]. The model function appears in Equation 1. The function takes in a nominal parameter value, x, and an uncertainty measure, σ_x, and outputs an adjusted parameter value, $\hat{x}$. This function in its basic form has two parameters that are fixed: the weight, or slope, of the activation function, W_x; and the offset, or minimum activation level, O_x. The equation can also be clamped between certain values if necessary, shown as x_{min} and x_{max}. Equation 1 can be modified for multiple types of parameters and data on a variety of robot platforms.

$$\hat{x} = clamp_{\{x_{min},x_{max}\}}(x \times (1 - W_x) \times relu(\sigma_x - O_x)) \quad (1)$$

Given our identification earlier of the three target parameters (look-ahead distance L, output linear velocity command V_ℓ, and output angular velocity command V_θ), we implemented three equations, which appear as Equations 2, 3, and 4. Each of the twist adaptation equations require the weight and offset parameters to be determined. The look-ahead equation by contrast has five parameters; in addition to the weight and offset parameters, the minimum, maximum, and base look-ahead parameters must be determined. Importantly, the weight for the look-ahead is scaled such that it can take positive or negative values, and the EA is restricted to choosing base, minimum, and maximum look-ahead distances between 0.5 and 10 meters. Combined, the three equations yield nine parameters (a.k.a. *genes* that are subject to evolution using the EA; these are listed in Table I.

$$\hat{L} = clamp_{\{L_{min},L_{max}\}}(L \times (1 - W_l) \times relu(||\Sigma||_F - O_l)) \quad (2)$$

$$\hat{V_\ell} = clamp_{\{0,V_\ell\}}(V_\ell \times (1 - W_{V_\ell}) \times relu(||\Sigma||_F - O_{V_\ell})) \quad (3)$$

$$\hat{V_\theta} = clamp_{\{0,V_\theta\}}(V_\theta \times (1 - W_{V_\theta}) \times relu(||\Sigma||_F - O_{V_\theta})) \quad (4)$$

Our objective is to employ a measure of the localization uncertainty to adapt the values of the linear velocity, angular velocity, and look-ahead distance in real time. Here, we employed the Frobenius norm $||\Sigma||_F$ of the localization covariance matrix Σ as a measure of uncertainty that guides our adaptation framework. It is possible that other, potentially more-complex, measures of uncertainty might be more effective in this optimization problem. This particular aspect of our work is left for further investigation.

TABLE I
ADAPTATION PARAMETERS (GENES) FOR EVOLUTION

	Equation	Minimum	Maximum
W_x	1	0	10
W_θ	2	0	10
O_x	1	0	1
O_θ	2	0	1
L_b	3	L_{min}	L_{max}
L_{min}	3	0.5	L_{max}
L_{max}	3	L_{min}	10
W_L	3	-5	5
O_L	3	0	1

The fitness function we used for our EA takes the average cross-track error, e_{avg}, and distance along the path d the adapted system is able to travel and calculates a fitness using Function 5

$$f(d, e_{avg}) = d \times \frac{npdf(e_{avg}, 0.25)}{npdf(0, 0.25)} \quad (5)$$

where $npdf(a, \sigma)$ is the value of the probability distribution function (pdf) for a zero-centered Gaussian pdf with standard deviation σ evaluated at a. We used the Gaussian pdf to calculate the fitness components from the cross-track error in order to ensure the function was smooth and to minimize the penalty for small amounts of cross-track error. The EA was configured with a population size of 50 evolved over the course of 25 generations. Genes were randomly uniformly initialized subject to a 7.5% mutation rate and a 5% crossover rate. We executed the EA 5 times with different random number generator seeds in order to capture potential variation in optimal sets of parameters.

C. Paths

A simulated path is generated as a semi-random set of straight paths and curves, each segment of which has equal length. Segments are allowed to turn left if the previous segment is heading in the positive X direction or the negative Y direction, or right if the previous segment is heading in the positive X direction or the positive Y direction. These turning restrictions prevent the path from turning in on itself and guarantees that the robot cannot skip segments without incurring a significant cross-track error, which can be detected by an evaluation module. For this initial study, we kept the path fixed; this path is shown in Figure 3. Future work will vary the path once per EA run, and later may vary per generation to further challenge population members.

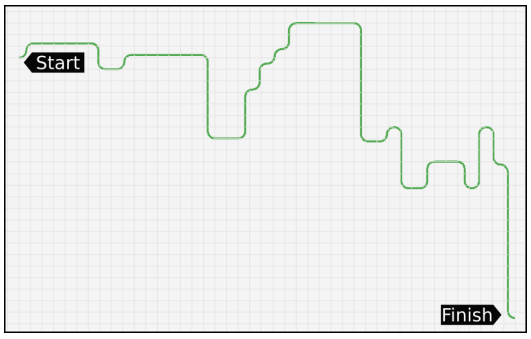

Fig. 3. A specific path used in simulation during EA, with start and end points marked. Grid size is 10 meters square.

D. Simulating Odometry and Uncertainty

To reduce the computational complexity of the simulation, we developed a simplified approximation of an INS that uses the simulated robot's ground truth position and orientation, and adds a time-dependent amount of Gaussian noise to the ground truth X and Y positions and the platform orientation.

TABLE II
TABLE OF RESULTS

Run	Fitness	Distance	Avg. Cross-Track Error
1	**71.992**	**112.5m**	0.152m
2	60.992	85.25m	**0.141m**
3	62.635	112.25m	0.196m
4	71.604	88.2m	0.220m
5	69.367	80.2m	0.266m
Baseline	4.641	14.8m	0.380m

The noisy position is then reported to the simulated platform along with an uncertainty estimate, represented as a 3x3 Gaussian covariance matrix, based on the amount of time-dependent noise added to the simulation. We model uncertainty this way, as it is possible to obtain similar types of uncertainty estimates with a Kalman-based filter, such as an open-source software implementation developed by Moore and Stouch for ROS [30]. While localization uncertainty may not necessarily be Gaussian outside of simulations, the performance of Gaussian-based localization filters is sufficient to achieve centimeter-level accuracy using low-cost sensors [31]. Future work may involve more challenging assumptions about the noise model for localization uncertainty.

E. Simulation Monitoring

During the evaluation of each population member in the EA, the simulation monitors and records average cross-track error and distance for fitness evaluation. To reduce the time spent simulating invalid or unsuitable genomes, the maximum cross-track error is monitored, and if it exceeds 2.5 meters, the simulation ends. Once the the simulation reaches 240 simulated seconds, the simulation ends.

IV. Results of Evolutionary Search

A. Path Following Performance

The evolutionary runs discovered several sets of parameters that were able to successfully adapt the system to periods of high localization uncertainty. As shown in Table II, the top genome, which came from run 1, was able to follow 112.5 meters of the path in 240 simulated seconds, with an average cross-track error of 0.152 meters. By contrast, a simulated non-adapted system with fixed look-ahead distance of 2.65m and deterministic forward and angular commands was unable to follow the path for the full 240 seconds, instead exceeding the 250cm threshold during a period of high localization error. The specific parameters varied between different genomes; some genomes chose a positive weight for the W_L parameter, while others chose a negative weight, corresponding to an increase and decrease in look-ahead distance, respectively, as localization uncertainty increases. Figure 4 shows an example of the improvement in path following behavior. Figure 5 reveals that the EA-derived parameters causes the system to reduce the output forward and angular velocity commands when uncertainty increases.

Early development of our adaptation functions used strictly linear equations, which resulted in reductions in forward and angular velocity commands even when uncertainty was

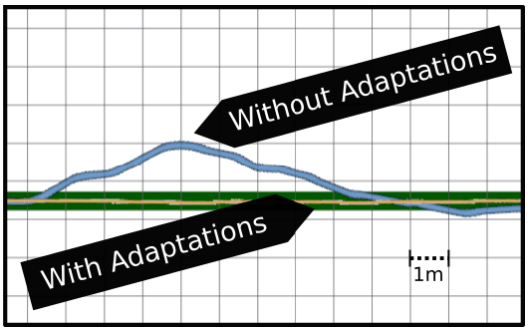

Fig. 4. Example of path-following performance of non-adapted (blue) and adapted controller in simulation, using results from run 1. Grid size is 1 meter square.

low. The introduction of the non-linear ReLU function as the basis for our adaptation functions eliminated the steady-state forward and linear velocity errors. In addition, the offset values determined by the EA serve as a threshold that can measure how sensitive each parameter is to localization error.

B. Evaluation of Fitness Function Effectiveness

Figure 6 plots the average and maximum fitness for one of the evolutionary runs, demonstrating the objective improvement of the fitness. Based on this result as well those in Table II., we conclude that the fitness function effectively guided the EA toward a suitable set of adaptation parameters that reduced the platform's average cross-track error as the uncertainty increased and decreased throughout the simulations. Despite the fitness function not factoring in maximum cross-track error, the adapted systems' maximum cross-track error stayed well below the 250 centimeter threshold. EAs that incorporate multi-objective optimization could be explored in the future to determine if a trade-off exists between average and maximum cross-track error for our application.

V. Validation on Physical Robot

To validate that a controller with evolved adaptation parameters performed adequately on a physical robot, we tested path following on the EvoRally platform at the MSU Spartan Mobility Village, a 330-acre space with natural areas as well as winding and intersecting streets. The vehicle was configured to utilize a graph-based localization filter as an input to the controller to smooth localization results. The GNSS-derived position covariance matrix was used as the input to the adaptation module, with differential GNSS measurements obtained in real time from the Michigan Department of Transportation's Continuously Operating Reference Station (CORS) network. A course was defined using a series of GNSS waypoints that roughly formed a square in a parking lot, as shown in Figure 7. The location of each waypoint was surveyed with RTK GNSS using an RTK fix solution for accuracy; tape was laid out on the path for our benefit, but was not used by EvoRally's cameras for path following.

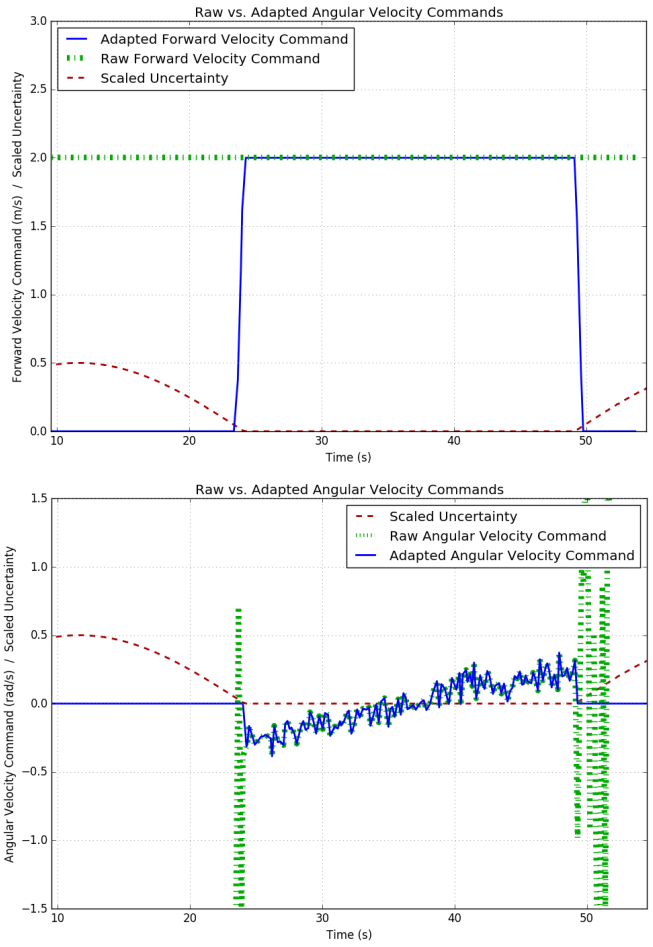

Fig. 5. Forward velocity command profile (top) and steering command (bottom) of adapted controller under uncertainty. Large shifts in angular velocity command occurred as simulated GNSS position became highly noisy.

During testing, the GNSS solution provided by the platform's GNSS receiver fluctuated between RTK fix, RTK float, and Single-Point Positioning (SPP). RTK Fix has the smallest covariance, and SPP has the largest. These conditions provided us an opportunity to evaluate the robot's performance under dynamic localization conditions, without the need to introduce artificial noise.

When the adaptation module was not activated and the GNSS solution was other than RTK fix, the vehicle deviated significantly from the path, often requiring us to perform a manual takeover of the vehicle in order to avoid an accident. When the adaptation module was activated, the platform automatically stopped if the localization solution was SPP, would drive at full or near-full speed with a short look-ahead distance when the solution was RTK fix, and would drive at slower speeds with a longer look-ahead distance when the solution was RTK float, as shown in Figure 8. In summary, our localization adaptation module was able to successfully adapt to actual, changing GNSS fix qualities, without our having explicitly programmed this behavior in

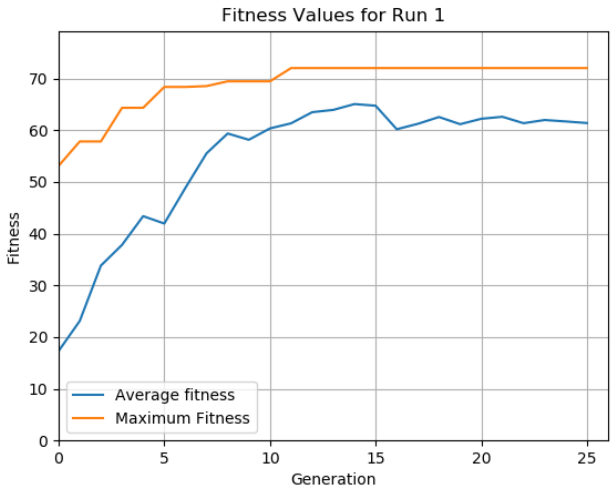

Fig. 6. Average and Maximum fitness scores achieved over the course of the EA for Run 1.

Fig. 7. Photo of course with approximate positions of GNSS waypoints marked in white tape.

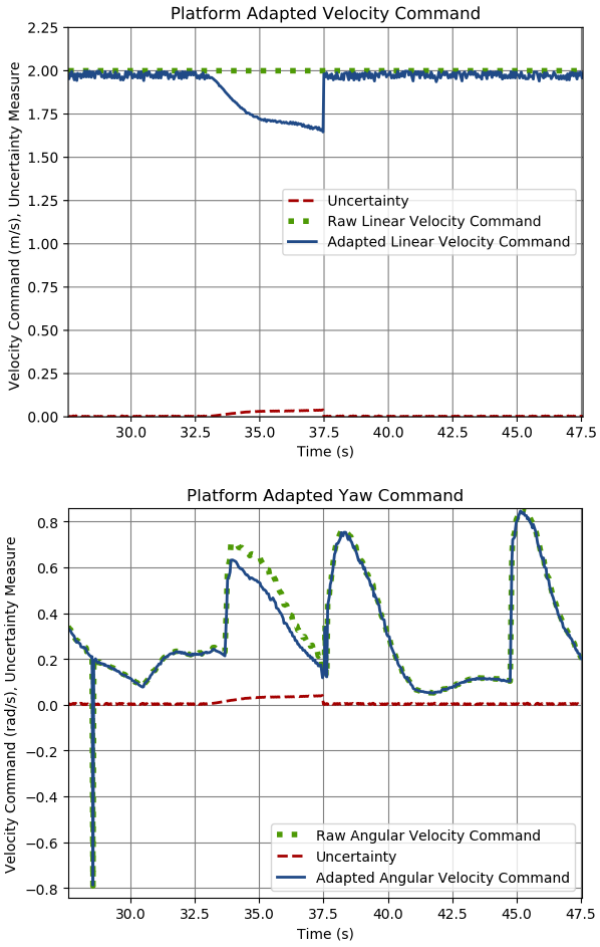

Fig. 8. Top: Logged commanded and adapted forward velocity on the platform during testing. Bottom: Logged commanded and adapted angular velocity on the platform during testing. Note the increase in uncertainty, which occurred due to GNSS solution changing from RTK fix to RTK float.

response to changes in GNSS fix type. Despite the difference in the simulated versus actual paths, the adaptation module performed well, which suggests that the *reality gap* may be small.

VI. Conclusions and Future Directions

In this paper, we investigated the application of evolutionary search to the problem of adaptation in the presence of localization uncertainty, developed a set of non-linear adaptation equations to modify system parameters and outputs in response to localization uncertainty, and optimized the parameters using an EA. These techniques improved the path following performance of a simulated ground vehicle robot and were successfully applied to the corresponding physical platform, the MSU EvoRally vehicle. The use of non-linear activation functions to adapt robots for localization uncertainty, as well as the use of an EA to evaluate and adapt robot platforms for localization uncertainty, may have application to other robot platforms that rely on highly accurate localization results for path planning purposes.

This work can be extended in several ways. First, since GNSS uncertainty is often location-dependent, with higher lateral uncertainties occurring in natural or urban canyons, we plan to integrate into our simulator more realistic models of GNSS noise as well as location-dependent noise. Second, we plan to combine Evo-ROS with novelty search, as in [23], to create challenging scenarios that drive evolution. Such an approach would generate a series of tests that vary the path and the time-based and location-based uncertainty; evolving against these scenarios would likely produce a more robust solution. Parameters chosen might include specific path sequences as well as points on the map where localization uncertainty is increased. Third, since localization uncertainty affects obstacle avoidance, future testing will include obstacles and additional adaptation parameters related to obstacle avoidance. Finally, we also plan to evolve parameters for other types of ground vehicle robot platforms such as full-scale automated vehicles.

Acknowledgements

This work has been supported in part by the U.S. National Science Foundation under grant DBI-0939454, by the U.S. Air Force Research Laboratory under agreements

FA8750-16-2-0284 and FA8750-19-2-0002, and by the MSU Foundation Strategic Partnership Grants (SPG) program. We would like to thank Brian Goldfain of the Georgia Institute of Technology for the design and documentation of the AutoRally platform, and for providing technical assistance with hardware and software used in this research. We also thank Glen Simon, Jared Clark and Jonathon Fleck for the development of Evo-ROS and the construction of the MSU EvoRally platform. We thank Steven Yik, who provided assistance in fabricating parts for the robot; without his timely assistance this paper would not have been possible.

REFERENCES

[1] M. Schreiber, C. Knöppel, and U. Franke, "Laneloc: Lane marking based localization using highly accurate maps," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 449–454.

[2] P. Goel, S. I. Roumeliotis, and G. S. Sukhatme, "Robust localization using relative and absolute position estimates," in *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, vol. 2. IEEE, 1999, pp. 1134–1140.

[3] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[4] C. Olson, "Probabilistic self-localization for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 1, pp. 55–66, 2 2000.

[5] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg, "Autorally: An open platform for aggressive autonomous driving," *IEEE Control Systems Magazine*, vol. 39, no. 1, pp. 26–55, 2019.

[6] M. O. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," *SpringerPlus*, vol. 5, no. 1, p. 1897, 2016.

[7] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 4312–4319.

[8] H. Alismail, L. D. Baker, and B. Browning, "Continuous trajectory estimation for 3d slam from actuated lidar," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6096–6101.

[9] S. Pang, D. Kent, X. Cai, H. Al-Qassab, D. Morris, and H. Radha, "3d scan registration based localization for autonomous vehicles-a comparison of ndt and icp under realistic conditions," in *Proceedings of the 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp. 1–5.

[10] B. M. Kitt, J. Rehder, A. D. Chambers, M. Schonbein, H. Lategahn, and S. Singh, "Monocular visual odometry using a planar road model to solve scale ambiguity," in *Proceedings of European Conference on Mobile Robots*, 2011.

[11] S. Nolfi, J. Bongard, P. Husbands, and D. Floreano, "Evolutionary robotics," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2016.

[12] "Awards for human-competitive results produced by genetic and evolutionary computation," competition held as part of the annual Genetic and Evolutionary Computation Conference (GECCO), sponsored by ACM SIGEVO. Results available at http://www.human-competitive.org.

[13] A. J. Ramirez, A. C. Jensen, B. H. C. Cheng, and D. B. Knoester, "Automatically exploring how uncertainty impacts the behavior of dynamically adaptive systems," in *Proceedings of the 26th International Conference on Automated Software Engineering*, Lawrence, Kansas, November 2011, pp. 568–571.

[14] R. A. Brooks, "Artificial life and real robots," in *Proceedings of the First European Conference on Artificial Life*. MIT Press, Cambridge, MA, 1992, pp. 3–10.

[15] N. Jakobi, "Running across the reality gap: Octopod locomotion evolved in a minimal simulation," in *Proceedings of the First European Workshop on Evolutionary Robotics*. Paris, France: Springer-Verlag, 1998, pp. 39–58.

[16] J. C. Bongard and H. Lipson, "Once more unto the breach: Co-evolving a robot and its simulator," in *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, Boston, Massachusetts, USA, 2004, pp. 57–62.

[17] S. Koos, J. B. Mouret, and S. Doncieux, "Crossing the reality gap in evolutionary robotics by promoting transferable controllers," in *Proceedings of the 2010 ACM Genetic and Evolutionary Computation Conference*. Portland, Oregon, USA: ACM, 2010, pp. 119–126.

[18] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603.

[19] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette, "Evolutionary algorithms for reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 11, pp. 241–276, 1999.

[20] F. Silva, M. Duarte, L. Correia, S. M. Oliveira, and . A. L. Christensen, "Open issues in evolutionary robotics," *Evolutionary Computation*, vol. 24, no. 2, pp. 205–236, 2016.

[21] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *IEEE ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, Kobe, Japan, 2009.

[22] G. A. Simon, A. J. Clark, J. M. Moore, and P. K. McKinley, "Evo-ROS: Integrating evolution and the Robot Operating System," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion (Workshop on Evolutionary Computation Software Systems)*, Kyoto, Japan, July 2018, pp. 1386–1393.

[23] M. A. Langford, G. A. Simon, P. K. McKinley, and B. H. C. Cheng, "Applying evolution and novelty search to enhance the resilience of autonomous systems," in *Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Montreal, Quebec, Canada, May 2019.

[24] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI)*. Cambridge, MA, USA: MIT Press, 2008.

[25] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[26] S. F. Campbell, "Steering control of an autonomous ground vehicle with application to the DARPA urban challenge," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[27] G. V. Raffo, G. K. Gomes, J. E. Normey-Rico, C. R. Kelber, and L. B. Becker, "A predictive controller for autonomous vehicle path tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 92–102, 2009.

[28] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 2146–2153.

[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[30] T. Moore and D. Stouch, "A generalized extended Kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[31] Y. Zhuang, Q. Wang, M. Shi, P. Cao, L. Qi, and J. Yang, "Low-power centimeter-level localization for indoor mobile robots based on ensemble Kalman smoother using received signal strength," *IEEE Internet of Things Journal*, 2019.