Motion Planning for Simple Two-Wheeled Robots
Ronald I. Greenberg[1] and Jeffery M. Karp
rig@cs.luc.edu (Loyola University Chicago) and jmk01@live.com

Motion Planning for Simple Two-Wheeled Robots

Abstract

This paper considers various simple ways of navigating in a 2-dimensional territory with a two-wheeled robot of a type typical in educational robotics. We determine shortest paths under various modes of operation and compare.

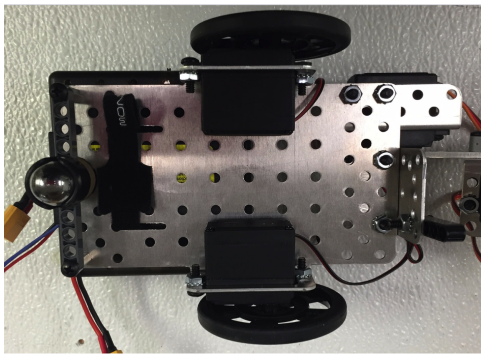
1 Introduction

Providing students with robotics experiences has become a popular and successful mechanism for broadening participation in computing and STEM more generally, retaining more students in these fields, and improving their learning. Robotics videos were found to be the most popular component of a series of brief computing outreach visits [12], and many studies have reported on successful robotics programs, especially with constructions from kits of LEGO and other pieces, e.g. [4, 7, 9, 15]. Robotics is also being integrated into new high school computing curricula such as the Exploring Computer Science (ECS) curriculum [6], which has spread to many locations in the United States [5], has had a particularly strong impact in Chicago [3, 13], and is attracting international attention as well. The discussion in this paper, with its mathematical content can also provide excellent integration between computing and mathematics as with programs such as Bootstrap [1].

This paper provides some guidance students may use for motion planning when working with simple educational robots with a typical, basic locomotion mechanism. (While some guidance has been prepared regarding building robots using LEGO [8, 11], little guidance for students is available regarding motion planning.) Specifically, we focus on navigation through a two-dimensional field, using a two-wheeled robot as is the situation for many educational robots (with a third balance point being provided by a tracking ball or such). For example, the popular Botball® [10] educational robotics program[2] provides parts often built into a robot with two wheels controlled independently by separate motors that can drive them forwards or backwards at settings up to a maximum speed determined by the motor provided. Figure 1 shows the underside of the 2017 Botball Demobot. (Botball actually also provides another two-wheeled robotics platform in the form of an iRobot® Roomba® base, and the discussion in this paper may be useful when working with other two-wheeled robots, even such sophisticated ones as a self-balancing Segway PT. But we focus especially on simple robots constructed from LEGO; these tend to move slowly, which increases the importance of planning the quickest route, and they generally have a great deal of flexibility in wheel placement, which is a point of design for which we can explore the ramifications.)

[1]Supported in part by National Science Foundation grants CNS-1542971 and CNS-1543217.

[2]Botball has spread to many locations on four continents [2] and has major culminating tournaments/conferences in both North America (GCER) and Europe (ECER).



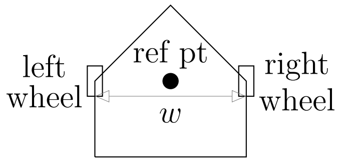


Figure 1: On the left is the underside of the 2017 Botball Demobot showing the two wheels with independent motors and the tracking ball that serves as a third balance point. On the right is the way we will draw our robot with the reference point centered between the wheels.

The key parameter for the robots we will consider is the distance between the centers of the two wheels, typically referred to as the *wheel track* or *track width* (e.g. [14]), which we denote by w. Intuitively, a small w makes the robot able to move more nimbly (assuming one avoids values that are so small as to lead to problems with rollover, which happens to be the subject of the previous reference). In the limit, with $w = 0$, we would be essentially dealing with moving a single point around our two-dimensional field. Scholarly work on motion planning has considered even complicated scenarios with many obstacles, moving obstacles, etc. [16]. In this paper, we allow only for the possibility of some mild peripheral obstacles, but we go beyond the most heavily studied single-point scenario to examine practicalities with typical two-wheeled robots. We will not consider using sensors, various types of which are sometimes provided in robotics kits, but rather will assume navigation by way of dead reckoning through a known terrain.

We will think about moving a certain reference point on the robot from a starting position at coordinates $(0, 0)$, without significant loss of generality, to a target position (x, y), but we will keep in mind that the movement is constrained by the use of our two wheels at separation w. Unlike wheels in a typical modern automobile or even a child's wagon, these wheels do not turn from side to side but rather only rotate forwards or backwards as powered by their two independent motors. We ignore any acceleration and deceleration requirements and rather use the simplifying approximation that wheels are instantaneously switched to any desired speed within the available range. We also assume $y \geq w$ and $x \geq 0$.

Figure 1 shows how we will sketch our robot in later figures. We will begin by considering simplied navigation schemes in which most motion is rectilinear (horizontal or vertical), and then we will consider more general navigation.

2 Primarily Rectilinear Navigation

One natural simplifying approach to navigation is to utilize a subroutine that performs a 90° right turn, a subroutine that performs a 90° left turn, and a subroutine that sends the robot straight ahead for a specified amount of time.

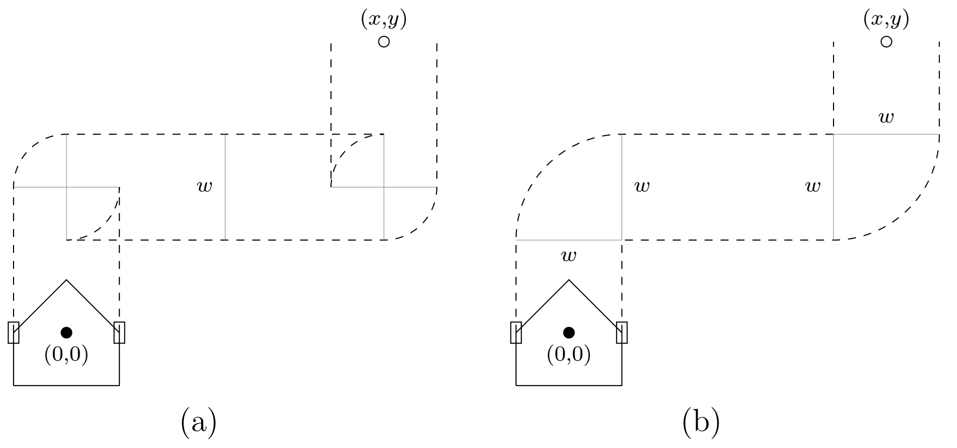


Figure 2: The paths of the robot wheels under primarily rectilinear navigation, using rotations in (a) and ordinary turns in (b).

There are two straightforward ways to program the turns. One approach for a right turn is to put full forward power on the left wheel while putting full backward power on the right wheel. (We use full power under the assumption that maximum progress is desired to complete a navigation task in minimum time, since robotics competitions typically impose a time limit.) A left turn would be analogous with the roles of the wheels reversed. Let us refer to these as right and left rotations, and one may note that this is actually a way to represent the system as a single moving point, since a reference point centered directly between the wheels will not move during a rotation. We want to consider other modes of navigation, however, because they may be more efficient.

The second way to program a right turn is to put full forward power on the left wheel while not rotating the right wheel. In this case, the left wheel will trace an arc of motion that is more like what we do when driving autombiles. Again, there is an analogous left turn, and we will refer to these as ordinary right and left turns.

Figures 2(a) and 2(b) show the paths under the "rotation" and "ordinary turn" approaches when moving the centered reference point from $(0,0)$ to (x,y) assuming we take a "middling" path through the terrain that is likely to avoid obstacles in a typical educational robotics setting. (If there is actually some more particular need to avoid an obstacle, the point at which the first turn is taken can be easily adjusted without affecting the navigation time.)

Under the rotation approach of Fig. 2(a), both wheels are always in motion, so we can compute the time as being proportional to the distance traveled by either wheel, i.e., $x + y + \pi w/2$. Under the ordinary turn approach of Fig. 2(b), we can again consider either wheel, but when that wheel is stationary, we must account for the distance traveled by the other wheel; thus the time is proportional to $(x-w)+(y-w)+\pi w = x+y+(\pi-2)w$. The ordinary turn approach is therefore superior at a time savings proportional to $2w - \pi w/2 \approx w/2$

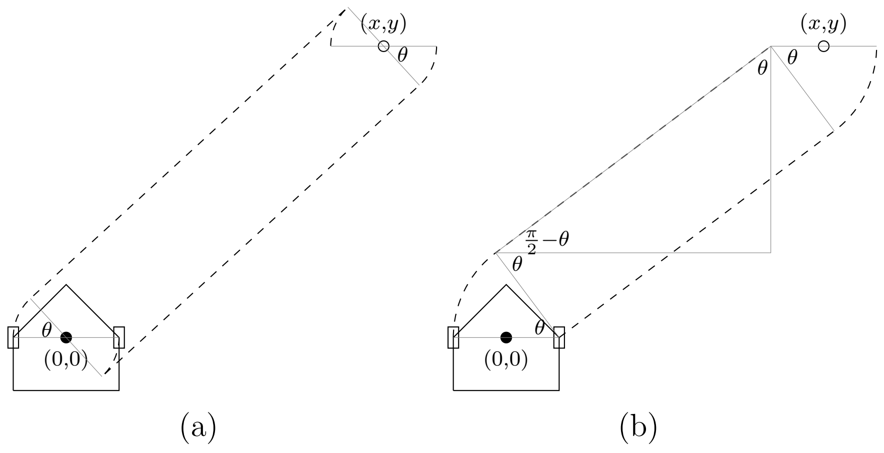


Figure 3: The paths of the robot wheels using rotations in (a) and ordinary turns in (b).

3 Generalized Navigation

While the rectilinear navigational approach of the prior section is simple, we would expect to be able to navigate more quickly by proceeding on a path closer to a straight line. We can still work with subroutines for turning right and left and for driving straight; we just need to parameterize the turns so that they can be executed for an appropriate amount of time according to the degrees desired, and we need to do some trigonometric calculation.

As in the previous section, we will consider "rotations" and "ordinary turns".

Figures 3(a) and 3(b) show the paths under the "rotation" and "ordinary turn" approaches when moving the centered reference point from $(0, 0)$ to (x, y) assuming we use the rotations or turns just to line us up for straight-line navigation. The rotation approach does have a potential drawback that it is more likely to run into some sort of obstacle, e.g., the perimeter of a tight starting box in a typical educational robotics setting, while the approach with ordinary turns has a drawback of being computationally more complex.

Under the rotation approach of Fig. 3(a), both wheels are always in motion, so we can compute the time as being proportional to the distance traveled by either wheel, i.e.,

$$\sqrt{x^2 + y^2} + \theta w \tag{1}$$

with θ in radians and

$$\tan\theta = x/y\ . \tag{2}$$

Under the ordinary turn approach of Fig. 3(b), we can again consider either wheel, but when that wheel is stationary, we must account for the distance traveled by the other wheel; thus the time is proportional to

$$\sqrt{(x - w + w\cos\theta)^2 + (y - w\sin\theta)^2} + 2\theta w \tag{3}$$

with θ in radians and

$$\tan\theta = \frac{x - w + w\cos\theta}{y - w\sin\theta}\ . \tag{4}$$

We intend to continue this work by investigating how to solve for θ in (4) and comparing the times of (1) for rotations and (3) for ordinary turns. We can at least say immediately that as x or y or both get large, the solutions for θ in (2) and (4) approach the same value, and the ordinary turn approach takes longer by an amount of time proportional to θw.

4 Conclusion

We have achieved partial resolution of the question of how best to navigate between two designated points with a typical two-wheeled educational robot. While, the "straight-line" routings of Sect. 3 can be expected to be faster (according to the basic triange inequality), the "rectiliniear" approach in Sect. 2 may be preferred for simplicity and/or avoidance of obstacles near the start and end positions. We also have seen that under rectilinear routing, it is faster to use ordinary turns rather than rotations; the opposite is true for straight-line routing, at least with large travel distance, but ordinary turns may remain preferable there for obstacle avoidance. We also note that the navigation time generally increases with the track width.

Further matters for investigation are noted in Sect. 3. It also will be desirable to consider the case in which the orientation (angle) of the robot is to be changed in the final position relative to the start position. Also, the routes selected are intuitively shortest for the various modes of operation considered, but it would be desirable to provide a formal proof. It would also be desirable to account for acceleration and deceleration requirements or at least to also consider an alternative trajectory in Section 3 that has less dramatic speed changes though it would tend to have greater length, the natural choice being a path comprised of exactly two circular arcs.

References

[1] bootstrapworld.org. Bootstrap. `http://www.bootstrapworld.org` accessed 1/15/17.

[2] Botball Educational Robotics Program. Regions & teams. `http://www.botball.org/regions-teams`. Accessed Jan. 2, 2017.

[3] Lucia Dettori, Ronald I. Greenberg, Steven McGee, and Dale Reed. The impact of the Exploring Computer Science instructional model in Chicago Public Schools. *Computing in Science & Engineering (Special Issue: Best of RESPECT 2015)*, 18(2):10–17, March/April 2016.

[4] Barbara Ericson and Tom McKlin. Effective and sustainable computing summer camps. In *SIGCSE '12*, pages 289–294. Association for Computing Machinery, 2012.

[5] Exploring Computer Science. A national program. `http://www.exploringcs.org/about/ecs-now`, 2016. Accessed Dec. 29, 2016.

[6] Joanna Goode and Gail Chapman. Exploring computer science (version 6.2). `http://www.exploringcs.org/curriculum`, 2015.

[7] Laura M. Grabowski and Pearl Brazier. Robots, recruitment and retention: Broadening participation through CS0. In *Proceedings of 2011 Frontiers in Education Conference (FIE)*, pages F4H1–5, 2011.

[8] Ronald I. Greenberg. Pythagorean approximations for LEGO: Merging educational robot construction with programming and data analysis. In *Proceedings of the 8th International Conference on Robotics in Education, RiE 2017*, Sofia, Bulgaria, April 2017. To appear. Revised and expanded version of "Pythagorean Combinations for LEGO Robot Building" in 2016 Global Confence on Educational Robotics.

[9] Seung Han Kim and Jae Wook Jeon. Introduction for freshmen to embedded systems using LEGO Mindstorms. *IEEE Transactions on Education*, 52(1):99–108, 2009.

[10] KISS Institute for Practical Robotics. Botball® educational robotics program. `http://www.botball.org`, 2015. Accessed June 8, 2016.

[11] Fred G. Martin. The art of LEGO design. *The Robotics Practitioner: The Journal for Robot Builders*, 1(2), Spring 1995.

[12] Steven McGee, Ronald I. Greenberg, Dale F. Reed, and Jennifer Duck. Evaluation of the IMPACTS computer science presentations. *The Journal for Computing Teachers*, pages 26–40, Summer 2013. International Society for Technology in Education, `www.iste.org`.

[13] Steven McGee, Randi McGee-Tekula, Jennifer Duck, Ronald I. Greenberg, Lucia Dettori, Dale F. Reed, Brenda Wilkerson, Don Yanek, Andrew M. Rasmussen, and Gail Chapman. Does a taste of computing increase computer science enrollment? *Computing in Science & Engineering (Special Issue: Best of RESPECT 2016)*, 9(3):8–18, April 2017.

[14] National Highway Traffic Safety Administration. Variable ride-height. `http://www.safecar.gov/Vehicle-Shoppers/Rollover/Variable-Ride-Height`. Accessed Jan. 2, 2017.

[15] R. Brook Osborne, Anthony J. Thomas, and Jeffrey R. N. Forbes. Teaching with robots: A service learning approach to mentor training. In *SIGCSE '10*, pages 172–176. Association for Computing Machinery, 2010.

[16] Micha Sharir. Algorithmic motion planning in robotics. *IEEE Computer*, 22(3):9–20, March 1989.