AA 274: Principles of Robotic Autonomy — Winter 2018

Lecture 1: Mobile Robot Kinematics

Scribes: S. Spears, C. Covert, A. Hosler, Z. Chase, and H.M. Ewald

The following notes summarize the first lecture of AA 274: Principles of Robotic Autonomy. This lecture covers two introductory materials (Mobile Robot Kinematics and Motion Control), is organized into meaningful sections, and is annotated with formal definitions, graphics and literature references.

Mobile Robot Kinematics

In this class, an emphasis will be placed on mobile robots and vehicles as opposed to stationary machines. With mobile robots, the focus is on motion and trajectories rather than configurations and interactions with the environment. As such, we need to consider the dynamics and kinematic constraints that govern the motion of each robot.

Kinematic Constraints

The motion state, or configuration, of a robot at any point in time is given by the configuration vector [1]:

$$\xi = [\xi_1, \xi_2, \ ... \ \xi_n]^T$$

For example, a differential drive robot would have $\xi = [x, y, \theta]^T$, denoting a 2D position and an orientation angle.

A time series of changing configurations $\xi(t)$ is called a trajectory, and the set of differential equations $\dot{\xi} = f(\xi, u)$ that govern the robot's motion is known as the robot's dynamic model. We will see later that general dynamic models are already known for most robots. They can, however, also be derived from any given robot's kinematic constraints.

Kinematic constraints take the form of mathematical statements involving ξ, that must hold at all times during robot operation. We introduce some formal notation and terminology at this point:

Definition 1 (Holonomic Constraints) *A kinematic constraint is holonomic if it can be expressed as:*

$$h_i(\xi) = 0, \ \text{ for } \ i = 1, ..., k < n$$

Each holonomic constraint reduces the number of degrees of freedom of the robot's configuration by 1 (i.e. it reduces the space of accessible configurations of the vehicle to an n-k dimensional subset.

Definition 2 (Kinematic Constraints) *A mobile robot system with n degrees of freedom has kinematic constraints that can be described in a general way as follows:*

$$a_i(\xi, \dot{\xi}) = 0, \ \text{ for } \ i = 1, ..., k < n$$

A subcategory of kinematic constraints can be expressed in the Pfaffian form:

$$a_i(\xi) \cdot \dot{\xi} = 0, \ \text{ for } \ i = 1, ..., k < n$$

We can see that a holonomic constraint always also imposes a kinematic constraint:

$$h_i(\xi) = 0 \Rightarrow \frac{d\,h_i(\xi)}{dt} = \frac{\partial\,h_i(\xi)}{\partial \xi}\dot{\xi} = 0, \ \text{ for } \ i = 1, ..., k < n$$

However, the converse is generally not true [1].

Nonholonomic Constraints

The general kinematic constraint in the form $a_i(\xi, \dot{\xi})$ can either be a holonomic or a nonholonomic constraint. It is holonomic if it can be integrated to the holonomic form of $h_i(\xi) = 0$. Otherwise, it is nonholonomic [1].

In practice, both types of constraints affect the system differently: holonomic constraints reduce accessibility of the robot by limiting the robot configuration to a lower-dimensional subspace (each constraint removes one degree of freedom), whereas nonholonomic constraints do not limit accessibility at all; instead, they constrain the possible velocities at each configuration state. In particular, Pfaffian form constraints $a_i(\xi) \cdot \dot{\xi} = 0$ allow only velocities from the null space of $a_i(\xi)$.

A kinematic system with any nonholonomic constraints is denoted as a nonholonomic mechanical system.

Example: Single No Slip Disk [1]

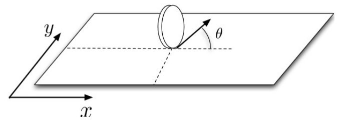

Figure 1.1: Single No Slip Disk Diagram

We consider a disk rolling on its edge to demonstrate a nonholonomic system. Like a differential drive robot, the configuration vector of this system is $\xi = [x, y, \theta]^T$. The kinematic constraint on this system is the no slip constraint, i.e. the disk cannot move laterally - only longitudinally along its orientation.

To formulate this constraint mathematically, we express the two orthogonal directions like this:

$$d_{long} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \perp d_{lat} = \begin{pmatrix} \sin\theta \\ -\cos\theta \end{pmatrix}$$

We can enforce the no slip condition with a dot product statement between the velocity vector and d_{lat} as follows:

$$[\dot{x}, \dot{y}] \begin{pmatrix} \sin\theta \\ -\cos\theta \end{pmatrix} = 0 \iff [\sin\theta, -\cos\theta, 0]\dot{\xi} = 0$$

This is a nonholonomic constraint in Pfaffian form. In this system there is no loss of accessibility.

The example shows that wheeled vehicles are generally nonholonomic, meaning they can reach any configuration $[x, y, \theta]^T$ but will have constrained trajectories to get there. A common holonomic system, on the other hand, would be a robotic arm that cannot reach any possible configuration due to mechanical interactions. In this case, its joints may constrain it such that some configurations are unreachable.

Types of wheels

There are four main wheel types in mobile robotics. Since they are different in the way they constrain the robot's motion, knowing them is important to finding out the kinematic constraints and dynamic model of the vehicle.

The **Standard Wheel** is the most ubiquitous of the wheel types. Its defining property is that the wheel is mounted on the robot's frame directly above the wheel's point of contact with the ground. They can either be of fixed orientation, or they can be fitted to be steerable[2].

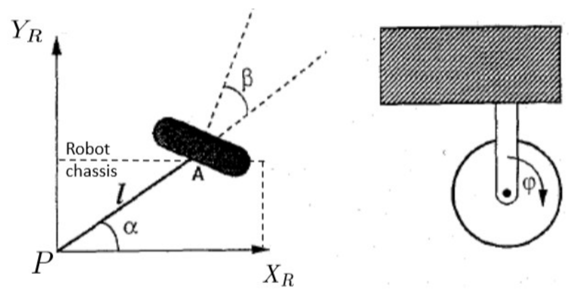

Figure 1.2: Standard Wheel

The **Castor Wheel** differs from the fixed wheel in that it is mounted on the robot in an off-centered position and always fastened to a rotating axis. The wheel's orientation is then either actively steered or passively governed by the robot's motion. The main advantage of the Castor Wheel is that the off-centered axis of rotation essentially allows the wheel to circumvent the no-slip constraint. Of course, it still cannot achieve true omnidirectional motion because it can only exert force along its orientation[2].

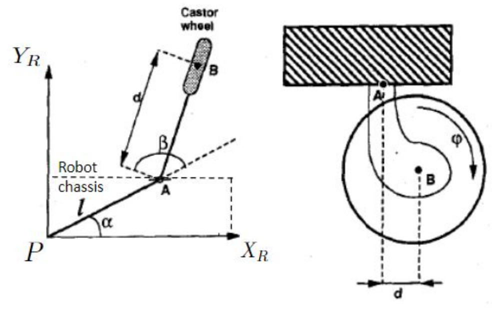

Figure 1.3: Castor Wheel

The **Swedish Wheel** and the **Spherical wheel** are special wheel designs that both achieve omnidirectional motion in different ways. The Swedish Wheel is a big wheel with smaller rollers mounted on the rim allowing active movement in all directions[2]. The Spherical Wheel uses a different concept, where the "wheel" itself is actually ball-shaped. This wheel is suspended in a cage where it is in contact with a series of small motors that can turn the sphere to roll in any direction. They are quite rare however, so they will not play a large part in this class.

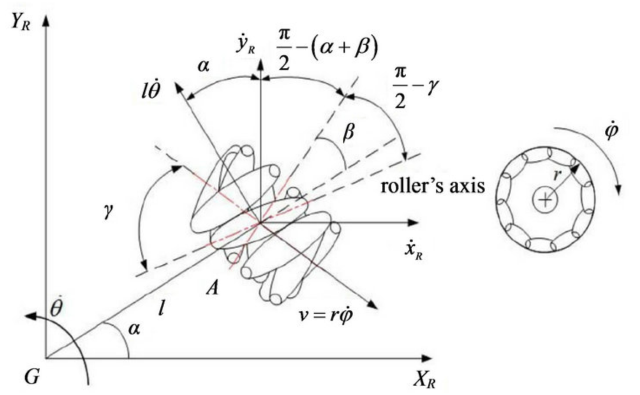

Figure 1.4: Omnidirectional Swedish Wheel [4]

Kinematic Robot Models

By putting together all the constraints that arise from the no slip conditions, we can come up with a "Kinematic Model" of our robot. While we do not have to do this in this class, to derive the kinematic model for your given robot, you would compile the no-slip constraints into a matrix, then look at the null space of that constraint matrix. This will tell you the set of possible velocities of your robot, which is the vehicle's kinematic model. We will use some simple kinematic models in this class to describe the motion of our mobile robot:

"Unicycle" (a.k.a. "differential drive") Model:

This configuration type utilizes two rear standard wheels and an optional passive wheel in the front of the vehicle. In order to achieve rotation, the unicycle model makes use of differential drive by controlling the rate/direction of rotation for each wheel independently[2].

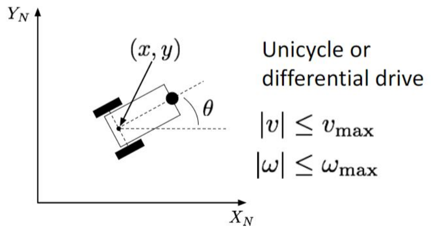

Figure 1.5: Unicycle Model

The kinematic model of unicycle robots is as follows:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w$$

"Tricycle" (or simple car) Model:

Unlike the unicycle model, this configuration type utilizes two rear standard wheels and an active steering wheel(s) in the front of the vehicle. The use of both differential drive and active steering allow the tricycle model to naturally model the kinematic constraints of a simple car[2].

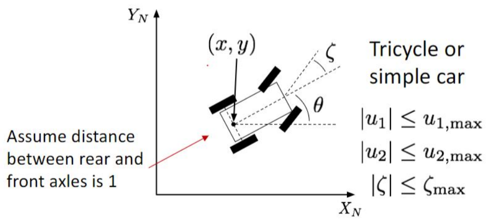

Figure 1.6: Tricycle Model

The kinematic model of tricycle robots is as follows:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\xi} \end{pmatrix} = \begin{pmatrix} \cos\xi\cos\theta \\ \cos\xi\sin\theta \\ \sin\xi \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_2$$

Warning: A kinematic state space model should only be interpreted as a subsystem of a more general dynamical model of the robot.

You control the robot by controlling actual forces and torques, not linear and angular speeds. Usually we assume that there is a black box that converts desired angular and linear velocities to actual torques on each wheel. Alternatively, we could reason directly with the dynamical model, but we won't do this until later in the class.

Simplified car models

Assuming we do not care about the direction of the front wheels, we can set $v = u_1 \cos \xi$, and $w = u_1 \sin \xi$, and plug these two equalities into our kinematic model of the tricycle model, yielding:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\xi} \end{pmatrix} = \begin{pmatrix} \cos \xi \cos \theta \\ \cos \xi \sin \theta \\ \sin \xi \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_2 \Rightarrow \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos \theta \\ \sin \theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w$$

As it can clearly be seen, this assumption has reduced our simple car-like system to the equivalent kinematic representation of the unicycle model. Of note, however, is that while the kinematic model reduces to become indiscriminate of the unicycle model, there is a coupling of the constraints that changes the behavior of the vehicle.

Below are a few examples of how the relationship of v and w relate to $u_{1,max}$ [3]:

$|v| \leq u_{1,max} \cos(\zeta_{max}),\ |w| \leq |v| \tan(\zeta_{max}) \rightarrow$ Car-like robot

$|v| = u_{1,max} \cos(\zeta_{max}),\ |w| \leq |v| \tan(\zeta_{max}) \rightarrow$ Reeds & Shepp's car

$v = u_{1,max} \cos(\zeta_{max}),\ |w| \leq |v| \tan(\zeta_{max}) \rightarrow$ Dubin's car

Motion Control

With an understanding of the kinematic model, the next logistical step is applying those constraint relations to control the vehicle along a path from a given initial configuration to a desired final configuration. This movement of the robot from point A to point B is referred to as *Motion Control*.

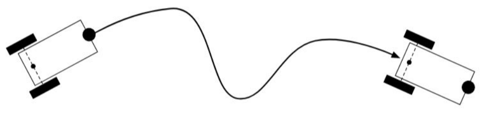

Figure 1.7: Simple Robot Path

The aim of this discussion, as covered in this class, will be to provide the fundamentals of optimal control and trajectory optimization, as well as to develop an appreciation for the methods of achieving those paths via open-loop, closed-loop, and two-degree-of-freedom control designs.

Optimal Control Problem

Our formulation of the optimal control problem is makes use of a state, as denoted by x instead of ξ, where ξ is simply the configuration of the vehicle and x is the combined state of configuration and respective velocities.

Since x may in turn denote non-zero velocity states, it may represent a dynamical model, so this problem represents a dynamics equation which may depend linearly on the state *x(t)*, on the control *u(t)*, and on time t with constraints in both state and control (i.e. upper bounds on acceleration).

The mathematical definition of the optimal control problem can be written as such:

$$\min_{\mathbf{u}} \quad h(\mathbf{x}(t_f), t_f) \; + \; \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt$$

$$\textit{subject to } \dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t), \;\; \textit{with} \;\; \mathbf{x}(t) \in \boldsymbol{X}, \quad \mathbf{u}(t) \in \boldsymbol{U}$$

$$\textit{where } \boldsymbol{x}(t) \in R^n, \; \boldsymbol{u}(t) \in R^m, \textit{ and } \boldsymbol{x}(t_0) = \boldsymbol{x}_0$$

In order to solve this problem given some initial condition, we optimize the cost function until we reach a desired final condition at t_f. In particular, the cost function is additive wherein the cost is (1) an integral of running cost that, at each time interval, penalizes your system for being in state x(t) and enacting control u(t), and (2) a terminal penalty cost (i.e. we might be a little bit away from our waypoints at the final time).

It is important to remember that the integral is simply an additive operation, so this problem may be approached using a discretization of the integral where we add immediate stage-wise costs. With this is mind, we prefer to use this generalized model for two reasons: (1) there are a number of tasks that may be modeled by additive costs (i.e. fuel consumption, control effort, etc), and (2) it dramatically simplifies computation (linear constraint with a number of key, deep, structural results that stem from such an additive assumption).

For those interested in learning more about Optimal Control Theory, a recommended reference to check out is: *D. K. Kirk. Optimal Control Theory: An introduction. Dover Publications, 2004.*

Form of Optimal Control

There exist different forms of an optimized control function, the focus of which revolve around the use of feedback in the iteration of the aforementioned cost function. For notation's sake, if our control input u*(t) represents optimality, then u*(t) is the control function that minimizes the optimization problem.

Closed-Loop Control

If your function is a function of the state and time, such a control function is referred to as a closed-loop control function:

$$\mathbf{u}^*(t) = \pi(\mathbf{x}(t), t)$$

The closed-loop stems from the fact that your control is a function that maps the state to a given control input. In general, closed-loop control is the most accurate form of motion control since the pairing of state and time within the feedback loop allows the vehicle to determine any error in its configuration and correct it along the way. In other words, any deviation from a determined path (i.e. disturbances from wind, slipping, etc) will be detected and corrected by the vehicle.

Open-Loop Control

At the other end of the spectrum is open-loop control (cost function depends only on time and initial conditions, and does not update the state in the feedback loop):

$$\mathbf{u}^*(t) = \mathbf{f}(\mathbf{x}(t_0), t)$$

Therefore, optimizing over open-loop is a much easier process since you optimize only over time instead of time and state (lower dimensionality). With open-loop control, however, deviations from nominal trajectory are impossible to correct.

Two-Step Design

A good compromise between closed-loop and open-loop control is a two-step design where essentially your control function (A) is an additive combination between a reference open-loop control function (B) plus some reference tracking closed-loop control policy (C), a closed-loop function that keeps the system as close as possible to the nominal behavior that is a function of the current state and the tracking error (D)). This is the same concept as a feed-forward control technique.

$$\underbrace{\mathbf{u}^*(t)}_{A} = \underbrace{\mathbf{u}_d(t)}_{B} + \underbrace{\pi}_{C}(\mathbf{x}(t), \underbrace{\mathbf{x}(t) - \mathbf{x}_d(t)}_{D})$$

Open-Loop Control Methods (Discretization)

In particular for open-loop control, there are two broad classes of methods: indirect and direct methods. While both methods involve the steps of discretizing and optimizing the control function, the order in which those steps are completed determines which method is being applied.

Indirect Method

In an indirect method, an optimal solution is found indirectly by first characterizing the necessary condition that an optimal control function should satisfy, and then solving for those optimality conditions. Therefore, this solution follows the order of ”first optimize, then discretize.”

Direct Method

Direct methods are the reverse of indirect methods. With an optimal control problem that is defined in continuous time, we first discretize our solution and then transcribe it into a finite dimensional, nonlinear programming (NLP) problem, which we can then solve. This is method therefore requires us to ”discretize first, then optimize.”

References

[1] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo. *Robotics: Modelling, Planning, and Control.* Springer, 2008 (chapter 11).

[2] R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza. *Introduction to Autonomous Mobile Robots.* MIT Press, 2nd Edition, 2011.

[3] J.-P. Laumond. *Robot Motion Planning and Control.* 1998 texttthttps://homepages.laas.fr/jpl/promotion/chap1.pdf

[4] L. Lin, H. Shih. *Modeling and Adaptive Control of an Omni-Mecanum-Wheeled Robot.* Intelligent Control and Automation, 4, 166-179. doi: 10.4236/ica.2013.42021.