Abstract

The purpose of this study is to minimize the arc length for the path described by the model of a robot platform when the path is constrained to have smooth transitions given by sigmoid functions. The optimization required a proof of stability for the resulting control law, the selection of the best sigmoid function among ten functions, and the definition of two gains necessary to parameterize the control law. The optimization was carried out by simulating the system under several kinematic and dynamical conditions, and the best sigmoid function was a hyperbolic tangent. Thus, the motion control first implied the simulation of a reference model to define an optimal path, and later the control of an actual robot platform, which followed the optimal path. The use of the optimized path reduced the complexity of the controller while allowing natural and intuitive paths for the robot platform.

Keywords path optimization, smooth control law, motion control

1. Introduction

Path optimization has been a growing area in motion planning for robot platforms. Optimization looks for the best route from point A to point B under certain constraints such as energy consumption, speed limits, or obstacle avoidance [1-3]. Path optimization techniques encompass pseudospectral optimization [4], elastic bands [5], or bio-inspired algorithms such as ant colony optimization [6] among many others. Applications include but are not limited to aerial vehicle navigation [7], underwater vehicle command [8], and motion of robot platforms [9]. Almost every application requiring motion can benefit from optimization, which offers potential savings in energy and more natural and intuitive paths for devices.

The work in [10] proposes the use of a smooth control law to define the path of a robot instead of limiting the motion planning to define the goal, as in traditional mobile robotics. This control strategy divides the kinematics of the model into a fast component that defines the heading angle of the robot and a slow component that sets the forward speed. The resulting trajectory is intuitive and comfortable, making it suitable for applications such as motion planning for a wheelchair. The authors test a single sigmoid function, for which the application of the control law requires the definition of two constants (k₁ and k₂). The values of these constants were determined by experimentation; thus, their selection did not result from an optimization process as in the present paper, in which the optimization uses ten sigmoid functions with the goal of minimizing the arc length.
Most of the works in motion control for robot platforms divide the analysis of the plant in two. For instance, the work in [11] proposes a continuous and smooth control law. In this case, an outer control problem defines the angular velocity in terms of radial and angular errors, and also a constant $k_1$. The inner control defines the velocity of the robot in terms of the outer control part, errors in angular velocity and a constant, $k_2$. The paper presents a proof of stability, but the work lacks application. Nevertheless, simulations demonstrate the performance of the control law, which requires a long time to stabilize.

The applications of path optimization are numerous. One of the most futuristic applications is the UAV, particularly the PAV (Personal Aerial Vehicle), interest in which has increased as a result of the development of aerial space technology. For instance, the work in [12] presents a methodology to define the best route for a PAV in order to minimize probabilistic risks. Thus, the best path reaches point B after avoiding one or more obstacles, for example, a no-fly zone or a storm.

This paper uses ten sigmoid functions, instead of a single function as in [10], and experimentally looks for every possible kinematic condition for pairs of departure and arrival angles of a platform model, in order to choose the three sigmoid functions that produce the shortest paths. These paths require smoother control laws, which in addition prove more intuitive than the paths that result from the application of a traditional PID controller to set a path. This paper also proposes a method to choose the best sigmoid function, which consists of setting a path that forces the platform to express a broad dynamic range. The validation in [10] consists of applying the definition of the single sigmoid function to the control of a wheelchair, but there are no comparisons with other techniques in order to evaluate the control performance. In contrast, this paper compares every result with traditional PID control of a kinetic and dynamic model, and also presents the control of an actual robot platform, a National Instruments robot platform (DaNI 2.0).

This paper is organized as follows. Section 2 presents the design of a path for a robot based on sigmoid functions, including a proof of stability using Lyapunov stability theory. In Section 3 an optimization process comes up with the sigmoid function that generates the shortest arc length produced by the motion of a robot under a wide range of dynamic conditions. The definition of a path for the model of a robot platform in Simulink serves to test the optimal sigmoid function. This model includes kinematics and dynamics, as shown in Section 4. Finally, in Section 5, the optimal sigmoid function defines a reference path for the DaNI 2.0 robot.

2. Model and control laws

Traditional motion control defines the coordinates of a target, $\langle x_f, y_f, \theta_f \rangle$, as shown in Figure 1, leaving intermediate points unspecified. In contrast, the control scheme in this paper defines the whole path from an initial coordinate to the target. In addition, this control law requires a reference model, which tracks the target describing the route for the actual robot platform. This reference may be the model of the robot platform itself. This case guarantees that the actual platform can follow the reference path. One of the main features of the path defined in this paper is its smoothness. As a result, the reference model, located at $\langle x_m, y_m, \theta_m \rangle$, follows the dotted line in Figure 1. The coordinates of the actual robot platform in Figure 1 are $\langle x, y, \theta \rangle$.

Thus, the reference model tracks the target, while the robot follows the model.

The control law for the reference model aims to reduce the distance $r$, and at the same time transform the angular position from $\theta_m$ into $\theta$. The work of the robot platform controller consists of minimizing the distance $r$, between platform and reference model while changing the angle $\theta$ in such a way as to reduce its difference with $\theta_m$.

The model of the platform corresponds to a differential wheeled mobile robot in a 2D environment, as shown in equation (1). This model may be divided into two parts for analysis. The first part corresponds to the first two variables, $r$ and $\theta$. The dynamic of this part is slow compared with the dynamic of angle $\delta$. Thus, the control strategy consists of defining $\omega_m$ in such a way that $\delta$ has an appropriate value. Thus, once $\delta$ is controlled, the remaining part of the control process consists of reducing the radius $r$, for instance, by defining a constant speed $v_m$ as in this paper.

$$
\begin{bmatrix}
\dot{r} \\
\dot{\theta} \\
\dot{\delta}
\end{bmatrix} =
\begin{bmatrix}
-v_m \cos(\delta) \\
0 \\
\omega_m \sin(\delta) + \omega_m
\end{bmatrix}
$$

Figure 1. Target (green), reference model (dashed red), and actual platform (solid red)
In this section, we define $\omega_n$ such that transitions in $\delta_i$ remain smooth across the whole reference path. The work in [10] proposes the use of an $arctan(x)$ function as a candidate to define the smooth transitions. Extending this work, this paper compares ten sigmoid functions in order to find the best function to minimize the arc length travelled by a reference model between the initial coordinates $<x,y,\theta>$ and the final coordinates $<x_f,y_f,\theta_f>$. The basic set of functions comprises five sigmoid functions: the inverse tangent ($sf1$), the hyperbolic tangent ($sf2$), the error function ($sf3$), the algebraic function ($sf4$), and the Gudermannian function ($sf5$), as defined in equation (2).

The constant $k_i$ serves to generate ten functions from these five:

\[
sf1(x) = k_{i1} \arctan(k_{i1}x) \\
 sf2(x) = k_{i2} \tanh(k_{i2}x) \\
 sf3(x) = \frac{k_{i3}}{\sqrt{1+k_{i3}^2}} \int_0^x e^{-(k_{i3}^2)t^2} dt \\
 sf4(x) = k_{i4} \frac{k_{i4}^2 x}{\sqrt{1+(k_{i4}x)^2}} \\
 sf5(x) = k_{i5} 2 \arctan \left( \frac{k_{i5}^2 x}{2} \right) \tag{2}\]

The first five functions, in the upper part of Figure 2, share a unitary slope at $x = 0$, which can be achieved by properly defining $k_{i1}$ for instance the first function $atan = \arctan(x)$. The other five functions in the bottom of Figure 2, to complete ten trial functions are also based on the definitions in equation (2), but all these reach 0.99 at $x = \pi$. A normalization process computes adequate gains of $k_i$ in each case, for instance $atan1 = (1/1.5692)arctan(18.4338x)$. The other functions follow the same rules.

![Figure 2. Sigmoid functions. Slopes are normalized in the upper plot, amplitudes and steady-state values in the lower.](image)

Figure 2. Sigmoid functions. Slopes are normalized in the upper plot, amplitudes and steady-state values in the lower.

The use of a Lyapunov candidate function as in equation (3) serves to demonstrate that the sigmoid functions produce stable control laws for the slow part of the model ($r$ and $\theta$). The control angle is defined as shown in equation (4).

\[
 V = \frac{1}{2} \left( r^2 + \theta^2 \right) \tag{3} \\
 \dot{\delta}_i = sf(-k_i \theta_i) \tag{4} \\
 \dot{\delta}_i = sf(-k_i \theta_i) \tag{4}
\]

The derivative of the function $V$ is written in equation (5).

\[
 \dot{V} = r_i \dot{r}_i + \dot{\theta}_i \theta_i \\
 = -r_i \nu_m \cos(sf(-k_i \theta_i)) + \frac{\omega_m}{r_i} \theta_i \sin(sf(-k_i \theta_i)) \tag{5}
\]

An observation of the sigmoid functions $sf1$, $sf2$, $sf3$, $sf4$ and equation (5) allows the analysis in equation (6).

\[
\cos(sf(-k_i \theta_i)) > 0, \text{ } \forall \theta_i \in [-\pi, \pi] \\
\text{sgn}(sf(-k_i \theta_i)) = -\text{sgn}(\theta_i) \tag{6}
\]

The rate of change of $V$ (i.e., dot $V$) is negative definite, which guarantees asymptotic stability. The sigmoid function $sf5$ is negative semidefinite (NSD), thus establishing stability according to Lyapunov’s second method. Now we define the control law for the steering as the difference between $\delta_i$ and the sigmoid function, as shown in equation (7). Thus, given the model in equation (1), the derivative of $z$ corresponds to the expression in equation (8).

\[
 z = \delta_i - sf(-k_i \theta_i) \tag{7} \\
 \dot{z} = \frac{\nu_m}{r_i} \sin(sf(-k_i \theta_i)) + \omega_m - sf(-k_i \theta_i) \dot{\theta}_i \tag{8}
\]

If the rate of change of $z$ has the opposite sign of $z$, as defined in equation (9), then we can guarantee stability of the control law. The final control law results from equating $\dot{z}$ in equation (8) and equation (9), as shown in equation (10).

\[
 \dot{z} = -k_2 \frac{\nu_m}{r_i} z \tag{9} \\
 \omega_m = \frac{\nu_m}{r_i} \left[ -k_2 \left( \delta_i - sf(-k_i \theta_i) \right) - \sin(\delta_i) \right] \tag{10}
\]

The robot model is controlled by defining a constant linear speed $\nu_m$, while the angular speed $\omega_m$ leads the model to reach the target coordinates. Thus, optimizing the motion path corresponds to finding $k_1$, $k_2$, and $sf$ in equation (10) such that the arc length between an initial coordinate $<x,y,\theta>$ and the target $<x_f,y_f,\theta_f>$ is minimized.
3. Path optimization

The motion planning for the model of a robot platform requires defining the coordinate $\langle x, y, \theta \rangle$ at every instant in time. Therefore the design of an optimal path looks to find those coordinates that minimize the trip of the platform under certain constraints. The minimization process will exclusively consider paths that terminate within $1^\circ$ of $\theta$. In addition, all simulations define the linear speed of the platform $v_\infty$ in 0.2 m/s. Furthermore, in order to avoid instabilities, the speed falls to $v = r_i/5$ if $r_i$ is smaller than 5 cm. The arc length computation uses only the length for the constant speed path. The model of the platform was simulated in discrete mode using a sampling time of 20 ms and a final time equal to 30 s.

The optimization comprises two stages: 1) finding the three sigmoid functions that produce the shortest paths among the ten trial functions; 2) finding the sigmoid function that produces the shortest arc length, and also the corresponding pair $k_1, k_2$ that minimizes the path. In the first stage, combinations of departure-arrival angles are tried, choosing the sigmoid function that minimizes the arc length for each trial. A whole set of values $k_1$ and $k_2$ are tried for each function. Once each pair of angles has been tried, the three functions that repeat the most are chosen as winners. In the second stage, and using the path defined below and the three best functions of the first stage, the sigmoid function that minimizes the arc length is chosen, and also the pair $k_1$ and $k_2$ that produces the minimum length as the best pair of gains.

![Figure 3. Motion of the robot platform between two points using five sigmoid functions](image)

The first stage starts with setting a departure angle ($\theta_0$) and an arrival angle ($\theta_1$) in the range ($-\pi$, $\pi$), for instance $\theta_0 = 3\pi/4$ and $\theta_1 = -3\pi/4$, as shown in Figure 3. Every initial coordinate has the form $\langle 0, 0, \theta_0 \rangle$, whereas the final coordinate equals $\langle 1, 1, \theta_1 \rangle$. Then, for each pair $\langle k_1, k_2 \rangle$, the optimization process finds the sigmoid function that generates the shortest path. This process forms an elevation map, as shown in Figure 4, where each pixel represents the sigmoid function with the shortest path associated to each coordinate $\langle k_1, k_2 \rangle$.

The winning function of all shown in Figure 4 is erf. This sigmoid corresponds to the function that minimizes the arc length across the whole plane. The ranges for $k_1, k_2$, were chosen experimentally, given that, in general, $k_1$ defines the arrival angle, whereas $k_2$ defines the curvature of the whole path. Thus, the range for $k_1$ was chosen to allow errors within $1^\circ$ for $\theta_y$, and the wide range for $k_2$ assures that the best curvature is within permissible values.

![Figure 4. Best sigmoid functions for pairs $\langle k_1, k_2 \rangle$ for fixed depart and arrive angles](image)

Finding the shortest arc length given a single pair $\langle \theta_x, \theta_y \rangle$, as shown in Figure 4, presents a very restricted case. A more elaborate minimization uses any possible departure and arrival angles, as shown in Figure 5. Each pixel in this figure summarizes a whole behaviour, such the one in Figure 4, in one single point, and corresponds to the sigmoid function that minimizes the path length across the plane, regardless of $k_1$ and $k_2$ values. The best sigmoid in Figure 5, also the result of the first stage, covers the maximum area of the elevation map. The three best sigmoid functions are erf, alf, and tanh1. These three functions cover more than 70% of the plot in Figure 5.

The second stage of the optimization process uses the three best sigmoid functions from the first stage to find the best pair $\langle k_1, k_2 \rangle$ as well as the best function. Instead of reading those constants from Figure 5, in this stage a simulated platform follows an imaginary path, forcing the platform to express a broad range of dynamics. Thus, the path is formed by a set of segments of unit length, as shown in Figure 6.

The initial coordinate in Figure 6 is $\langle 0, 0, 0 \rangle$ in the $xy$-plane, and the final one is $\langle 3.26, -2.73, 0 \rangle$. The transition angles between consecutive segments change from -150° to 150° in increments of 30°. This stage defines the angle to cross every intersection as perpendicular to the line that bisects two consecutive segments. The value for $k_1$ varies from 0.2 to 2, while the value for $k_2$ varies from 1 to 150. The best result in this second stage was achieved using tanh1 with $k_1 = 1$, and the wide range for $k_2$ assures that the best curvature is within permissible values.

![Figure 5. Motion of the robot platform between two points using five sigmoid functions](image)
According to the scheme in Figure 1, the reference model defines a smooth path to reach the target \( x_f, y_f, \theta_f \). The goals for the reference model consist of shrinking \( r_c \) to zero and changing \( \theta_c \) to approach \( \theta_f \). At the same time, the robot platform follows the path described by the reference model as closely as possible. Therefore, the control law for the platform actually may require two laws. The distance \( r_c \) should go to zero, while \( \theta \) approaches \( \theta_f \). Definitions for variable \( r_c \) and \( \theta_c \) correspond to the ones given in equation (12).

\[
r_c = \sqrt{(x_m - x)^2 + (y_m - y)^2}
\]
\[
\theta_c = \arctan\left(\frac{y_m - y}{x_m - x}\right)
\]  

(12)

Once the initial and final coordinates for each segment of the path have been defined, the application of the reference model concept defines \( x_{ref}, y_{ref} \) for each instant in time.

The first control algorithm (PID 1) controls the \( \theta \) angle, while the second (PID 2) regulates the distance between platform and reference model, \( r_c \). These are traditional PID controllers, where the actuating signal, \( u_c \), equals \( kp(e) + ki(e) + kd(de/dt) \), with input \( e \). The error \( e = \theta_c - \theta \) for PID 1, and \( e = r_c \) for PID 2. Experimental gains for both controllers are 0.1, 0, 0 for angle control, and 0.25, 0, 0 for radial control. The gains when the platform follows a reference path are 0.2, 0, 0 for the angle and 0.5, 0, 0 for the radius. The differences between these gains find their basis in the error for each strategy. In classical control, the goal is far from the robot, while the platform is close to the reference model. Each controller, including the one that represents the dynamic of the wheels includes natural limits, for instance, maximum speeds and maximum angles. The constants \( x_0, y_0, \theta_0 \) in Figure 7 are the initial conditions for the robot.

\[
\theta = \theta_0 + \frac{1}{b} \left( v_R - v_L \right) dt
\]
\[
x = x_0 + \frac{1}{2} \left( v_R + v_L \right) \cos(\theta) dt
\]
\[
y = y_0 + \frac{1}{2} \left( v_R + v_L \right) \sin(\theta) dt
\]  

(11)

4. Application

This section uses a model of a robot platform as the actual plant, whereas the next section uses the commercial platform DaNI2.0. In order to approach normal operation conditions, the model includes not only the kinematics, given by equation (1), but the dynamical description for some parts of the platform. The model of each wheel includes a control loop governed by a PI with gains \( kp = 1 \) and \( ki = 1 \), and a transfer function \( H(s) = 1/(0.2s + 1) \) representing the dynamic of each motor, as shown in [14]. The variable speeds \( v_R \) and \( v_L \) in equation (11) represent linear speed of the right and left wheel, as defined in [14], and \( b \), the distance between wheels, equals 0.37 m.
trip duration. Additionally, the robot cannot pass through point 3, because it receives the coordinates for point 4 before reaching point 3.

Figure 8. Position control with and without a reference model

5. Real-time control application

The experimental path optimization was tested using the robot DaNI 2.0 from National Instruments. This three-wheeled, pre-assembled, mobile robot platform has sensors, motors, and an NI 9632 board mounted on top of a robot base. The control algorithm looks for a match between \( \theta \) and \( \theta_c \), every 50 ms, while maintaining the linear speed constant at 0.2 m/s.

The motion control starts by defining the reference path, given a model of the platform as defined in equation (1) with the control rule of equation (10). This process is performed before the actual motion of the robot. Every pair of consecutive sub-goals (numbered from 1 to 7 in Figure 9) serves as initial and final sub-goal for defining \( \{x_m, y_m\} \). The resulting \( \{x_m, y_m, \theta\} \) coordinates are saved in a text file. The communication with the robot as well as the estimation for position and angle (using signals coming from the robot’s encoders) are wired into Labview. However, the control algorithm is programmed in Matlab. Thus, every 50 ms, Labview calls a function which is run in Matlab. This function reads the current position \( \{x,y,\theta\} \), as well as an index for the current time from Labview, and the corresponding pair of coordinates \( \{x_m, y_m, \theta\} \) from the text file. The outputs for the function are the left \( v_l \) and right \( v_r \) speeds, which are later transformed into linear speed \( v \) and angular velocity \( \omega \) in Labview. These last two are the parameters used to control the robot in Labview. Matlab internally computes \( \Theta \), based on \( \{x_m, y_m\} \) and \( \{x,y,\theta\} \) according to equation (11). Then the actuating signal is defined as a proportion of the difference between \( \theta \), and \( \Theta \), plus the constant 0.2 representing the nominal linear speed. The same script records \( x, y, \theta, \Theta, v_l, v_r \), in order to run comparisons and also to tune the P controller.

The path that was used to test the motion of the robot platform is the same as in Figures 6 and 8. The only difference is that the distance between one landmark and the next is two metres, instead of one. The results in Figure 9 show that the reference model presents a smooth and easy path for the robot. This may be guaranteed by two conditions: 1) the maximum error in angle is lower than \( \pi/6 \) radians, which appears at the beginning of the motion; 2) the average error is less than \( \pi/15 \) radians. On the other hand, a P controller was enough to set the dynamic, which shows that a very low complexity for the controller is required to control the system.

The upper panel in Figure 9 shows the path defined by the reference model and the actual path of the platform. Excluding the behaviour immediately after landmark 3, there is a good match between these paths. The reference model defines an ideal position \( x_m \) and \( y_m \) at every instant in time, as shown in the middle panel of Figure 9. The lower panel shows the error in the control variable \( \theta \).

6. Conclusions

A smooth motion allows a robot platform to express natural dynamics, avoiding strong transitions that force the robot and controller beyond their design limits. Thus, the use of a reference model to lead the motion of a robot resulted in strong simplifications in the motion controller’s complexity. The reference model that was used in this paper comes from a kinematics model of the plant. Thus, future studies may use more elaborate models, including aspects beyond kinematics, such as dynamics, among many other topics.

The optimization in this paper seeks the minimum arc length generated by a kinematics model of a robot. The
sigmoid function optimal for performing smooth motion was a hyperbolic tangent, normalized to reach amplitude 0.99 when the angle is \( \pi \) radians. Optimal gains for the controller were 1 for \( k_1 \) and 5 for \( k_2 \). In addition, the experiments with an actual robot platform use a sampling time of 50 ms. Subsequent studies may consider additional aspects in the optimization, such as energy consumption, elapsed time, and especially variable linear speeds, which may help to deal with centrifugal force as explained in [16].

7. References

[1] Lin C-C, Chen K-C, Chuang W-J (2012) Motion Planning Using a Memetic Evolution Algorithm for Swarm Robots. Int. J. Adv. Robot. Syst. 9:25 1-9. doi: 10.5772/45669
[2] Santos A, Sarcinelli-Filho M, Carelli R (2013) An Analytical Approach to Avoid Obstacles in Mobile Robot Navigation. Int. J. Adv. Robot. Syst. 10:278 1-13. doi: 10.5772/56613
[3] Zhang Z-H, Xiong Y-S, Fan P, Liu Y, Cheng M (2012) Real-Time Navigation of Nonholonomic Mobile Robots under Velocity Vector Control. Int. J. Adv. Robot. Syst. 9:85 1-9. doi: 10.5772/51855
[4] Martin S, Hillier N & Corke P (2010) Practical application of pseudospectral optimization to robot path planning. Australasian Conference on Robotics and Automation 2010. Australian Robotics & Automation Association. 2010 Dec. 1-3; Brisbane, Queensland. ACRA. pp. 1-9.
[5] Lamiraux F, Bonnafous D (2002) Reactive Trajectory Deformation for Nonholonomic Systems: Applications to Mobile Robots. IEEE International Conference on Robotics and Automation. 2002 May 11-15; Washington, DC, USA. ICRA. pp. 3099-3104. doi: 10.1109/ROBOT.2002.1013703.
[6] Ostfeld A (editor) (2011) Ant Colony Optimization - Methods and Applications. InTech. 3-11. doi: 10.5772/877. http://www.intechopen.com/books/ant-colony-optimization-methods-and-applications. Accessed on 15 Apr 2014.
[7] Habid D, Jamal H, Khan S (2013) Employing Multiple Unmanned Aerial Vehicles for Co-Operative Path Planning. Int. J. Adv. Robot. Syst. 10:235 1-9. doi: 10.5772/56286.
[8] García F, Guerrero A (2013) Intelligent Navigation for a Solar Powered Unmanned Underwater Vehicle. Int. J. Adv. Robot. Syst. 10:185 1-13. doi: 10.5772/56029.
[9] Kolberg E, Beiderman Y, Talyosef R (2013) Robotic Platform for Automated Search and Rescue Missions of Humans. Int. J. Adv. Robot. Syst. 10:91 1-10. doi: 10.5772/54714.
[10] Park J, Kuipers B (2011) A smooth control law for graceful motion of differential wheeled mobile robots in 2D environment. IEEE International Conference on Robotics and Automation. 2011, May 9-13; Shanghai, China. ICRA 2011. pp. 4896-4902.
[11] Petrov P (2010) Modeling and Adaptive Path Control of a Differential Drive Mobile Robot Faculty of Mechanical Engineering. Proceedings of the 12th WSEAS International Conference on Automatic Control, Modelling & Simulation. 2010, May 29-31; Catania, Sicily, Italy. ACMOS -10. pp. 403-408.
[12] Williams B, Blackmore L (2013) Probabilistic Planning for Continuous Dynamic Systems under Bounded Risk. Journal of Artificial Intelligence Research. 46 511-577.
[13] Salem F (2013) Dynamic and Kinematics Models and Control for Differential Drive Mobile Robots. International Journal of Current Engineering and Technology. 3:2 253-263.
[14] Melo L, Mangili J, Dias F, Rosario J (2010) Virtual Simulator for Design of Mobile Robot Control and Navigation Systems. In: Mobile Robots Navigation. InTech. pp. 515-535. doi: 10.5772/8980.
[15] Luo R, Lin Y, Kao C (2010) Autonomous Mobile Robot Navigation and Localization Based on Floor Plan Map Information and Sensory Fusion Approach. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems. 2010, Sep. 5-7; Fort Douglas, Salt Lake City, UT, USA. MFI 2010. pp. 121-126.
[16] Yazici A (2013) A Smooth Tour Construction Approach for a Mobile Robot with Kinematic Constraints. Int. J. Adv. Robot. Syst. 10:360 1-11. doi: 10.5772/56942.