BotterHeads

Aryan Rajput | Ritvik Mahajan | Sameer Talwar | Shreshth Mehrotra

Three Major Aspects of the Program:

1. Balancing

The bike comes attached with a flywheel which is used to balance it.

When tilted slightly, the bike experiences a torque due to its own weight, and falls as a result. The idea behind balancing the bike is to use the flywheel to generate a ***reaction torque*** on the bike.

For instance, if the bike falls to the right, i.e. has a clockwise angular acceleration, then we need to ensure that the **flywheel also has a clockwise angular acceleration.** This essentially means that the torque on the flywheel is clockwise, which implies that there is an anticlockwise reaction torque on the bike (good ol' Newton's 3rd law), which tries to counteract the original, clockwise torque.

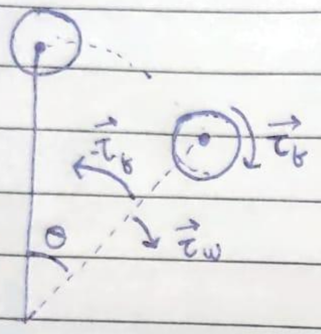
If we model the system as an inverted pendulum, then the torque due to its weight can be written as $ksin(\theta)$, where θ is the angle from the vertical. Further, this can be written as $k\theta$, owing to the small angle approximation for $sin(\theta)$. (Assuming we will eventually control the bike well enough initially for the approximation to hold good).The constant k depends on the geometry and mass of the bike.

The constant k can be approximated by letting the bike simply fall over (upto small values of θ) and observing $\theta(t)$, as here it simply follows the equation:

$$I\frac{d^2\theta}{dt^2} = k\theta$$

We then try to calculate the appropriate reaction torque,using PD control. We can form the DE of motion for the bike,with the constraint that the solution for $\theta(t)$ should be **critically damped,**which ensures that θ and $d\theta/dt$ (angular velocity), both are driven to zero quickly, without oscillation.

The value for k_p was chosen to be slightly greater than k for convenience, and k_d was calculated using the condition for critical damping.



Reaction

$$\uparrow$$

$$I_b \frac{d^2\theta}{dt^2} = \left(-\vec{\tau}_f + \vec{\tau}_w\right)$$

$$I_b \frac{d^2\theta}{dt^2} = K\theta - \left(K_p\theta + K_d\frac{d\theta}{dt}\right)$$

$$\boxed{I_b\ddot{\theta} + K_d\dot{\theta} + (K_p - K)\theta = 0}$$

For critically damped solution: $\boxed{K_d^2 = 4I_b(K_p - K)}$.

$$\tau_f = K_p\theta + K_d\dot{\theta}$$

$$\alpha_f = \tau_f / I_f = \frac{d\omega_f}{dt}$$

$$\boxed{\omega_f \mathrel{+}= \alpha_f\,dt} \text{ , so } \boxed{\omega_f \mathrel{+}= \frac{\tau_f}{I_f}dt = \left(\frac{K_p\theta + K_d\dot{\theta}}{I_f}\right)dt}$$

controlling Flywheel angular velocity to generate the appropriate Flywheel (and hence, reaction) Torque..

2. Go-to-goal Navigation (GTG)

This aspect required navigating the bike to a destination whose coordinates are known.
Thus, it was achieved simply by aligning the bike to the goal vector (a vector drawn keeping the bike as origin and the goal coordinates as the end points).
It further involved two steps, namely-

- ➔Driving the bike forward (i.e. publishing some velocity to the rear wheel)
- ➔Turning the handle, and in turn the bike, towards the goal (i.e. publishing angular velocity to the handle joint actuator accordingly)

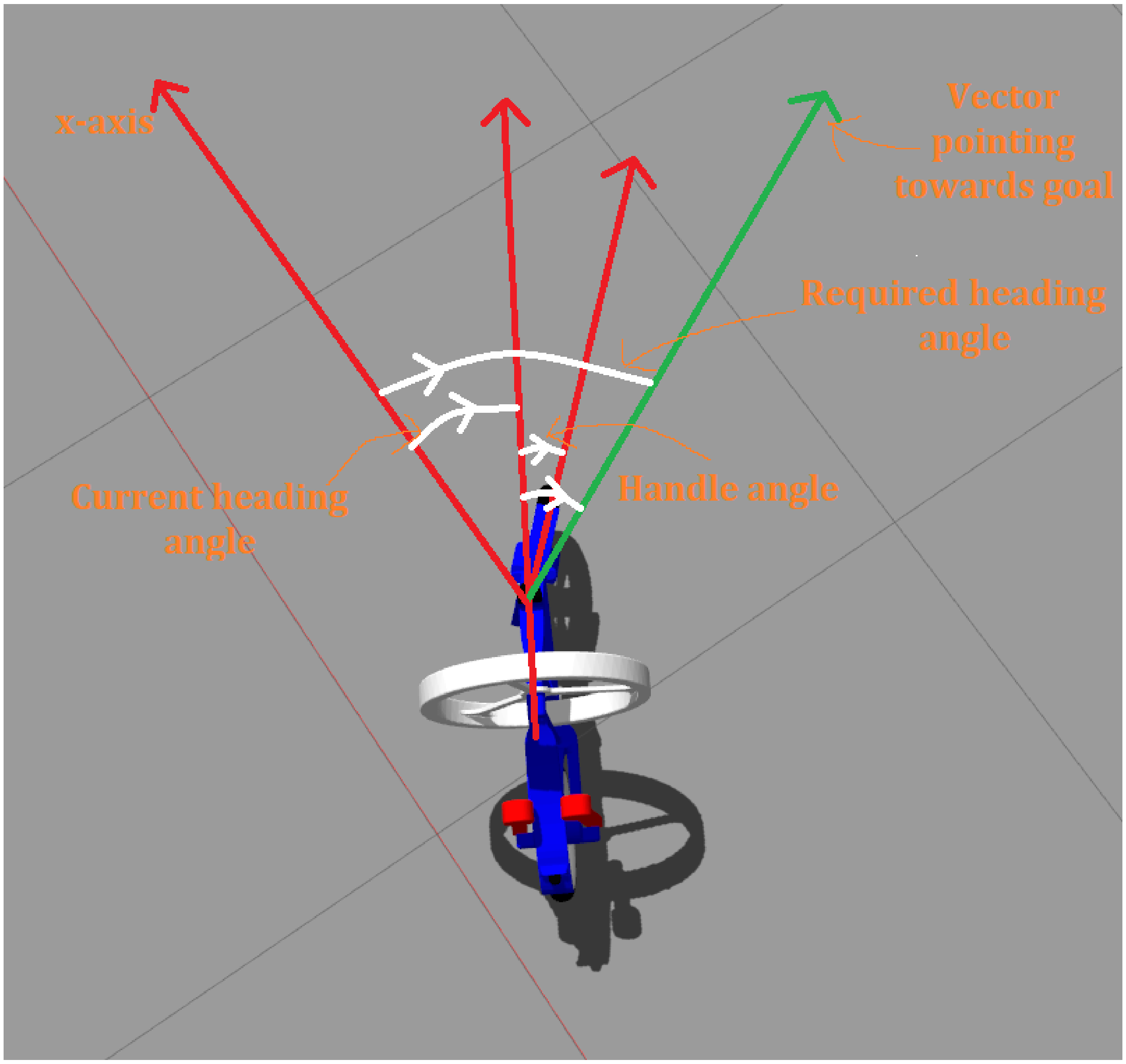
a. Driving the bike forward

PID was implemented to control the rear wheel angular velocity (in turn controlling the forward linear velocity of the bike) by treating the distance between current position of the bike and the goal position as the error term(attempting to minimise it).

$$e = \sqrt{(x_g - x)^2 + (y_g - y)^2}$$

$$\omega_{rearwheel} = k_p e + k_d \frac{de}{dt} + k_i \int e dt$$

b. Bike turning and handle control



Different angles used for controlling:

- The **current heading angle** or the **bike angle** is the current pose of the bike, i.e. the angle its own axis makes with the x-axis. If the handle angle is 0, the bike will move in this direction.

- **Required heading angle** is the angle between the x-axis and the vector pointing towards the goal, i.e. the angle along which the bike should head to, to reach the goal.

- **Handle angle** is the position of the handle with respect to the axis of the bike.

A Proportional controller is applied to control the handle angular velocity with the error term being the angle between the bike and the goal, i.e.

$$error(e) = \theta_{req} - \theta_{curr}$$
$$\omega_{handle} = k_p e$$

This error is minimised by publishing appropriate values of angular speed to the handle actuator.

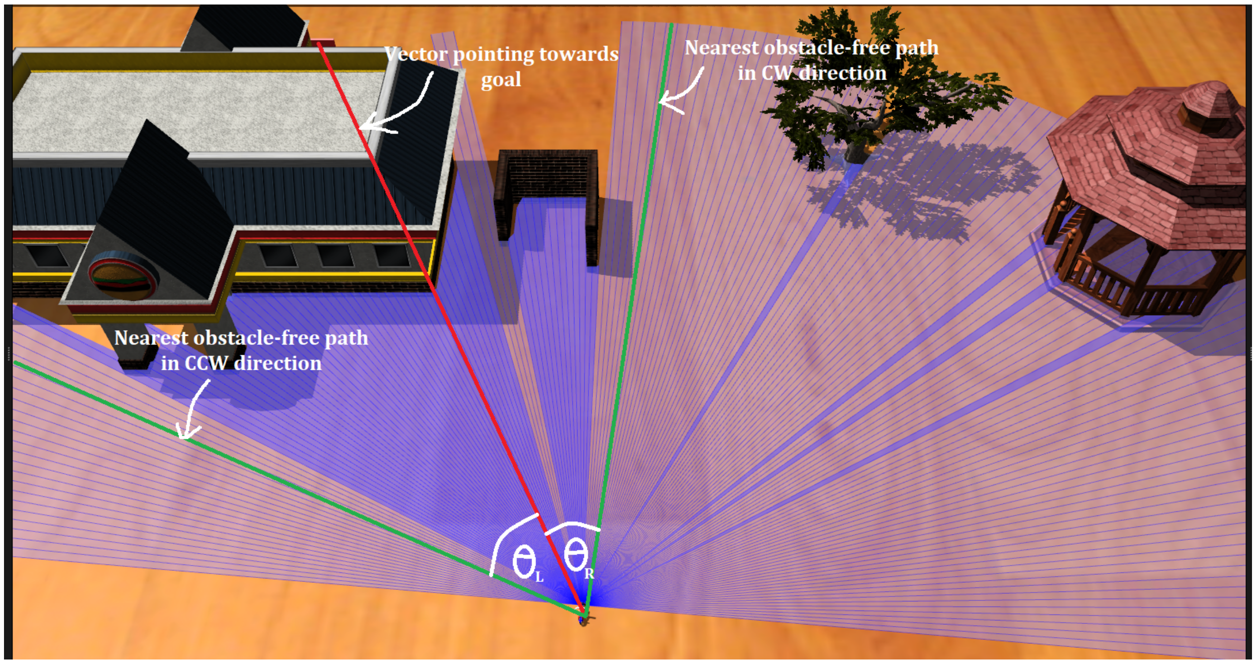
Once the bike aligns with the goal vector, a second P controller comes into play. It aims to reduce the angle between the bike axis and handle to 0 (this time, the handle angle is taken as the error term). This is done to avoid oscillations / overshooting in the bike's journey.

$$\omega_{handle} = k_p e = k_p \theta_{handle}$$

The combination of the aforementioned two steps (basically, combination of three PID controls for two segments treated separately, forward movement and turning) aligns the bike along the goal vector, while simultaneously minimizing the distance between the goal and the bike, which ultimately drives the bike towards the goal.

3. Path planning with obstacle detection and avoidance

As is the case with our lives, the bike also faces many obstacles in its journey to its final destination.


Vector pointing towards goal
Nearest obstacle-free path in CW direction
Nearest obstacle-free path in CCW direction
θ_L
θ_R

The LiDAR was configured to return 180 sample points

The LiDAR attached to the bike returns 180 values, corresponding to 180 °, with each value indicating the distance between the bike and the nearest obstacle in that sector (If no obstacles are detected, the value returned is *inf* , which was changed to 30,the max range of the LiDAR).

The required heading angle, without considering any obstacles (henceforth ideal path), is calculated as shown in section 2. If the 10° wide sector centered around this required heading angle is free of any obstacles (to a certain threshold), the bike follows the GTG behaviour explained in section 2.

However, if some obstacles are encountered in this path, the bike begins its search for the next best path. This is done by searching for the nearest (at least) 10° wide sector which is free of obstacles in both CW (right) and CCW (left) directions. The sector which is closest to the ideal path is chosen (In the image shown above, since $\theta_R < \theta_L$, the bike takes the right path(No pun intended)). After calculating the required path, the control of the bike is done through the algorithms mentioned in section 2.

Upon finally reaching the goal:

The obstacle avoidance behaviour is suspended and the bike follows just GTG behaviour once it reaches near the goal. This is done since the goal location (parking space) in the problem statement is surrounded by walls (obstacles) which may hinder the bike's approach towards the goal.

After reaching the goal, the bike switches to “Eat 5 Star, do nothing” mode: the bike along with the flywheel come to a gradual stop, aided by the stands.

Note:

- The script may not run properly in the 1st code run due to errors / glitches in Gazebo which the human mind cannot comprehend.
- Mutual fund investments are subject to market risks, read all scheme related documents carefully.

***