FRONT WHEEL DRIVEN TRI-CYCLE ROBOT COURSE FOLLOWING VELOCITY COMMAND GENERATION ALGORITHM

BACKGROUND

Field of the Invention:

- The invention is an algorithm to generate velocity command that forces a front wheel drive tri-cycle robot to follow a virtual course (line or curve) in an known map.

Description of the Related Art:

- Autonomous Vehicle Navigation is one of the core functions in robotic algorithm. Robot navigation functions described in popular papers [1], [2], [3], are based on forward time simulation technology and costmap technology [4]. Such solution works perfectly for the cases where following exact course is not mandatory and robot speed steadiness is not a concern. Two issues are surfaced if applying such technology to a course following robot,
  1. the algorithm is not able to drive robot to follow course closely.
  2. the algorithm is not able to generate homogeneous velocity commands.

  Both draw-backs are due to the random nature of costmap. Such draw-backs could be critical to some of the applications, e.g. an industrial conveyor to move heavy parts from one place to another, an inspection robot to scan through each part of a room, and etc.

SUMMARY OF INVENTION

- An autonomous robot navigates itself by generating velocity commands that drives the robot base platform to follow a global course from starting point to destination. The velocity commands are generated according to the inputs of the algorithm, namely, the course to follow and the costmap[4] the robot is supposed to travel through. The output of the algorithm is the command set to drive a front wheel drive tricycle robot, including the front wheel velocity V_w (m/s) and front wheel bias angle θ (rad), which are translated from the motion command with regard to vehicle rotation center, linear transition, V_x (m/s), and angular velocity, ω (rad/s). The velocity command is set to drive vehicle rotation center to follow the given course.
- In one embodiment of the present invention, a method is provided to generate a the reference linear transition command, $\hat{V}_x$, of a front wheel drive tricycle robot, based on a given course. The method includes steps of providing the current velocity command set based on the relative position of the robot with regard to the given course. The method includes steps of analyzing geometrical features of local course, calculating reference linear transition speed $\hat{V}_x$, and reference angular velocity $\hat{\omega}$ from the fixed front wheel speed input V_w .
- In another embodiment of the present invention, a method is provided to generate the best estimation of angular velocity command, ω_{best} , of a front wheel drive tricycle robot, based the relative position of the robot with regard to the given course. The method includes steps to construct velocity sampling space with fixed reference linear transition $\hat{V}_x$ and angular velocity samples ω_s equally spaced in the range between $[\omega_{min}, \omega_{max}]$ as input, calculating possible local trajectories based on $\hat{V}_x$ and ω_s , scoring each possible trajectory against the input costmap, $\mathbb{C}$. Finally, all trajectories are sorted based on their scores and the best estimated angular velocity ω_{best} is derived.
- In another embodiment of the present invention, a method is provided to take the best estimated linear transition velocity V_{best} and best estimated anguler velocity ω_{best} as input and to generate the final wheel velocity $\hat{V}_w$ and angular position $\hat{\theta}$ commands.

BRIEF DESCRIPTION OF THE DRAWINGS

- A full understanding of the invention can be gained from the following description of the preferred embodiment when read in conjunction with the accompanying drawings in which:

  FIGS. 1-5 are flow charts showing the method and system of the invention.

DETAILED DESCRIPTION

- The method of the invention provides an efficient algorithm to compute robotic motion commands which forces the robotic platform to follow a given course *G[]*. A course is a serial of geological coordinates that describes the locations to be transversed by the robot. It may reside in any form of media, e.g. can be a file stored in a host computer, or a data array generated by other program and resided in shared memory. The algorithm takes the environment information input in the form of a costmap $\mathbb{C}$. A costmap is a discretized stochastic representation of geological features of the terrain, which has zeros cost along the given course, and ascending cost as the distance to the course increases. A costmap also incorporates the cost of possible obstacles that may cause collision along the motion. It may reside in any form of media, e.g. can be a file stored in a host computer, or a data array generated by other program and resided in shared memory. It then retrieves the wheel distance between front and rear axes, *H*. The algorithm then analyzes the geometrical features of the given course. A pair of reference velocity commands, $\hat{V}_x$ and $\hat{\omega}$, are generated accordingly. The reference linear velocity $\hat{V}_x$ is then taken as input to a forward trajectories simulation, which searches for best estimation of angular velocity command, ω_{best} within the range of $[\omega_{min}, \omega_{max}]$. The reference linear transition velocity command $\hat{V}_x$ and best estimated angular velocity command ω_{best} is then transformed to a set of actual front wheel motion commands, ie. wheel speed $\hat{V}_w$ and angular position $\hat{\theta}$.
- FIGS. 1-5 show a flow chart which illustrates the invention. Referring in particular to FIG. 1, in box 20, the algorithm starts by retrieving the global course record *G[]*, and most recent current position index, *i*. Line 21 leads to box 22 where the coordinates of three adjacent *G[]* points are inferred. Line 23 leads to box 24 where the distance Δd between the current position point *G[i]* and the next position point *G[i+1]* is calculated with the coordinate information given in box 22. After that, Line 25 leads to box 26 where the radian of arc through *G[i]*, *G[i+1]*, and *G[i+2]* is calculated. With all the preparations above, line 27 leads to box 28 where the approximate radius near the vicinity of current point is derived.
- The result is brought over via line 29 to box 30 on FIG. 2, where another set of parameters are retrieved from local media, namely the desired wheel speed V_w and the wheel axes distance *H*. Line 31 leads to box 32 where the reference linear

transition velocity $\hat{V}_x$ is calculated. Line 33 leads to box 34 where the reference angular velocity $\hat{\omega}$ is calculated.

- The result propagated through line 37 to box 40 on FIG. 3, where predefined upper and lower bound of the angular velocity is retrieved from local media. Line 31 brings the global course record *G[]*, and most recent current position index, *i*, from box 20 on FIG. 1 to box 42, where current forward course length *L* and costmap $\mathbb{C}$ are retrieved from local media. The information above was fed into box 44 via line 43 and 41, to initialize forward path simulation, where initial values of linear transition velocity is fixed $\hat{V}_x$, angular velocity is set to ω_{min}, the angular simulation granularity is set to $\Delta\omega=(\omega_{max}-\omega_{min})/N$, time granularity is Δt, and path starting point coordinate is set to $x=G[i].x$, $y=G[i].y$. The simulation starts by setting linear transition velocity to $\hat{V}_x$, and angular velocity to ω_s as specified in box 46. After this, Line 47 leads to box 48, where the simulated forward motion distance ΔL within Δt is calculated via an integration over time, and the coordinate is updated with the new coordinate at the end of Δt. Line 49 leads to box 50, where the accumulative cost of the simulated path is updated with the cost of the newly generated coordinate, $\mathbb{C}(x,y)$. The accumulated forward distance ΔL is then compared with the current forward course length *L*. If $\Delta L \le L$, line 64 leads back to box 48 and continue simulation. Otherwise, line 53 leads to another decision box 54, where the accumulative cost of the current simulated path is compared with the smallest non-negative cost path so far. In case the current simulated path is with a lower cost, line 55 leads to box 56, where the smallest non-negative cost path, $cost_{min}$, is replaced with the current simulated path, $cost(\hat{V}_x,\omega_s)$. Otherwise, line 61 joins line 57 and lead to another decision box 58, where simulated angular velocity ω_s progresses to next step and is checked against the upper bound ω_{max}. In case upper bound is not reached yet, a set of updated velocity commands is send to box 46 via line 63, so as to continue simulation. Otherwise, the result is fed to decision box 70 on FIG. 4 through line 59.
- The decision box 70 on FIG. 4 checks the validity of the final simulation result. If the $cost_{min}$ is valid, ie. $cost_{min} \ge 0$, the corresponding linear transition velocity $\hat{V}_x$ and angular velocity ω_s, are accepted as the best velocity commands, (V_{best},ω_{best}). Otherwise, $(V_{best},\omega_{best})=(0,0)$. Line 75 brings the result to the final stage of the algorithm.
- On FIG. 5, the best velocity commands are first examined by decision box 80. If linear transition velocity V_{best} is positive, line 81 leads to box 82, where the final front wheel commands are generated according to generic formulas. Otherwise, line 83 leads to decision box 84. If ω_{best} is none-zero, line 85 leads to box 86 where the final front wheel commands are generated according to special formulas. Otherwise, wheel speed $\hat{V}_w$ is set to zero and bias angle $\hat{\theta}$ is set to the last valid command $\hat{\theta}_{last}$.

CLAIMS

- The invention claimed is:

1. An algorithm to generate velocity commands that forces the robot to travel along a known course, across a known terrain.
2. The algorithm of claim 1, analyzes the course to be transversed; extract the geometrical features of the immediate course.
3. The algorithm of claim 1, produces a reference linear transition velocity based on the geometrical features of the robot and the course.
4. The algorithm of claim 1, takes the reference linear transition velocity and searches for all possible angular velocity value to find out the best angular velocity that generates a simulated trajectory with minimum non-zero cost.
5. The algorithm of claim 1, transforms the velocity commands with regard to the rotation center of a tricycle robot to the velocity commands driving the front wheel.

ABSTRACT

- The present invention provides a method to drive a robot to follow a predefined course. The algorithm specified in the invention analyzes the course to be followed as well as the geometrical dimensions of the robot, and produces the best combinations of velocity commands for the robot to follow the predefined course while maintaining constant speed on the driving wheel.

REFERENCE

[1]. Brian P. Gerkey , Kurt Konolige, “Planning and control in unstructured terrain”, In Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2008

[2]. Dieter Fox, Wolfram Burgard, Sebastian Thrun, “The dynamic window approach to collision avoidance”, pp.23-33, Robotics & Automation Magazine, IEEE (Volume:4 , Issue: 1) , Mar. 1997

[3]. Alonzo Kelly , "An Intelligent Predictive Controller for Autonomous Vehicles," tech. report CMU-RI-TR-94-20, Robotics Institute, Carnegie Mellon University, May, 1994

[4]. David V. Lu, Dave Hershberger, William D. Smart. “Layered Costmaps for Context-Sensitive Navigation”, IROS2014. Chicago, USA, September 2014