Swerve Drive

We consider robots equipped with four wheels which can be indepedently driven and steered. This provides a holonomic drivetrain which presents a unique control problem. We seek to answer the question, 'how do we drive the individual wheels to achieve a desired net linear or angular velocity?' Assuming an ideal system, inverse kinematics can be derived to address this problem.

Robot Model

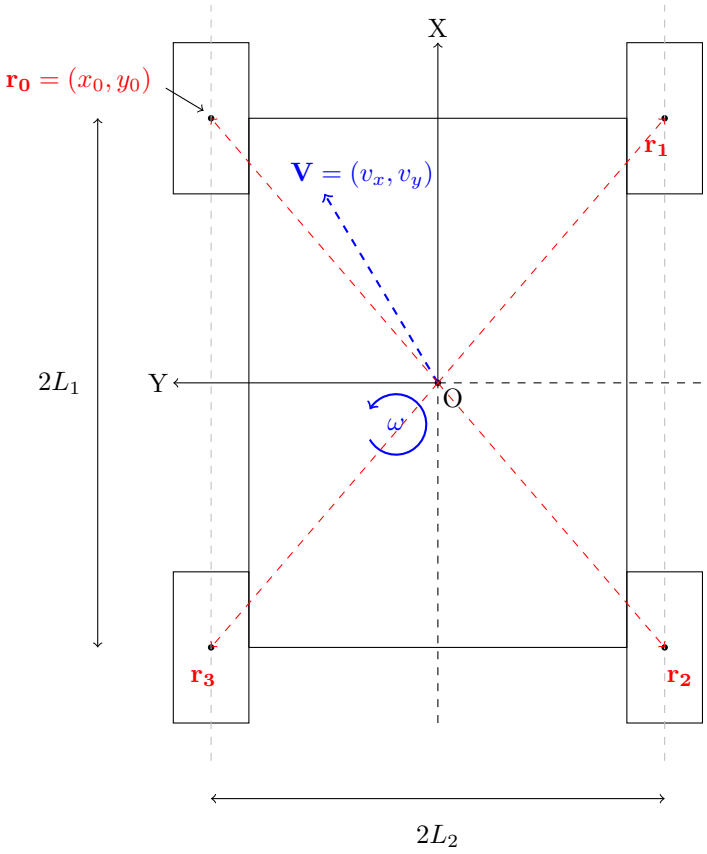

Figure 1: Robot Model

Consistent with ROS coordinate system conventions we define a right-handed coordinate system with the X axis as 'forward' and the Y axis as 'strafe' in a leftwards direction, from a birds-eye view of the robot. The Z axis extends out of the page, and angles increase counter-clockwise. The origin O is the geometric

centre of the robot.

Each wheel is mounted at $\mathbf{r}_n = (x_n, y_n)$ where $\mathbf{r}_n$ is the displacement vector from the centre of rotation (COR) of the robot to the wheel. In this diagram, the centre of rotation is also the geometric centre of the robot, although it doesn't have to be.

$\mathbf{V} = (v_x, v_y)$ is the (desired) net linear velocity of the robot at the COR, and ω is the (desired) angular velocity about the COR.

Wheel Model

Consider one of the four wheels.

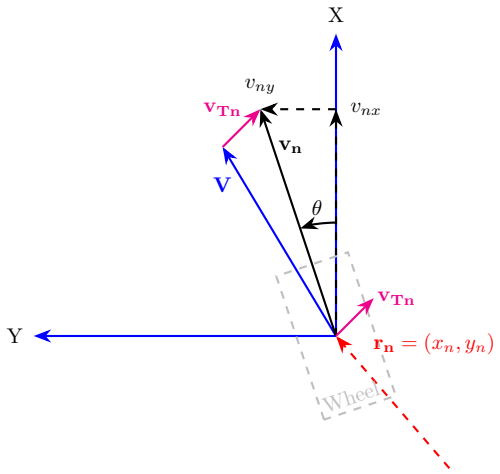

Figure 2: Wheel Model

The wheel is inclined at an angle θ from the X axis, and is travelling at net velocity $\mathbf{v}_n$, which consists of the net linear velocity $\mathbf{V}$ and a tangential velocity component $\mathbf{v}_{Tn}$ provided by the angular velocity of the robot ω.

In three dimensions, the tangential velocity is calculated by $\mathbf{v}_{Tn} = \boldsymbol{\omega} \times \mathbf{r}_n$, so

$$
\begin{aligned}
\mathbf{v}_{Tn} &= \boldsymbol{\omega} \times \mathbf{r}_n \\
&= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 0 & \omega \\ x_n & y_n & 0 \end{vmatrix} \\
&= \mathbf{i} \begin{vmatrix} 0 & \omega \\ y_n & 0 \end{vmatrix} - \mathbf{j} \begin{vmatrix} 0 & \omega \\ x_n & 0 \end{vmatrix} + \mathbf{k} \begin{vmatrix} 0 & 0 \\ x_n & y_n \end{vmatrix} \\
&= (-\omega y_n, \omega x_n, 0)
\end{aligned}
$$

which is equivalent to $\mathbf{v}_{Tn} = (-\omega y_n, \omega x_n)$ in two dimensions.

The wheel has net velocity $\mathbf{v}_n = \mathbf{V} + \mathbf{v}_{Tn} = (v_x, v_y) + (-\omega y_n, \omega x_n) = (v_x - \omega y_n, v_y + \omega x_n)$.

For all four wheel velocities, the control system can then be represented in matrix form as

$$\begin{bmatrix} v_{0x} & v_{0y} \\ v_{1x} & v_{1y} \\ v_{2x} & v_{2y} \\ v_{3x} & v_{3y} \end{bmatrix} = \begin{bmatrix} 1 & -y_0 & x_0 \\ 1 & -y_1 & x_1 \\ 1 & -y_2 & x_2 \\ 1 & -y_3 & x_3 \end{bmatrix} \begin{bmatrix} v_x & v_y \\ \omega & 0 \\ 0 & \omega \end{bmatrix}$$

Equivalently $V = BU$ where V is the matrix of outputs, B is the control (coefficient) matrix, and U is the matrix of inputs.

These are the X- and Y-components of the wheel velocity vectors, so the angle at which to drive wheel n is given by $\theta_n = \arctan2(v_{ny}, v_{nx})$ where arctan2 is a quadrant-aware arctangent function, and the linear speed of each wheel is given by $v_n = \sqrt{v_{nx}^2 + v_{ny}^2}$. This could be divided by the wheel radius to find the rotational speed of each wheel.

If the COR is at the geometric centre and origin of a rectangular drivetrain with wheel separations $2L_1$ and $2L_2$ as in the Robot Model diagram, then the model parameters are as follows.

n	$\mathbf{r}_n = (x_n, y_n)$
0	(L_1, L_2)
1	$(L_1, -L_2)$
2	$(-L_1, -L_2)$
3	$(-L_1, L_2)$

The control matrix becomes

$$B = \begin{bmatrix} 1 & -L_2 & L_1 \\ 1 & L_2 & L_1 \\ 1 & L_2 & -L_1 \\ 1 & -L_2 & -L_1 \end{bmatrix}$$

Suppose the wheels can individually be driven at a maximum speed of v_{max}, meaning the maximum value of either v_x or v_y is also v_{max}. Maximum angular velocity occurs when each wheel is angled 45 degrees and driven at v_{max} in the same direction, giving a $\omega_{max} = \frac{v_{max}}{r}$ where $r = \sqrt{L_1^2 + L_2^2} = |\mathbf{r}_n|$. The inputs can therefore be expressed as proportions of these maximum values. Let k_x, k_y, k_ω be between -1 and 1 such that $v_x = k_x v_{max}$, $v_y = k_y v_{max}$, and $\omega = k_\omega \omega_{max}$. We can use these to show that

$$V = v_{max} \begin{bmatrix} 1 & -\frac{L_2}{r} & \frac{L_1}{r} \\ 1 & \frac{L_2}{r} & \frac{L_1}{r} \\ 1 & \frac{L_2}{r} & -\frac{L_1}{r} \\ 1 & -\frac{L_2}{r} & -\frac{L_1}{r} \end{bmatrix} \begin{bmatrix} k_x & k_y \\ k_\omega & 0 \\ 0 & k_\omega \end{bmatrix}$$

Since $r = \sqrt{L_1^2 + L_2^2}$, $\frac{L_2}{r}$ and $\frac{L_1}{r}$ are simply the ratios of the sides of a right-angled triangle with side lengths L_2 and L_1 with its hypotenuse.

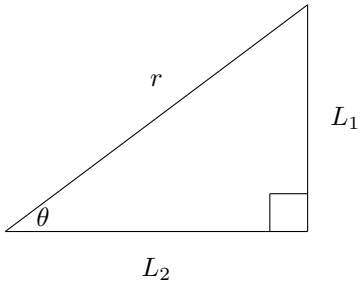

Figure 3: Triangle

Let $\theta = \arctan \frac{L_1}{L_2}$ as in the diagram. Then

$$V = v_{max} \begin{bmatrix} 1 & -\cos\theta & \sin\theta \\ 1 & \cos\theta & \sin\theta \\ 1 & \cos\theta & -\sin\theta \\ 1 & -\cos\theta & -\sin\theta \end{bmatrix} \begin{bmatrix} k_x & k_y \\ k_\omega & 0 \\ 0 & k_\omega \end{bmatrix}$$

We can rewrite $V = \sin\Theta \cdot U = v_{max} \sin\Theta \cdot K$ where

$$\Theta = \begin{bmatrix} \frac{\pi}{2} & \theta - \frac{\pi}{2} & \theta \\ \frac{\pi}{2} & \frac{\pi}{2} - \theta & \theta \\ \frac{\pi}{2} & \frac{\pi}{2} - \theta & -\theta \\ \frac{\pi}{2} & \theta - \frac{\pi}{2} & -\theta \end{bmatrix}$$

and

$$K = \begin{bmatrix} k_x & k_y \\ k_\omega & 0 \\ 0 & k_\omega \end{bmatrix}$$

So we have managed to characterise the model in terms of a single constant θ. We can eliminate v_{max} by dividing out to get $P = \sin\Theta \cdot K$ where

$$P = \begin{bmatrix} p_{0x} & p_{0y} \\ p_{1x} & p_{1y} \\ p_{2x} & p_{2y} \\ p_{3x} & p_{3y} \end{bmatrix}$$

and $p_n \in [-1, 1]$ is the proportion of 'effort' applied to the wheel, with $|p_n| = 1$ corresponding to maximum speed.

We have derived an ideal model whose only physical dependency is the wheel locations.

Application

Let us redefine the input and output matrices to use complex numbers.

$$\mathbf{p} = \begin{bmatrix} p_{0x} + ip_{0y} \\ p_{1x} + ip_{1y} \\ p_{2x} + ip_{2y} \\ p_{3x} + ip_{3y} \end{bmatrix}$$

$$\mathbf{k} = \begin{bmatrix} k_x + ik_y \\ k_\omega + i0 \\ 0 + ik_\omega \end{bmatrix}$$

These are now complex vectors instead of matrices, and we have $\mathbf{p} = \sin\Theta \cdot \mathbf{k}$. The output calculations are simplified to $\theta_n = \arg(p_{nx} + ip_{ny})$ and $p_n = |p_{nx} + ip_{ny}|$. These are much simpler to code using e.g. `numpy` and we get useful characteristics such as quadrant-aware, bounded arctan for free.

Non-ideal Characteristics

Assuming an ideal model provides freedoms such as instantaneously changing the wheel angle, however in practice this is not possible. In particular, consider driving forward at full-speed $(k_x = 1, k_y = 0, k_\omega = 0) \rightarrow (p_n = 1, \theta_n = 0)$, and then immediately reversing at full speed $(p_n = 1, \theta_n = \pi)$. According to this model, we need to turn the wheels 90 degrees to reverse, whereas we can simply drive them in the opposite direction without changing the angle. This is a limitation as p_n is a positive magnitude, while negative p_n indicates driving a wheel in reverse.

Consider a wheel with current velocity v and desired velocity v' as in the following diagram.

The angle ϕ between v and v' is greater than 90 degrees, but the angle between v and $-v'$ is much smaller. In this case, it will be faster to turn the wheel to the angle of $-v'$ and drive it in reverse instead of turning to v' and driving it forward.

In general, $-v'$ will be closer to v if the angle between v and v' is greater than 90 degrees. A simple test for this is to test whether the dot product $v \cdot v'$ is negative (this comes from the formula for cosine of the angle between two vectors).

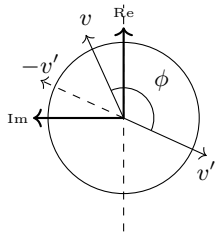

Figure 4: Vectors

Thus, we can change the sign of p_n and the angle of θ_n based on the sign of $(-p'_{nx} - ip'_{ny}) \cdot (p_{nx} + ip_{ny}) = -p'_{nx}p_{nx} - p'_{ny}p_{ny}$ (complex dot product) and the angle of $-p'_{nx} - ip'_{ny}$.

Note if the current or future velocity is 0, we can just compare on the angular positions of the wheels instead of the speeds.

Further Development

This is meant to be a simple, generalised, first-principles model for controlling the drivetrain. I hope it provides a suitable base from which to build more advanced concepts such as traction control.