Comparison of Vehicle Models for Explicit Steering and Skid Steering

Brandon Luders

September 1, 2009

This documentation summarizes the work done in developing vehicle models using both explicit steering (DARPA Grand Challenge car [1], Agile Robotics forklift) and skid steering (Agile Robotics rover, iRobot). Section 1 details the vehicle model for explicit steering, while Section 2 details the vehicle model for skid steering. Section 3 provides additional information which different readers may find useful:

- *RRTeam members:* Section 3.1 lists the measured values of vehicle parameters for the iRobot, gives linearized vehicle dynamics, and summarizes the rules of the ACL Robotics Challenge.
- *Vishnu:* Section 3.2 outlines the tasks moving forward in controlling the Pioneer rovers, including driving a skid-steered vehicle with pure pursuit and experimentally measuring the vehicle model parameters.

Note that Ref. [2] considers a similar comparison of vehicle steering models.

1 Explicit Steering

In the explicit steering model, there are separate, decoupled inputs for speed control and steering control. Steering is controlled by physically rotating one or both of the vehicle axes; all wheels move at the same speed.

Figure 1 shows a diagram of the explicit steering vehicle model. In this figure, x and y are the coordinate directions; θ is the heading, measured counterclockwise from the $+x$ axis; δ is the steering angle, measured counterclockwise from θ; v is the vehicle speed; and L is the characteristic length, here the distance between the center of the front and rear axles.

The simplest system model for this vehicle is

$$\dot{x} = v\cos\theta, \tag{1}$$

$$\dot{y} = v\sin\theta, \tag{2}$$

$$\dot{\theta} = \frac{v}{L}\tan\delta. \tag{3}$$

In this case, the state vector is (x, y, θ), while the input vector is (v, δ).

Often, the vehicle speed is controlled indirectly, by appling some acceleration force.

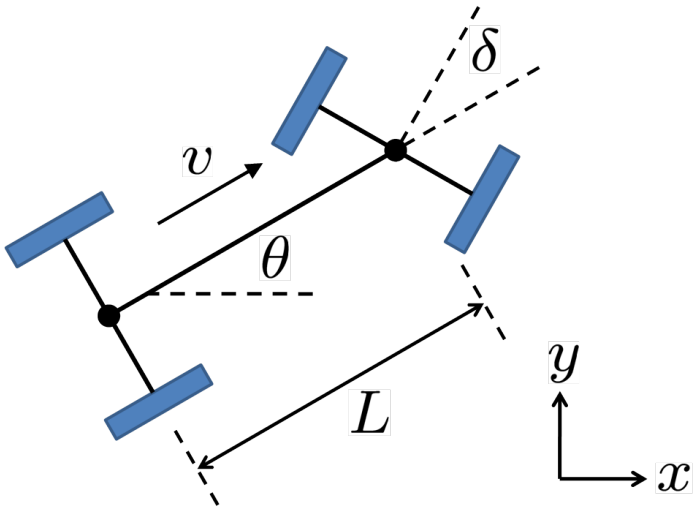

Figure 1: Explicit steering model for a wheeled vehicle.

Then the system model takes the form

$$\dot{x} = v\cos\theta, \tag{4}$$
$$\dot{y} = v\sin\theta, \tag{5}$$
$$\dot{\theta} = \frac{v}{L}\tan\delta, \tag{6}$$
$$\dot{v} = a, \tag{7}$$

where a is the vehicle acceleration. In this case, the state vector is (x, y, θ, v), while the input vector is (δ, a).

Furthermore, it is typically the case that the acceleration and steering inputs cannot be applied instantaneously, and will have some delay before the desired values are achieved. This can be modeled by adding first-order reference dynamics to the system, such that

$$\dot{x} = v\cos\theta, \tag{8}$$
$$\dot{y} = v\sin\theta, \tag{9}$$
$$\dot{\theta} = \frac{v}{L}\tan\delta, \tag{10}$$
$$\dot{\delta} = \frac{1}{T_\delta}(\delta_R - \delta), \tag{11}$$
$$\dot{v} = a, \tag{12}$$
$$\dot{a} = \frac{1}{T_a}(a_R - a), \tag{13}$$

where δ_R is the reference steering angle, a_R is the reference acceleration, and T_δ and T_a are their respective time constants. In this case, the state vector is $(x, y, \theta, \delta, v, a)$, while the input vector is (δ_R, a_R).

The model for the DARPA Grand Challenge [1] and Agile Robotics vehicles adds an additional sideslip term into (3),

$$\dot{\theta} = \frac{v}{L}\tan\delta\frac{1}{1+(v/v_{CH})^2}, \tag{14}$$

Table 1: DGC Vehicle Parameters

Parameter	Name	Value
L	Characteristic length	2.855 m
v_{CH}	Characteristic velocity	20.0 m/s
T_δ	Steering time constant	0.05 s
T_a	Acceleration time constant	0.3 s
$a_{\min}$	Minimum acceleration	-6.0 m/s^2
$a_{\max}$	Maximum acceleration	1.8 m/s^2
$\delta_{\max}$	Maximum steering angle	0.5435 rad
$\dot{\delta}_{\max}$	Maximum slew rate	0.3294 rad/s

where v_{CH} is termed the characteristic velocity.

The constraints on the vehicle are as follows:

$$a_{\min} \leq a \leq a_{\max}, \tag{15}$$

$$\|\delta\| \leq \delta_{\max}, \tag{16}$$

$$\|\dot{\delta}\| \leq \dot{\delta}_{\max}, \tag{17}$$

where $a_{\min}$ is the minimum acceleration (which may be negative), $a_{\max}$ is the maximum acceleration, $\delta_{\max}$ is the maximum steering angle, and $\dot{\delta}_{\max}$ is the maximum slew rate.

Finally, Table 1 lists the values of the parameters used on the DGC vehicle [1].

2 Skid Steering Model

In the skid steering model, the inputs to the vehicle are exclusively the speed of the wheels on each side of the vehicle. Unlike in the explicit steering model, the wheel speeds on each side of the vehicle are allowed to take different values, and may even be moving in opposite directions. This gives the vehicle capabilities not achievable with explicit steering, such as being able to rotate in place (zero turning radius). If the vehicle has more than 1 wheel on each side, all wheels on the same side of the vehicle are linked to the same speed. However, neither axis is ever rotated; the position of all wheels is fixed. Thus, skidding is actually necessary for the vehicle to turn (hence the name).

There are two ways to represent the speed inputs for the skid steering model. The most intuitive way is to treat the speed on each side as a separate input. In this case, the input vector is (v_L, v_R), where v_L is the speed of the left-side wheels, and v_R is the speed of the right-side wheels. An alternative formulation, useful for deriving the vehicle model, uses the input vector $(v, \Delta v)$, where v is the *mean* wheel speed and Δv is the wheel speed *differential*. The relationship between these two input formulations is

$$v_L = v - \frac{1}{2}\Delta v, \tag{18}$$

$$v_R = v + \frac{1}{2}\Delta v. \tag{19}$$

Again, note that v_L and/or v_R may take on negative values, within whatever constraints are in place.

Figure 2 shows a diagram of the skid steering vehicle model. In this figure, x and y are the coordinate directions; θ is the heading, measured counterclockwise from the $+x$ axis; v is the mean wheel speed; v_L and v_R are the left wheel and right wheel speeds, respectively; and L is the characteristic length, here the length of the wheel axle. If the vehicle is performing a turn, then let R denote the turning radius and $\dot{\theta}$ denote the turn rate, which is positive in the counterclockwise direction.

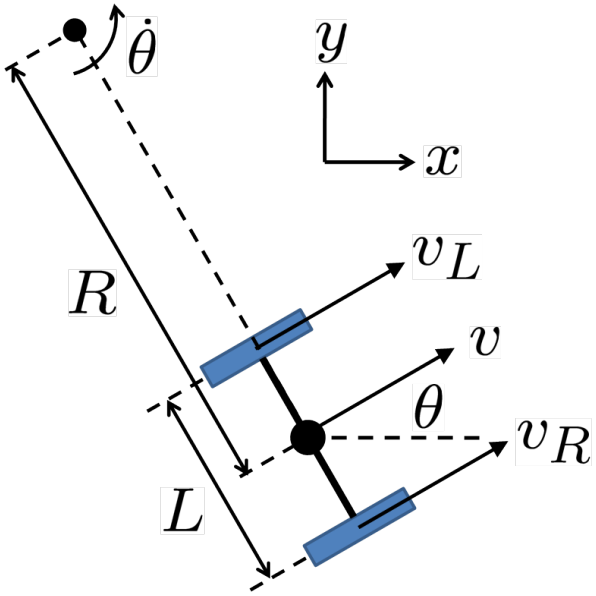

Figure 2: Skid steering model for a wheeled vehicle.

The skid steering state-space model is now derived. The translational dynamics are the same as (1)-(2) for the explicit steering model,

$$\dot{x} = v\cos\theta, \tag{20}$$
$$\dot{y} = v\sin\theta. \tag{21}$$

To derive the steering dynamics, we apply the relationship between linear and angular velocity to both sets of wheels and the axle center [3]:

$$v_L = \left(R-\frac{1}{2}L\right)\dot{\theta}, \tag{22}$$
$$v_R = \left(R+\frac{1}{2}L\right)\dot{\theta}, \tag{23}$$
$$v = R\dot{\theta}. \tag{24}$$

Combine (22)-(23) and simplify:

$$\begin{aligned} \frac{v_L}{R-\frac{1}{2}L} &= \frac{v_R}{R+\frac{1}{2}L} \\ v_L\left(R+\frac{1}{2}L\right) &= v_R\left(R-\frac{1}{2}L\right) \\ \frac{1}{2}L(v_L+v_R) &= R(v_R-v_L) \end{aligned}$$

Applying (18)-(19) to this last equation yields

$$Lv = R(v_R - v_L)$$

Finally, insert (24) to arrive at the desired relationship,

$$\dot{\theta} = \frac{\Delta v}{L}. \tag{25}$$

In summary, by combining (20)-(21) and (25) the simplest system model for a skid steering vehicle is

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= \frac{\Delta v}{L}. \end{aligned}$$

In this case, the state vector is (x, y, θ), while the input vector is $(v, \Delta v)$. This can also be written using the input vector (v_L, v_R) as

$$\dot{x} = \frac{1}{2}v_R\cos\theta + \frac{1}{2}v_L\cos\theta, \tag{26}$$

$$\dot{y} = \frac{1}{2}v_R\sin\theta + \frac{1}{2}v_L\sin\theta, \tag{27}$$

$$\dot{\theta} = \frac{1}{L}v_R - \frac{1}{L}v_L. \tag{28}$$

Similar to before, it is typically the case that the speed inputs cannot be applied instantaneously, and will have some delay before the desired values are achieved. This can be modeled by adding first-order reference dynamics to the system, such that

$$\dot{x} = \frac{1}{2}v_R\cos\theta + \frac{1}{2}v_L\cos\theta, \tag{29}$$

$$\dot{y} = \frac{1}{2}v_R\sin\theta + \frac{1}{2}v_L\sin\theta, \tag{30}$$

$$\dot{\theta} = \frac{1}{L}v_R - \frac{1}{L}v_L, \tag{31}$$

$$\dot{v_R} = \frac{1}{T_v}v_R^{ref} - \frac{1}{T_v}v_R, \tag{32}$$

$$\dot{v_L} = \frac{1}{T_v}v_L^{ref} - \frac{1}{T_v}v_L, \tag{33}$$

where v_R^{ref} and v_L^{ref} are the reference right and left wheel speeds, respectively, and T_v is the speed time constant (assumed due to the symmetry to be the same for all wheels). In this case, the state vector is (x, y, θ, v_R, v_L), while the input vector is (v_R^{ref}, v_L^{ref}). Similar dynamics can be derived using $(v, \Delta v)$, but it is not as intuitive.

The constraints on the vehicle are as follows:

$$v_{\min} \leq v_L \leq v_{\max}, \tag{34}$$

$$v_{\min} \leq v_R \leq v_{\max}, \tag{35}$$

where $v_{\min}$ is the minmum wheel speed (which may be negative) and $v_{\max}$ is the maximum wheel speed, both of which are symmetric for both sets of wheels.

Table 2: iRobot Vehicle Parameters

Parameter	Name	Value
L	Characteristic length	0.262 m
T_v	Speed time constant	0.025 s
$v_{\min}$	Minimum speed	-0.5 m/s
$v_{\max}$	Maximum speed	0.5 m/s

Figure 3: The iRobot Create robot.

3 Supplemental Information

This section provides some additional information relevant to specific projects using skid steering vehicles.

3.1 ACL Robotics Challenge (RRTeam)

Table 2 lists the values of the parameters for the iRobot vehicle (figure 3), also referred to as the G-PUCC (**G**round **P**latform for **U**nmanned **C**ooperative **C**ontrol). The characteristic length was measured, and the speed limits are documented in the iRobot code. The time constant was estimated by observing that the rover increased its speed from 0.1 m/s to 0.4 m/s in 1/8th of a second, and adjusting the time constant until the 99% rise time matched this duration.

For some controllers and other applications, it may be useful to have a linearized model of the vehicle. The linearized model for (29)-(33) is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_R \\ \dot{v}_L \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\frac{1}{2}v_R\sin\theta - \frac{1}{2}v_L\sin\theta & \frac{1}{2}\cos\theta & \frac{1}{2}\cos\theta \\ 0 & 0 & \frac{1}{2}v_R\cos\theta + \frac{1}{2}v_L\cos\theta & \frac{1}{2}\sin\theta & \frac{1}{2}\sin\theta \\ 0 & 0 & 0 & \frac{1}{L} & -\frac{1}{L} \\ 0 & 0 & 0 & -\frac{1}{T_v} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{T_v} \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ v_R \\ v_L \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{T_v} & 0 \\ 0 & \frac{1}{T_v} \end{bmatrix} \begin{bmatrix} v_R^{ref} \\ v_L^{ref} \end{bmatrix}.$$

Finally, a summary of the rules for the ACL Robotics Challenge follows.

- **RRTeam Members:** Dan, Georges, Brandon, Mangal, Luke
- **Objective:** design paths from A to B in minimum time with zero contacts
  - Contact is heavy penalty or disqualification
  - If contact is with adversary, any team's bot whose bumper activates is at fault

- **Hardware:** iRobot Create
  - No hardware mods allowed
- **Environment:** RAVEN testbed, with (1) 6-8 drones, (2) 1 adversary, and (3) obstacles
  1. Drones
     - Also iRobots, but fully autonomous
     - Move in rectilinear paths: segments of forward motion interlaced with (zero turning radius) rotations
     - After contact or some duration of time, changes direction
     - Varying speeds, all lower than our vehicle
     - Each vehicle will probably have fixed parameters (time to rotate, rotation angle distribution, speed, etc.)
       * May or may not be deterministic for each individual drone
       * Adds possibility of learning a vehicle's behavior over time
  2. Adversary
     - iRobot of another team, completing course at same time
  3. Obstacles
     - Includes pole and possibly artificial obstacles
     - All relevant obstacles will have a Vicon stream available

3.2 Agile Robotics Rover

The Pioneer 3-AT rover (Figure 4), to be used in the Agile Robotics program, has fundamentally the same slip steering vehicle model as the iRobots. (It is still unclear whether it is controlled using the inputs (v_R, v_L) or the inputs $(v, \Delta v)$.) However, whereas the iRobot has only 2 wheels, the Pioneer rover has 4, with the two wheels on each side linked to the same speed. This configuration limits the ability of the vehicle to slip/skid, and thus limits its steering control authority. One way to represent this loss of control is to use in (31) an *effective* characteristic length, L', which is larger than the actual characteristic length. In Ref. [3], for example, $L = 0.4$ m and $L' = 0.58$ m.

I have a separate page of instructions for installing the MobileRobots software (ARIA libraries and MobileSim), which is available to anyone interested. The current task list in developing a control scheme for these rovers is below.

- **Finish putting together the infrastructure needed to run these rovers in the RAVEN testbed.** This includes but is not limited to:
  - Adding reflective dots to the rover and adding models in Vicon/Motion Analysis.
  - Figure out how to best do wireless vehicle control. We tried VNC today, but the fidelity of the connection did not seem sufficient for teleoperation (though it would probably be fine for running autonomous software). Furthermore, we had to connect via MIT wireless, because we could not connect to the "ravenbridge" router. Would it be possible to set up a wireless access point on the rover which external computers can connect to? (I'm bringing a wireless router into work this week.) Another option is the `ArNetworking` open-source software in the Aria libraries.

Figure 4: The Pioneer 3-AT robot.

  - Possibly set up other rovers in the same way.

- **Write some basic software to allow autonomous control of the vehicle.** To determine what the control signals are that are actually sent to the rover, one place to start would be to look at the code for the program `demo` which allows for teleoperational control via the arrow keys. The capability to receive Vicon data needs to be added. Additionally, a separate `Makefile` needs to be written (the code currently uses an existing `Makefile`).

- **Determine how well the vehicle can be controlled using conventional pure pursuit.** Our sense is that the control laws generated by pure pursuit are not specific to explicit-steering vehicles, and can also be applied to skid-steering vehicles (through Δv). To maintain some parallelism and avoid writing too much code, we'd like to be able to run the AR code (which uses pure pursuit) directly on the vehicle. However, some capabilities are not achievable in pure pursuit, such as rotating in place. Thus, a critical test is to implement and test this on a rover to see how useful the control scheme is. Useful resources for writing this implementation include Refs. [1,4], Sergio's FIDO code, and Georges' Ford code.

- **Identify an accurate model of the rover's true behavior.** To use the rovers in Agile Robotics, we will need a precise system model, beyond just the approximate equations (29)-(33). This will probably require the computation of an input-to-output mapping, such as $(v_R^{ref}, v_L^{ref}) \mapsto (v', \dot{\theta}')$ or $(v^{ref}, \Delta v^{ref}) \mapsto (v', \dot{\theta}')$, where $'$ denotes a measured value. Such a mapping should be very similar to the work done in deriving Sergio's vehicle model for FIDO. In Ref. [3], the proposed approach is to measure the turning radius R' and period T' of the vehicle for a fixed set of inputs. These can be used to compute other quantities, such as the true velocity v', angular velocity $\dot{\theta}'$, and characteristic length L':

$$\begin{aligned} v' &= 2\pi R'/T', \\ \dot{\theta}' &= 2\pi/T', \\ L' &= \Delta v/\dot{\theta}'. \end{aligned}$$

References

[1] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. P. How, "Motion planning in complex environments using closed-loop prediction," 2008, submitted to the Proceedings of the IEEE Conference on Guidance, Navigation, and Control.

[2] B. Shamah, "Experimental comparison of skid steering vs. explicit steering for a wheeled mobile robot," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, March 1999.

[3] C. Sae-Hau, "Multi-vehicle rover testbed using a new indoor positioning sensor," September 2003, SM thesis draft, Massachusetts Institute of Technology.

[4] S. Park, J. Deyst, and J. P. How, "Performance and lyapunov stability of a nonlinear path-following guidance method," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 6, pp. 1718–1728, November-December 2007.