Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC/CIE 2018
August 26-29, 2018, Quebec City, Quebec, Canada

DETC2018-85641

GAUSSIAN KERNEL CONTROLLER FOR PATH TRACKING IN MOBILE ROBOTS

Bijo Sebastian
Robotics and Mechatronics Lab
Mechanical Engineering Dept.
Virginia Tech
Blacksburg, VA 24061
bijo7@vt.edu

Adam Williams
Robotics and Mechatronics Lab
Mechanical Engineering Dept.
Virginia Tech
Blacksburg, VA 24061
aw13@vt.edu

Pinhas Ben-Tzvi
Robotics and Mechatronics Lab
Mechanical Engineering Dept.
Virginia Tech
Blacksburg, VA 24061
bentzvi@vt.edu

ABSTRACT
This paper describes the design of a Gaussian kernel based path tracking controller for mobile robots. In order to achieve successful navigation under hybrid navigation architectures, it is critical for the robot to follow the path provided by a high-level planner even while moving between waypoints. This is particularly difficult in real life situations involving robot motion in challenging terrains. Existing controllers for this purpose such as the pure pursuit does not ensure smooth motion of the robot or accurate tracking while moving between path segments. This paper describes the design of a controller that can ensure accurate path tracking even in the presence of disturbances, by utilizing the gradients of moving Gaussian kernels. In order to characterize the performance of the proposed controller, two different sets of simulations are conducted. Based on the results of the simulations, the Gaussian kernel controller ensures accurate tracking of the provided reference path while addressing the shortcomings of existing controllers. The paper concludes with a discussion on future directions for improvement.

NOMENCLATURE

∂_l	Look ahead distance (m)
G_r	Radius of the goal region (m)
N	Number of Path segments
μ	Center of the Gaussian
σ	Standard deviation for the Gaussian
G	Gradient of the Gaussian
F	Normalized 2D circular Gaussian
V	Desired linear velocity of the robot (m/s)
$V_{\max}$	Maximum linear velocity of the robot (m/s)
ω	Desired angular velocity of the robot (rad/s)
K_p	Proportional gain
L	Width of the Differential drive robot (m)
D	Diameter of the wheels of differential drive robot (m)
ω_l	Angular velocity of the left wheel of the differential (rad/s)
ω_r	Angular velocity of the right wheel of the differential (rad/s)

1 INTRODUCTION
One of the most popular topics in robotics and control is the exploration of various modes of autonomous navigation of mobile robots. An effective strategy is the implementation of hybrid navigation architecture, consisting of a deliberate high-level planner followed by a reactive low level controller [1, 2]. The high-level planner takes into account the map of the environment and develops a path to guide the robot to the goal. The path is generally output as a set of waypoints leading to the goal. The main responsibility of the reactive controller is to drive the robot through these waypoints. In addition, the reactive controller is also responsible for handling sudden changes in the environment that may require the robot to deviate from the path, or even execute an immediate stop. Such behavior can be triggered by moving obstacles or even by disturbances such as the slope of the terrain, slip or actuator limitations of the robot that the planner failed to consider. In order to handle these varied responsibilities, the reactive

controller is often implemented as a state machine or hybrid automaton [3]. These states consist of at minimum a "Go-to-Goal" behavior as well as an "Avoid Obstacle" behavior. One major advantage of such hybrid implementation architectures are that they enable the computationally expensive planner to be run at a very low frequency as compared to the reactive controller, while still providing accurate navigational capabilities. However, a negative aspect of this architecture is that planners often can only guarantee a feasible or optimal path provided that the robot follows the path accurately, including while it is moving between supplied waypoints. If the robot deviates from the proposed path while executing the motion under the action of the reactive controller, this can result in the path becoming suboptimal or in putting the robot in a position where it is unable to reach the goal. For instance, when the robot deviates from the path there is the possibility the robot may encounter already planned- for obstacles. In order to overcome this challenge, existing works have tried to incorporate more accurate models of the environment and the robot's motion, taking into account the kinematic constraints of the robot as well as the dynamics of robot terrain interactions, allowing the planner to predict a priori possible deviations that the robot might take [4–6]. In spite of the best efforts of researchers, the robots still often deviate from the proposed path in real life applications, especially when traversing challenging terrains. Under such conditions it becomes necessary to have a robust path tracking controller within the "Go-to- Goal" behavior that will drive the robot back onto the proposed path, rather than just driving the robot towards the next waypoint. This guarantees that the optimal path, already having been computed, will be followed while still maintaining the hybrid navigation architecture.

In addition, for autonomous platforms that are designed to operate alongside human beings, such as autonomous wheel chairs, inventory handling robots, and payload-transporting drones, motion produced by the path-tracking controller must be smooth and continuous. In other words, the controller should limit the velocity, acceleration and even jerk of the system while producing visibly safe, comfortable and intuitive motion of the robot to be considered [7, 8]. In this paper we propose a path tracking controller that produces smooth motions while meeting the aforementioned requirements.

2 LITERATURE REVIEW

Path tracking for autonomous mobile robots can be defined by the problem of stabilizing the robot on a reference path such that the tracking error converges to zero, and the progress of the robot along the reference path tends toward a nominal, nonzero rate [2]. One of the earliest path tracking strategies that showed promising results was the pure pursuit algorithm [9–10]. Other methods for path tracking involve modifying the reference path [11, 12] such that a feedback controller can converge the system onto the path. In addition to being more computationally intensive, these methods can fail when the path is designed independent of the dynamics of the robot using parametric curves such as B-splines. LQR based path tracking methods, while being optimal, require a linearized model of the system that possesses path dependent gains that must be adjusted for each path type [13]. Due to these complexities, pure pursuit remains as one of the most simple and effective algorithms in path tracking for a mobile robot. Due to its ease of implementation and robust nature of this method it is still very widely used in path tracking, especially for car-like robots [14, 15].

In the simplest form, pure pursuit (PP) can be considered as driving the robot to a constantly moving temporary goal point along the reference trajectory or path [10, 16]. For a given location of the robot, the algorithm starts with finding the closest point on the provided path. Next, the temporary goal point for the robot is chosen with a certain look ahead distance (∂_l) in front of the closest point along the path, in the direction of moving towards the final destination. This acts to guide the robot forward along the path. Finally, based on the type of the robot, control inputs (left and right wheel velocities for differential drive robot, steering angle and forward velocity for a car-like robot) are computed that will drive the robot to the temporary goal point. The whole process is repeated until the robot arrives within a given distance of the final goal point, referred to as the radius of the goal region (Gr). The look ahead distance is generally chosen to be a constant; however there are implementations where the look ahead distance is varied on-line based on the curvature of the path [17, 18]. The two main challenges arising in the original implementation of PP as described in literature are:

1. The algorithm can request sharp turns while the robot is moving at very high speeds.
2. The control action produced varies with the chosen look ahead distance.

Both issues must be addressed when implemented in hybrid architecture, as they may result in oscillatory motion of the robot when combined with state-of-the-art planning algorithms. Common implemented planners for autonomous robots include D*Lite [19] and RRT* [20], which in their simplest form approximate the motion of the robot with straight lines. As a result, the reference path is usually generated as a series of waypoints connected by line segments (henceforth referred to as path segments) leading to the goal. The pure pursuit algorithm will achieve zero tracking error while following a straight line segment, as the curvature of the path is zero. However, the changes in slope encountered while moving between path segments result in jerky motion of the robot. Such behavior is due to the fact that the feedback in pure pursuit is a function of the nearest point on the proposed path, as determined by the look ahead distance. As explained in [10, 16] changing the value of the look ahead distance changes the behavior of the controller. A small look ahead will allow the robot to follow the desired path more closely, but will produce oscillations in the motion of the robot. On the other hand, a large look ahead distance produces smooth motion but often

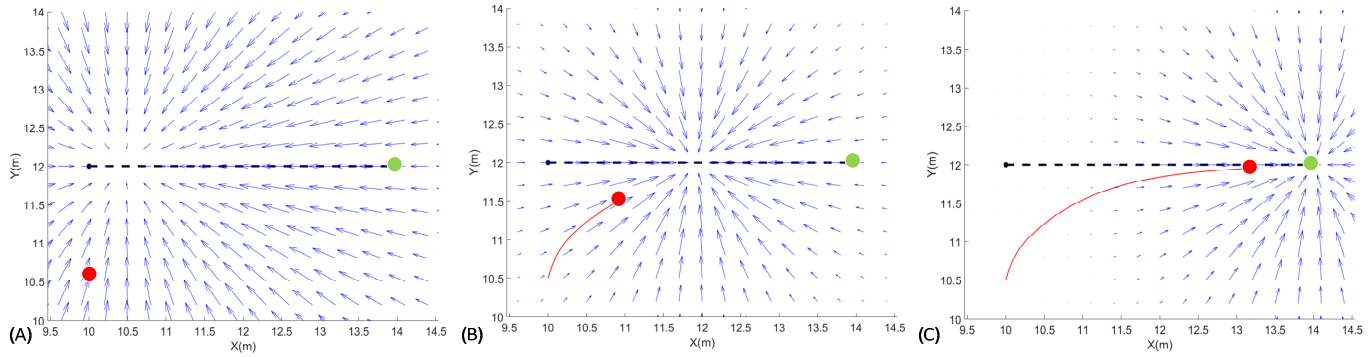

Figure 1. Commanded motion and gradient of Gaussian Kernel controller during different stages of path tracking. The robot path is shown in red, goal point in green and Gaussian gradient is shown in blue

results in cases where the robot does not follow the path very accurately, especially at sharp corners or when transitioning from one path segment to another. In such scenarios, the pure pursuit controller tends to cut the corners early, which is an especially dangerous behavior when operating in confined environments such as warehouses or hospitals.

Existing works in this domain have tried to solve the problem either by using smooth trajectories inside the planner or incorporating an external path smoothing operation performed before executing the path with a tracking controller. However, such smoothened paths can result in nonzero tracking error due to the curvature of the path, as mentioned above. A better approach would be to modify the pure pursuit algorithm itself to handle the above issues so that the algorithm implicitly produces smooth tracking for the robot. In the following sections, we propose modifications to the pure pursuit controller that result in an efficient path tracking algorithm which handles these shortcomings. The proposed algorithm is specifically designed and tested for differential drive robots, but can easily be generalized for other robot models.

3 PROPOSED CONTROLLER

The central idea behind the proposed controller is to generate control inputs using a Gaussian kernel centered at the temporary goal point to which the robot needs to be driven. In addition, the temporary goal point is computed as a weighted sum of the closest points to the robot on each of the individual path segments, with weights based on the relative distance of the robot from these points. The underlying assumptions for the Gaussian kernel (GK) controller working are detailed below:

1. A path leading to the goal has been computed by a planner, such as A* or RRT*. The path is then provided as a set of waypoints leading from a start location to the final goal point.
2. The kinematic parameters of the robot, i.e. the robot width and the wheel radius in the case of a differential drive robot, are assumed to be known.

Under these assumptions, the working of the controller can be explained as follows: Let N be the number of line segments on the path proposed by the planner. For any given position of the robot, the GK controller will find the closest point on each of the N segments. This can be done either analytically or through geometric methods. Temporary goal points are then chosen ∂_l distance in front of the closest point on each of the path segments. However, if the closest point is the end point on the goal side of a path segment, the temporary goal is chosen to be the end point itself. For each temporary goal point, a normalized 2D Gaussian distribution is created centered at that point. Each distribution has a standard deviation equal to the squared distance from the robot to the respective temporary goal point.

The controller consists of two dimensional circular Gaussian functions where the standard deviation along the X axis (σ_x) is equal to the standard deviation along Y axis (σ_y), allowing for the use of a single standard deviation term, $\sigma_x = \sigma_y = \sigma$. The normalized circular Gaussian is given by

$$F(x,y) = Ae^{-a((x-\mu_x)^2 + (y-\mu_y)^2)} \quad (1)$$

where,

$$A = \frac{1}{2\pi\sigma^2}$$

$$a = \frac{1}{2\sigma^2}$$

where, (μ_x, μ_y) is the center of the distribution.

The N Gaussian distributions can then be combined using the product rule, which for the special case of 2D circular Gaussian functions is given by

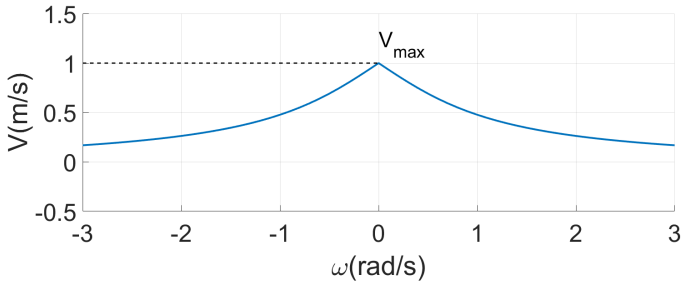

Figure 2. Plot showing the variation of V with ω for the proposed velocity controller

$$\boldsymbol{\mu}_f = \sum_{i=1}^{N} \frac{\boldsymbol{\mu}_i}{\sigma_i^2} \tag{2}$$

$$\frac{1}{\sigma_f^2} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \tag{3}$$

where $(\boldsymbol{\mu}_f, \sigma_f)$ denote the resulting final Gaussian and $(\boldsymbol{\mu}_i, \sigma_i)$ are the individual Gaussian distributions corresponding to the N path segments. Note that $\boldsymbol{\mu}_f, \boldsymbol{\mu}_i \in \mathbf{R}^2$. Combining the N temporary goal points for the robot, in this manner, results in the smooth motion of the center of the resulting Gaussian, gradually transitioning from one path segment to another as the robot progresses along the path.

The direction of the gradient of the distribution determines the desired orientation of the robot in order to approach or follow the path. The gradient $(\mathbf{G}(x,y))$ of the resulting Gaussian can be calculated using the following equation

$$\mathbf{G}(x,y) = -2a(\mathbf{X} - \boldsymbol{\mu}_f)F(x,y) \tag{4}$$

Note that $\mathbf{G}(x,y),\ \mathbf{X} \in \mathbf{R}^2$.

For a unicycle robot model, the desired angular velocity (ω) of the robot can be obtained from a simple proportional controller on the error in robot orientation as per the following equation;

$$\theta^* = \tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \tag{5}$$

$$\omega = K_p(\theta^* \Theta \theta);\ K_p > 0 \tag{6}$$

where θ is the current orientation of the robot, θ^* is the desired orientation, K_p is the proportional gain and Θ denotes difference taking into account the wraparound of angles.

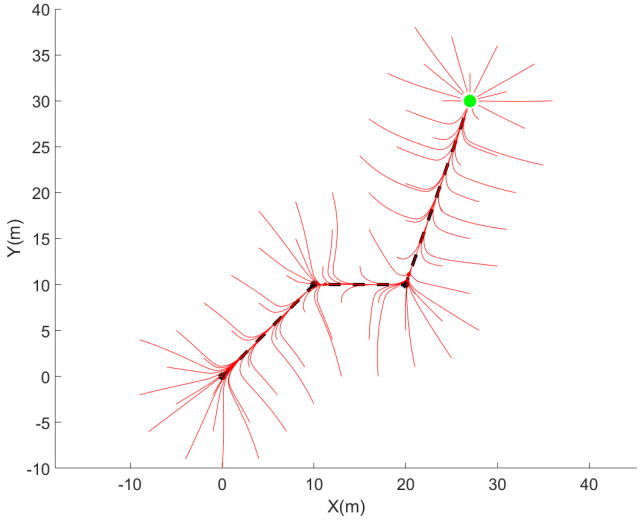

Figure 3. Path produced by the proposed controller for different start positions. Robot path is shown in red, reference path in black and goal point in green

The desired linear velocity (V) of the robot is varied based on the desired angular velocity of the robot;

$$V = V_{\max}\left(1 - \frac{2\tan^{-1}(|\omega|)}{\pi}\right) \tag{7}$$

The above formulation ensures a maximum linear robot velocity of $V_{\max}$, while down scaling nonlinearly for any nonzero ω, as shown in Fig. 2. Once the linear and angular velocities for a unicycle model are determined, they can be transformed into left and right wheel angular velocities (ω_l, ω_r) for a differential drive robot using the transformation equations;

$$\omega_l = \frac{2V - \omega L}{D}$$

$$\omega_r = \frac{2V + \omega L}{D} \tag{8}$$

where L is the width of the robot and D is the diameter of the wheels.

The working of the controller at various stages of tracking a path is shown in Fig. 1. The motions produced by the controller tracking a single path from random start points are shown in Fig. 3. In both the figures, the path to be tracked is shown in black, the robot is shown in red and the final goal is shown in green. Fig. 1 also illustrates the variations in gradient of the resulting Gaussian in blue for each stage. As shown by the gradient in Fig. 1(A) the final combined Gaussian distribution has a very high standard deviation when the robot is far away from the path. The distribution becomes more

localized as the robot gets closer to the path, as shown in Fig. 1(B) and (C). This changing Gaussian distribution produced by the GK controller drives the robot smoothly to the proposed path as demonstrated in Fig. 1 and Fig. 3. For generating both numerical simulations, a simple point robot model capable of freely moving in x and y direction was assumed. At each iteration, the robot was moved along the gradient vector as given by Eqn. (4). This was done solely for illustrating the path dictated by the controller. Proper simulations involving a differential drive robot model with control inputs for angular and linear velocity as given in Eqn. (5,7) are shown in the later sections of the paper.

Like the traditional pure pursuit algorithm, the proposed GK algorithm has only one tuning parameter; namely the look ahead distance (∂_l). The similarities in implementation mean that this controller is easily substituted into systems where pure pursuit was typically utilized.

Furthermore, the advantages of the controller are as follows:

1. The use of a separate velocity controller as given in Eqn. (7) scales the forward velocity of the robot so that the algorithm will request sharp turns only at low speed.
2. The consideration of the closest points in all of the path segments to produce the final Gaussian allows the algorithm to work with a small look ahead distance without producing oscillations in the motion of the robot. This in turn enables the algorithm to track the path more accurately, while still maintaining smooth motion.

These advantages particularly address the shortcomings of pure pursuit as described previously. The following section illustrates these factors by comparing the performance of the proposed controller with pure pursuit in simulation.

4 SIMULATION

For the purpose of validating the proposed control scheme and to compare its performance with pure pursuit, the two controllers were utilized to drive a simulated Pioneer P3DX robot and a tracked robot in the V-REP robotic simulator [21]. For the simulations, the controllers (both pure pursuit and Gaussian kernel) were run in MATLAB and co-simulated with V-REP through the remote API Bindings. Two sets of simulations were conducted, the first to compare the performance of the controllers while tracking the reference path when the robots are initialized at different start points and the second to evaluate their performance while tracking the reference path in the presence of disturbances.

4.1 Path tracking with different start points

In the initial simulation, the environment consisted of a Pioneer P3DX on a flat terrain 20m X 20m in size. A series of connected path segments, analogous to those provided by a high level planner such as RRT*, were passed to the controller as the desired path. The waypoint coordinates in meters were [(2, 2), (5, 8), (10, 8), (10, 12)], in that order, with (10, 12) being the final goal position. For this simulation, the robot tracks the path while starting from nine different initial start points, as shown Table 1. For each of the trials, the robot was first made to follow the reference path under the action of the pure pursuit algorithm provided in the MATLAB Robotics Toolbox [16].

TABLE 1. Results of path tracking simulation with different start points

Trial No:	Starting Point	Pure Pursuit		Gaussian Kernel	
		MCTE (m)	Time (s)	MCTE (m)	Time (s)
1	(0,0)	0.4859	38.19	0.4178	41.49
2	(4,0)	0.6798	37.30	0.4973	42.20
3	(0,5)	0.9078	32.70	0.6478	39.20
4	(10,4)	3.0187	17.60	3.0590	17.70
5	(4,10)	0.5249	22.10	0.3889	26.60
6	(7,5)	1.4380	18.30	1.0565	24.90
7	(8,10)	0.7910	12.10	0.7497	11.90
8	(12,5)	2.6338	16.70	2.3017	19.20
9	(10,10)	1.6312	6.70	0.9295	11.80

Once that action was completed, the robot then utilized the Gaussian kernel controller in order to track the path from the same series of starting points. In both cases, the controllers updated at a frequency of 50 Hz. For the purpose of this simulation, precise positioning information obtained from the simulators is used without any noise.

The look ahead parameter was manually tuned for the pure pursuit implementation as described in [10, 16] in order to produce optimal tracking behavior. The complete set of parameters for the pure pursuit implementation is as follows:

Desired linear velocity = 0.05m/s
Maximum angular velocity = 1.0rad/s
Look ahead distance = 0.8m
Radius of Goal region = 0.1m

The look ahead distance for the pure pursuit controller was found to be optimal at 0.8m, as any lower value produced oscillations in the motion of the simulated robot. As stated previously, it is desirable to have the smallest possible look ahead distance in order to more closely track the reference path.

Similarly the parameters were tuned for the Gaussian kernel controller and the values are:

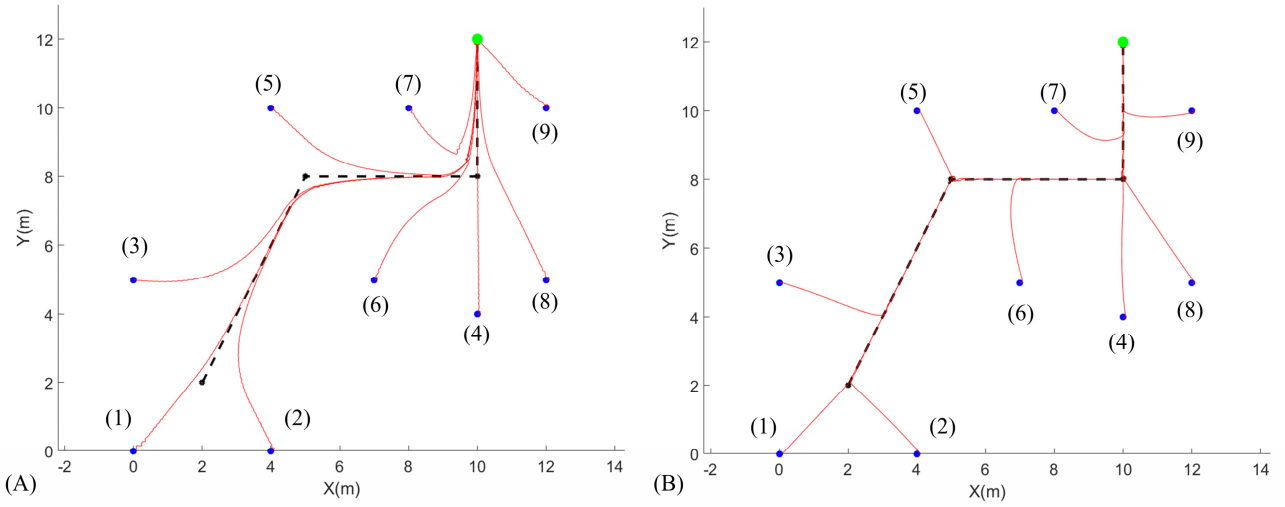

Figure 4. Comparison of the controller performance: (A) Pure pursuit, (B) Proposed controller

Maximum linear velocity = 0.05m/s
Proportional gain on angular velocity = 0.6
Look ahead distance = 0.1m
Radius of Goal region = 0.1m

Both pure pursuit and the proposed controller produce V and w control commands for the simulated robot. These commands are first converted into left and right wheel velocities (ω_l, ω_r) for the simulated robot using the transformation given in Eqn. (8) and then are executed by the inbuilt joint velocity controller in VREP. The trajectories followed by the robot are shown in Fig. 4. The various start points of the robot are shown in blue and labeled by their respective trial numbers, as in Table 1. The reference path is shown in black, while the actual path followed by the robot is in red and the final goal point is in green. In order to quantify the performance of the controllers for path tracking, we introduce the cross-track error, which is defined as the Euclidean distance between the robot and the closest point on the path computed for every iteration of the controller. The mean cross-track error (MCTE), the cumulative per trial cross track error averaged over the total number of controller iterations, is then calculated. The values of the MCTE, along with the time taken by the controller to drive the robot to the goal, for each trial are given in Table 1.

As shown by Fig. 4 the proposed controller results in closer path tracking as compared to pure pursuit method. This claim is also supported numerically by the mean cross track error values presented in Table 1. For all start points except that of trial 4, the GK controller results in smaller mean cross track error than pure pursuit. In the case of trial 4, both trajectories are similar in form, thus resulting in nearly equal mean cross track error. While Fig. 4 shows that the robot makes sharp turns under the action of the proposed controller, the motion produced is still smooth due to the fact the linear velocity controller as given by Eqn. (7) reduces the velocity in order to enable the robot to make the turn in place. This behavior then contributes to the fact that the proposed controller takes more time to reach the goal when compared to pure pursuit controller, as shown in Table 1.

4.2 Path tracking under disturbance

Autonomous motion under challenging terrain conditions, such as the motion of a tracked robot in rough terrain, can result in high levels of disturbance being introduced to the system. Some examples of such disturbances include slip experienced by the robot, actuator limitations restricting robot motion in high slope regions, and sensor noise. In all these circumstances, it is extremely valuable to have a robust path tracking controller to ensure that the robot follows the path accurately. In order to study path tracking under disturbance, the proposed controller was simulated on rough terrain, and its performance was compared to that of pure pursuit.

The simulation environment consisted of a differential drive tracked robot model operating on a rough terrain of size 9m X 7m, as shown in Fig. 5. The waypoint coordinates for the reference path, in meters, were [(2, 2), (0, 8), (6, 8)], with (6, 8) being the final goal. For this simulation, the robot was made to track the path starting from (0, 0), first under the action of a pure pursuit controller and then under the proposed GK controller. As before, the controllers were updated at a frequency of 50 Hz and precise positioning data is used. The desired linear velocity for pure pursuit and the maximum linear velocity for the proposed controller were increased to 0.06m/s in order to allow the robot handle the rough terrain. All other parameters were unchanged from the prior simulation. The height of the terrain varies from 0m to 3m and the maximum torque on each of the motors on the tracked robot was limited to 10 Nm inside the V-REP simulator.

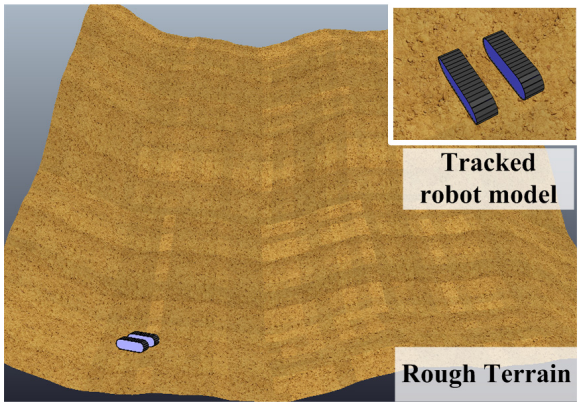

Figure 5. Simulated environment utilized for studying path tracking performance under disturbance in V-REP

The trajectories followed by the robot under the action of both the controllers are shown overlaid with the terrain topography map in Fig. 6. The start point is shown in cyan, reference path in black and the goal point in green. The path followed by the robot under the action of the pure pursuit controller is shown in red and the proposed controller is shown in blue. As shown by Fig. 6, even in the presence of disturbance, the GK controller enables the robot to follow the path more closely as compared to pure pursuit. For this simulation case, the starting point of the last path segment is in a region of steep slope due to the large gradient in terrain height. As shown in Fig. 6, the steep terrain causes the robot to deviate from the desired path at this point in both cases. But the GK controller recovers much faster as compared to the pure

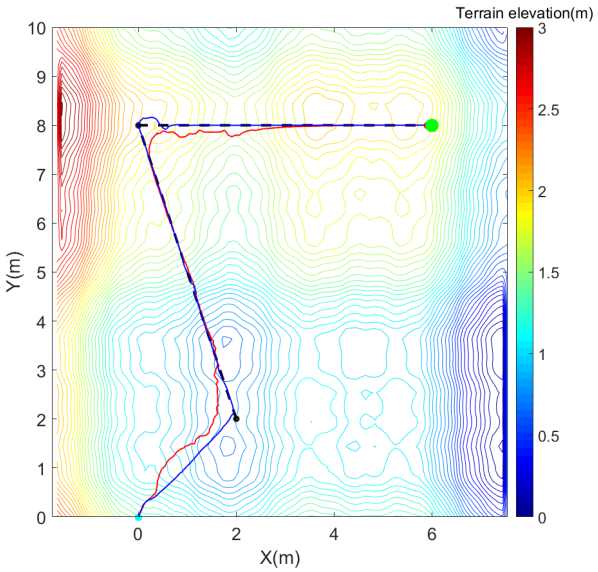

Figure 6. Comparison of path tracking in the presence of disturbance. Path followed under the action of the proposed controller is shown in blue and pure pursuit controller is shown in red

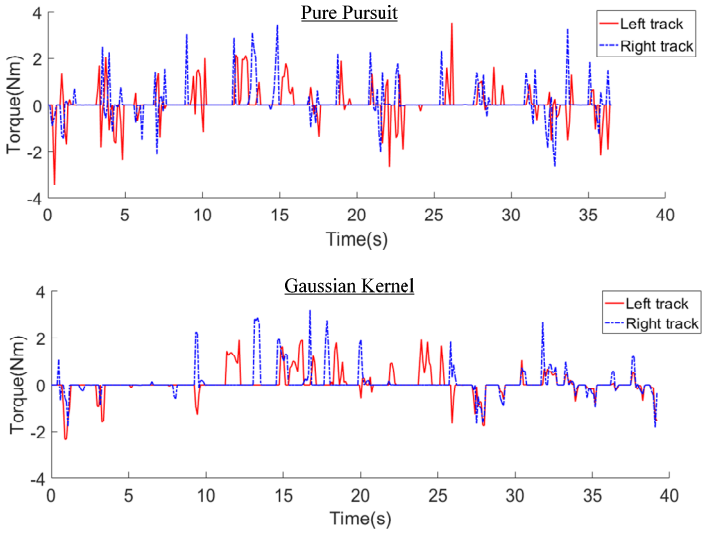

Figure 7. Torque applied on the left and right tracks of the robot for the path tracking under disturbance simulation

pursuit method. Figure 7 shows the torque acting on the left and right tracks for the above simulation case.

The results of both simulations demonstrate that the proposed GK controller has superior path tracking characteristics as compared to the pure pursuit algorithm. This behavior is demonstrated both when navigating from a variety of initialization points as well as in a non-uniform terrain.

5 CONCLUSION AND FUTURE WORK

This paper presented a novel Gaussian kernel controller for path tracking in mobile robots. The controller was designed to address shortcomings in existing pure pursuit implementations in order to produce smooth motions while ensuring accurate tracking of the provided reference path utilizing the gradients of moving Gaussian kernels. Based on the simulation results, the proposed controller demonstrated superior path tracking performance as compared to the traditional pure pursuit algorithm. The simulations compared the performance of controllers for two different cases, one where the robots tracked a single reference path when initialized from a variety of start points and the other where the robots tracked a reference path through a rough terrain in the presence of environmental disturbance.

Future plans for continued research into the applications and features of the proposed controller will focus on real-life implementation. Hardware tests will include the usage of the GK controller by a variety of unique mobile robots, including the Hybrid Mechanism Mobile Robot (HMMR) [22] and STORM [23]. Tests will be conducted to verify the performance of the controller in the presence of both sensory and environmental noise, such as in high slip conditions or with an intentionally noisy feedback device. In addition to differential drive robots, the controller will also be explored for

path tracking in other robot models, such as car-like robots and quadcopters. Additionally, the use of an ellipsoid Gaussian in the controller, and the attendant additional tuning adjustments required to redesign the controller in such a form will be explored. The author's hypothesize that the use of an ellipsoid Gaussian can lend itself to larger gradients adjacent to the path, should the two axes be chosen such that the larger standard deviation (and thus the major axis of the ellipsoid distribution) is parallel to the path and the smaller standard deviation (and minor ellipsoid axis) is perpendicular to the path. This may provide a stronger propensity of the controller to keep the robot on the path at all times.

ACKNOWLEDGMENT

This work is supported by the US Army Medical Research & Materiel Command's Telemedicine & Advanced Technology Research Center (TATRC), under Contract No. W81XWH-16-C- 0062. The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy, or decision unless so designated by other documentation.

REFERENCES

[1] Kruse, T., Pandey, A. K., Alami, R., and Kirsch, A., 2013. "Human-aware robot navigation: A survey". Robotics and Autonomous Systems, 61(12), dec, pp. 1726–1743.

[2] Paden, B., Cap, M., Yong, S. Z., Yershov, D., and Frazzoli, E., 2016. "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". IEEE Transactions on Intelligent Vehicles, 1(1), mar, pp. 33–55.

[3] Egerstedt, M., 2000. "Behavior based robotics using regularized hybrid automata". In International Workshop on Hybrid Systems: Computation and Control. IEEE, pp. 103– 116.

[4] Howard, T. M., and Kelly, A., 2007. "Optimal rough terrain trajectory generation for wheeled mobile robots". The International Journal of Robotics Research, 26(2), pp. 141– 166.

[5] Reina, G., Bellone, M., Spedicato, L., and Giannoccaro, N. I., 2014. "3D traversability awareness for rough terrain mobile robots". Sensor Review, 34(2), pp. 220–232.

[6] Currier, P. N., andWicks, A. L., 2013. "A novel method for prediction of mobile robot maneuvering spaces". Journal of Terramechanics, 50(2), apr, pp. 85–97.

[7] Park, J. J., and Kuipers, B., 2011. "A smooth control law for graceful motion of differential wheeled mobile robots in 2D environment". Proceedings - IEEE International Conference on Robotics and Automation, pp. 4896–4902.

[8] Gulati, S., and Kuipers, B., 2008. "High performance control for graceful motion of an intelligent wheelchair". Proceedings - IEEE International Conference on Robotics and Automation, pp. 3932–3938.

[9] Wallace, R., Stentz, A., Thorpe, C. E., Maravec, H., Whittaker, W., and Kanade, T., 1985. "First Results in Robot Road-Following.". Ijcai, pp. 1089–1095.

[10] Coulter, R. C., 1992. "Implementation of the Pure Pursuit Path Tracking Algorithm". Communication(January). [12] Amidi, O., and Thorpe, C. E., 1990. Integrated mobile robot control. Tech. Rep. May, Carnegie Mellon University, Pittsburgh, PA, may.

[11] Micaelli, A., Samson, C., Micaelli, A., and Samson, C., 1993. Trajectory tracking for unicycle-type and two steering- wheels mobile robots. Tech. rep., INRIA.

[12] Piazzi, A., Lo Bianco, C., and Romano, M., 2007. "Splines for the Smooth Path Generation of Wheeled Mobile Robots". IEEE Transactions on Robotics, 23(5), oct, pp. 1089–1095.

[13] Cordeiro, R. A., Azinheira, J. R., De Paiva, E. C., and Bueno, S. S., 2013. "Dynamic modeling and bio-inspired LQR approach for off-road robotic vehicle path tracking". 2013 16th International Conference on Advanced Robotics, ICAR 2013.

[14] Buehler, M., Iagnemma, K., and Singh, S., eds., 2009. The DARPA Urban Challenge, Vol. 56 of Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, Berlin, Heidelberg.

[15] Ohta, H., Akai, N., Takeuchi, E., Kato, S., and Edahiro, M., 2016. "Pure Pursuit Revisited: Field Testing of Autonomous Vehicles in Urban Areas". In 2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA), IEEE, pp. 7–12.

[16] Corke, P., 2017. Robotics, Vision and Control, Vol. 118 of Springer Tracts in Advanced Robotics. Springer International Publishing, Cham.

[17] Wit, J., Crane, C. D., and Armstrong, D., 2004. "Autonomous ground vehicle path tracking". Journal of Robotic Systems, 21(8), pp. 439–449.

[18] Giesbrecht, J. L., Mackay, D., Collier, J., and Verret, S., 2005. "Path tracking for unmanned ground vehicle navigation". DRDC Suffield TM(December).

[19] Koenig, S., and Likhachev, M., 2002. "D* Lite". Proceedings of the Eighteenth National Conference on Artificial Intelligence, pp. 476–483.

[20] Karaman, S., and Frazzoli, E., 2010. "Incremental Sampling-based Algorithms for Optimal Motion Planning". CoRR, abs/1005.0, may.

[21] Coppelia Robotics, 2018. V-REP.

[22] Ben-Tzvi, P., Goldenberg, A. A., and Zu, J. W., 2010. "Articulated hybrid mobile robot mechanism with compounded mobility and manipulation and on-board wireless sensor/actuator control interfaces". Mechatronics, 20(6), sep, pp. 627–639.

[23] Kumar, P., Saab, W., Ben-Tzvi, P., 2017. "A Hybrid Tracked-Wheeled Multi-Directional Mobile Robot". ASME Journal of Mechanisms and Robotics, Under review.