## 11.4. Odometry (from Volume 3, Embedded Systems: Real-time OS)

**Odometry** is a method to predict position from wheel rotations. We assume the wheels do not slip along the ground. If one wheel moves but the other does not, it will rotate about a single contact point of the wheel to the ground. If one wheel moves more than the other, then there will be both a motion and a rotation about a point somewhere along line defined by the axle connecting the two wheels. We define the robot center of gravity (cog) as a point equidistant from the pivot points. The robot position is defined as the (*x*,*y*) location and the compass direction, or yaw angle *θ*, of the cog. See Figure 11.10.



Figure 11.10. A robot with two drive wheels is defined by the wheel base and wheel diameter.

**Constants**

Number of slots/rotation, *n*=360

Wheel diameter, *d*= 70000 (0.0001cm)

 

Figure 11.11. To measure wheel motion we used an encoder on each wheel.

Wheelbase (distance between wheels), *w* = 140000 (0.0001cm)

Wheel circumference, *c* = π*d* = 219911 (0.0001cm)

**When to run odometry**

You should not run the calculations on each count.

*Option 1:* Run the odometry calculations at fixed rate. Choose a rate so the counts are about 20. If the counts go above 40, you could increase rate. If the counts drop below 10, you could decrease rate.

*Option 2:* Run the odometry when either count gets to a fixed limit, e.g., 20.

**Measurements**

*LCount* the number of slots of left wheel since the last calculation. *RCount* the number of slots of right wheel. At 166 RPM, there will be 10 counts in 10 ms. Some simple cases are found in Table 11.3, where *m* is any number from ‑K to +K.

|  |  |  |
| --- | --- | --- |
| ***LCount*** | ***RCount*** | *Motion* |
| *m* | *m* | straight line motion in the current direction |
| 0 | *m* | pivot about stopped left motor |
| *m* | 0 | pivot about stopped right motor |
| *m* | *-m* | pure rotation about cog |

Table 11.3. Example measurements, relationship between counts and motion.

**Derivations**

*Lr* = *LCount* **\****c/n* the arc distance traveled by the left wheel (0.0001cm)

*Rr = RCount\*c/n* the arc distance traveled by the right wheel (0.0001cm)

 

Figure 11.12. Motions occurring during a left turn.

Using similar triangles, we can find the new pivot point. Assuming *Rr* and *Lr* are both positive and *Rr*>*Lr*, we get

*L/Lr = (L+w)/Rr*

*L/Lr - L/Rr = w/Rr*

*L Rr - L Lr = w Lr*

*L = w Lr/(Rr - Lr)*

Notice also the change in yaw, dθ, is the same angle as the sector created by the change in axle. The change in angle is

**d**θ = **Lr**/**L** = Rr/(L+w)

We can divide the change in position into two components



Figure 11.13. Geometry of a left turn.

The exact calculation for position change is

*dz = (L+w/*2)\*tan(*dθ/*2)

but if dθ is small, we can approximate *dz* by the arc length.

*dz* = *dθ*/2\*(*L+w*/2)

**Initialize**

We initialize the system by specifying the initial position and yaw.

(*x, y, θ*) (0.0001cm, 0.0001cm, 2π/16384 radian)

**Calculations** (run this periodically, measuring *LCount* *RCount*)

*Lr* = *LCount* **\****c/n* (0.0001cm)

*Rr* = *RCount* **\****c/n* (0.0001cm)

*L = (w\*Lr)/(Rr - Lr)* (0.0001cm)

*dθ* = ((16384/2π)\**Lr***)**/*L*(2π/16384 radian)

*dz* = ((*dθ*/2)\*(*L+w*/2))/(16384/2π) (0.0001cm) *approximation*

or *dz* = (tan(*dθ*/2)\*(*L+w*/2))/1000 (0.0001cm) *more accurate*

*x = x + dz*\*cos(*θ*) (0.0001cm)

*y = y+ dz*\*sin(*θ*) (0.0001cm) *first part of move*

*θ = θ + dθ*(2π/16384 radian)

*x = x + dz*\*cos(*θ*) (0.0001cm)

*y = y+ dz*\*sin(*θ*) (0.0001cm) *second part of move*

**Special case** *LCount* equals *RCount*

**Special case** *LCount* equals 0, assume *RCount* < *n*



*Lr* = 0 (0.0001cm)

*Rr = RCount\*c/n* the arc distance traveled by the right wheel (0.001cm)

*L =* 0 (0.0001cm)

*dθ* = 2π *Rr* /2π*w*(radian)

*dθ* = *Rr /w* (radian)

*dθ* = (16384/2π) *Rr* /*w*(2π/16384 radian)

*dθ* = (2608*\*Rr+w/2)*/*w*(2π/16384 radian)



*dz* = ((*dθ*/2)\* *w*/2)/ (16384/2π) (0.0001cm) *approximation*

*dz* = (*dθ*\**w*+5215)/10430 (0.0001cm) *approximation*

or *dz* = (tan(*dθ*/2)\* *w*/2) (0.0001cm) *more accurate*

*x = x + dz*\*cos(*θ*) (0.0001cm)

*y = y+ dz*\*sin(*θ*) (0.0001cm) *first part of move*

*θ = θ + dθ*(2π/16384 radian)

*x = x + dz*\*cos(*θ*) (0.0001cm)

*y = y+ dz*\*sin(*θ*) (0.0001cm) *second part of move*