Pose estimator for single-axle tricycle with front-steered wheel and rolling rear wheels

The target of this assignment is to design a pose estimator program for a single-axle tricycle with front-steered wheel and rolling rear wheels. We are given the following data:

Coordinate system used: Right-handed

Given:

Front-wheel radius = 0.2m

Rear-wheels radius = 0.2m

Wheelbase(r) = 1m

Track(d) = 0.75m

Encoder Resolution = 512 ticks/revolution

Z constant value = 0

Pose of platform = Center of rear axis = (x, y, heading) = (0,0,0)

Given inputs from sensors:

1.) Time in seconds
2.) Steering angle in radians
3.) Encoder ticks in ticks(integer)
4.) Angular Velocity in rad/sec

Required output:

1.) X coordinate,
2.) Y coordinate and
3.) Heading angle of the platform.

If the steering wheel is set to an angle del(t) from the straight-line direction, the tricycle will rotate with angular velocity w(t) about a point (ICC, Instantaneous Center of Curvature) lying a distance R along the line perpendicular to and passing through the rear wheels.

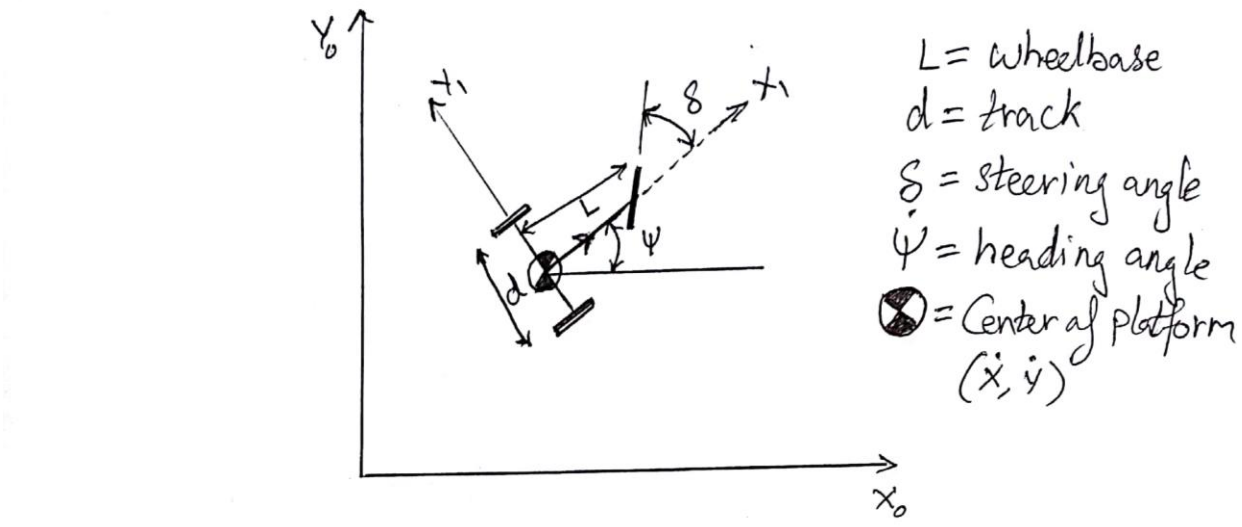
One approach is to use the ICC and the radius value from the ICC to the front and rear wheels to map the path of the wheels and find their locations based on steering angle, angular velocity and time. This is the approach discussed in most academic publications for the kinematics of a three wheeled front-steered mobile robot.

In this case, the radius of the front wheel would be more than the rear wheels and will in-turn turn more than the rear wheels, causing the platform to move along a concentric circle as the steering wheel. However, since we are not given the value of ICC or the radius values from it to the robot wheels, we have adapted an alternative approach to solve the problem. This approach only involves the variables available to us and is simple enough to implement.

The code has been tested against manual calculations for correctness and debugged to execute smoothly and exceptions have been handled as well.

Currently, the program has been built to take the sensor values as user inputs. This can be used for testing using available data. However, this can be modified to take direct sensor input as well.

The code has been written using C++ and the following approach has been used to solve the problem (**Note: The variable names used below may differ from the ones used in the code. The following is just to demonstrate the method used to solve the given problem.)**



Velocity of front wheel in the direction of travel

$$Vel\_x = \left(\frac{\text{Circumference of wheel} * \text{wheel encoder ticks}}{\text{time} * \text{encoder resolution}}\right)$$

Linear velocity of front wheel (Vel_linear) = $Vel\_x * \cos(\text{steering\_angle})$

$$\dot{\text{heading angle}} = \left(\frac{Vel\_linear}{\text{wheelbase}}\right) * \sin(\text{steering angle})$$

$$\dot{X}\_platform = back\_wheel\_radius * angular\_vel * \cos(heading\_angle)$$

$$\dot{Y}\_platform = back\_wheel\_radius * angular\_vel * \cos(heading\_angle)$$

This gives us the values of x_dot, y_dot and theta_dot. These values are then integrated with respect to time t = t_initial to t_final to obtain the x, y and heading angle values.

In order to perform this operation, Boost library has been used in C++ and Runge Kutta method order 4 has been used as the integration technique.

One of the several example values tested are:

Time = 10 seconds;

Steering angle = 0.5;

Encoder Ticks = 206;

Angular Velocity = 0.1;

Sample Output:

X coordinate, Y coordinate and Heading angle are: (0.430734, 0.106503, 0.242398)

The values have been verified using Matlab rk4 function:

https://www.mathworks.com/matlabcentral/fileexchange/29851-runge-kutta-4th-order-ode?focused=3773771&tab=function

References:

1.) Inverse kinematic models for mobile manipulators. – B.B.V.L Deepak, Nov 2012.
2.) On the Kinematic Modeling and Control of a Mobile Platform Equipped with Steering Wheels and Movable Legs. - P. Robuffo Giordano, M. Fuchs, A. Albu-Schaffer and G. Hirzinger. – ICRA09
3.) Kinematic Model of Wheeled Mobile Robots. - B B V L Deepak, Dayal R Parhi and Alok Kumar Jha. – RTET, Mar 2011.
4.) Geometric Kinematics and Applications of a Mobile Robot. - Dong-Sung Kim, Wook Hyun Kwon, and Hong Sung Park. – IJCAS Vol. 1, No. 3, Sept 2003.
5.) Turning Kinematically - Raul G. Longoria – Dept of Mech Engineering, UT, Austin, 2015.
6.) Numerical Integration in C++ using Boost Libraries (http://boccelliengineering.altervista.org/junk/boost_integration/boost_odeint.html)