Mentoring Middle School NXT Robotics: Math Development as a Primary Design Constraint

Richard Zajac
Kansas State University – Salina

Abstract

Challenges experienced with being a STEM mentor in a middle school robotics club are discussed. The immature cognitive and mathematical development of these students are found to impose major constraints on how robot algorithms and other learning opportunities must be designed and framed for students. The design of a competition robot for a line-following event that middle schoolers can understand is discussed as a representative example to illustrate these difficulties, and to highlight particular adaptive tricks and strategies for effective STEM mentoring.

Introduction

A weekly afterschool robotics program[1] based on Lego NXT Mindstorms kits has been active at Salina South Middle School (SMS) since 2007, providing a mutually beneficial outreach opportunity for community STEM mentors such as myself to engage middle school students. Since my own involvement, which began in 2009, this robotics club has competed each year in a regional competition.[2] This competition has been the focal point of the whole year's activity for the club. Each year, the competition comprises about five events, most of which require that a robot be built and programmed to perform specific tasks completely autonomously. Examples of previous events include:

- Following a curving, bendy line of black electrical tape to its dead-end and back in the shortest amount of time (a.k.a. "Line Follower");
- Maneuvering through a rectangular-grid maze, whose dimensions are usually provided in advance (a.k.a. "Maze Madness");
- Fine-tuning the robot's speed as it crosses a series of irregularly spaced lines so as to maintain an average rate of one line crossed per second (a.k.a. "Speed Limit");
- A one-on-one duel to force a competitor's robot out of an area designated by a black surface ("SumoBot").

These events provide problem-solving challenges to small teams of 1-5 students, to be solved within the contraints imposed by the limited contents of a standard Lego Mindstorms kit and the limited time available between the announcement of the event and the actual competition, usually about three months. These challenges are thus grounded in a real-life problem, and foster skill sets that straddle the boundaries between physics, engineering, math, and programming.

The Line Follower Challenge as a Mentoring Analogy

The Line Follower event has been a staple of the competition since its inception. As we shall see, it is also a keen metaphor for the real goal of a STEM mentor: guiding and inspiring the students. Discussion of the line follower also serves to illustrate to the reader and other potential STEM mentors that the reasoning skills needed to be an effective STEM mentor are in fact quite accessible.

Since a standard Lego NXT Mind storms kit only contains but a single light sensor (which can quantitatively detect how much light is reflected from a surface), line following robots made from a standard NXT kit actually follow the black/white boundary on one edge of the laid-down electrical tape, rather than the line itself. The measured intensity of light reflected from the tape/under layer boundary provides information about the relative surfeit of white area relative to black seen by the robot under the sensor, and can then be used to infer the robot's position relative to the tape edge. This information is used according to some particular algorithm to continuously adjust the robot's heading.

For today's student, the first response when being faced with a problem-solving opportunity is to consult the internet for pre-existing solutions. And it does not disappoint. A quick solution to programming the robot for a line-follower event is readily available[3] for download in the form of a Two-State algorithm (shown in Figure 1). What this same algorithm looks like as seen in the NXT graphical programming interface is shown in Figure 2. According to this algorithm, the

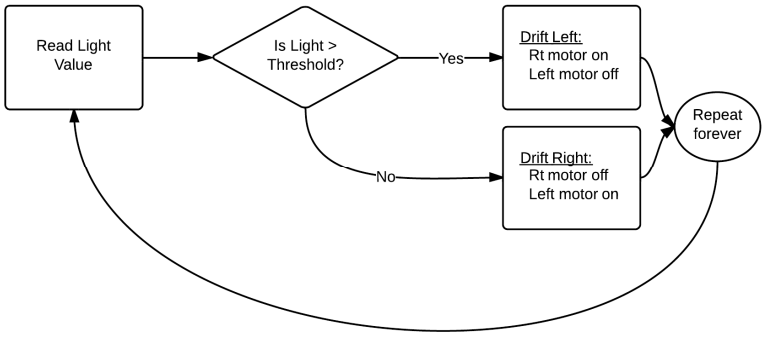

Figure 1. Two-State algorithm for following the right edge of a back line.

robot is perpetually angling itself forward-left or forward-right, depending on whether the instantaneous reading from its light sensor is greater or lesser than some predefined central value. In this way, the robot strives to maintain a proper balance between light and dark (and thereby remaining poised over the line's edge) while attempting to make good progress along its path.

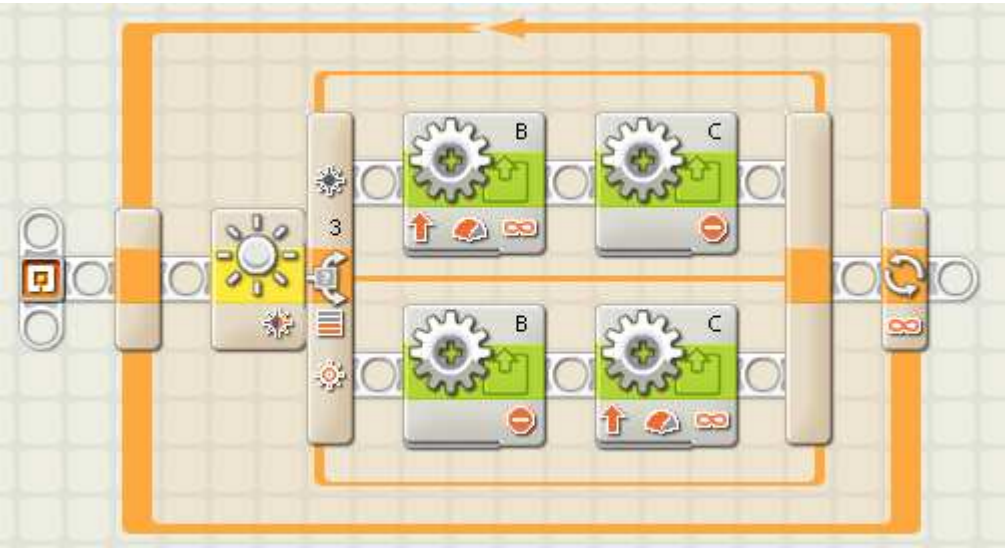

Figure 2. Two-State Line Follower Algorithm as implemented in the NXT Graphical Programming Interface. Here, Motor C is the left motor, and Motor B is the right motor.

In much the same way, STEM mentors must continually hover at the boundary between two states: (a) providing too much assistance in solving the problem, and (b) providing too little assistance. A robot that ventures too far off from either the black or white portions of the field will get lost and wander aimlessly – so too will a student wander aimlessly who is either not sufficiently challenged, or challenged too much (see Figure 3). In either case, the successful design is one that can (hopefully) keep the subject properly centered.

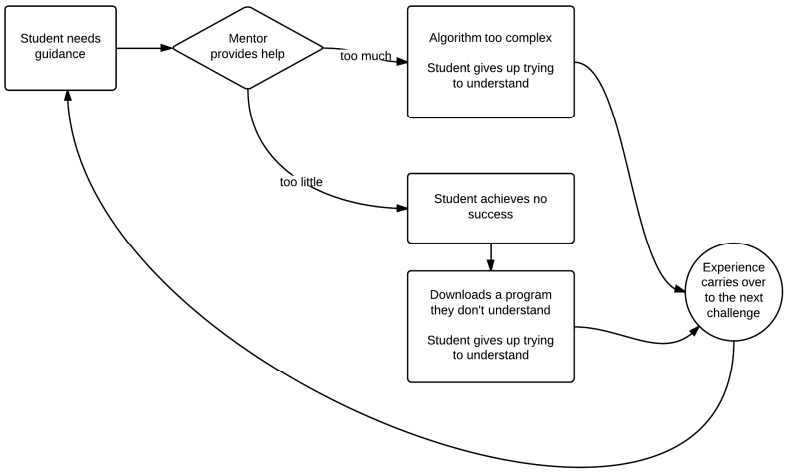

Figure 3. Two-State line follower algorithm as an analogy for balanced mentoring.

Line Following Algorithms

There are, of course, different strategies for a line-following algorithm, as we discuss briefly. For our purposes, a particular algorithm's success at making the robot quickly follow a line is far less important than its success at engaging the middle school student in STEM. A fast, effective algorithm that cannot be understood by a middle school student does not serve this larger purpose. On the other hand, an easy-to-understand algorithm that consistently underperforms at competition cannot be expected to fuel the student's enthusiasm.

Two-State Algorithm

This algorithm is discussed above. It has the chief benefits of requiring virtually no math skills, and being easily understood by middle school students. Despite being simple, it still requires that a few parameters be specified by the students, leaving them some limited opportunity for individual tailoring, experimentation, and risk-taking. As a matter of calibration, students must determine a good middle-of-the-road light value reading to set as the too-dark/too-bright threshold. As a matter of judgment, they must decide how fast and how angled to set the response motions prescribed in each of their two states. Nevertheless, the robot's motion is quite wobbly as it tries to follow the line. This algorithm is typically abandoned by the middle schoolers after they realize that the settings necessary to optimize the robot's motion along a smooth section of line leave the robot incapable of reliably executing the required turnabout where the line dead-ends.

Proportional Line Follower

Often students find a second program available for download[3] involving a more refined proportionality algorithm. This strategy aims to make the robot's degree of steering a continuous variable, which is assigned a value directly proportional to the amount by which the light sensor's reading deviates from the middle-of-the-road value in each iteration. In each cycle of the algorithm a correction is calculated which then gets added to the power supplied to one wheel's motor and subtracted from the other. This correction is linearly proportional to the deviation of the current light sensor's reading from its ideal (50% black) value.

For the majority of middle schoolers, linear proportionality is a slippery concept at best. These conceptual difficulties are exacerbated by the proportionality's involving subtracted amounts (motor power corrections and light reading deviations) rather than direct measurables. Finally, the clumsiness of the graphical programming interface's representation of mathematical operators is visually intimidating, and further dissuades middle school students from understanding how the algorithm works. This algorithm has the benefit that when a proportionality constant is properly chosen for a particular robot (typically arrived at by trial and error), the robot can then follow a line quite smoothly, without the wobbling so prominent in the Two-State algorithm. One-on-one interaction with the students reveals that they do not understand how it works. It is therefore not surprising that these students are incapable of performing any fine-tuning on the program. If the program's preset parameters result in a behavior that fails to properly turnabout at the line's dead-end, the students have no recourse for fixing it. Now, since each team builds its own robot, there is likely to be quite a lot of variability

in mechanical design, and the details of the various robots' wheel base will differ in multiple ways. The poorly-understood algorithm with its default preset parameters will naturally and arbitrarily be better tuned to one student's design than to another. The luck-of-the-draw success or failure of one team's robot versus another leads to underserved blame and unearned merit, with accompanying emotional issues. Such random success or failure undermines the larger lessons of student competitions, namely that persistence and fine-tuning improve performance, and that conceptual understanding is a fundamental problem-solving tool.

PID Controller

In industrial control systems, problems such as the present line follower challenge would commonly be solved with the use of a Proportional-Integral-Derivative controller.[4] Along with the proportionality strategy described above, this algorithm includes two more components:

(1) At every program cycle, the light reading's deviation from the ideal central value is added to a growing sum (cumulative error), which is used to calculate an additional steering correction. In this way, if the normal proportional steering response proves insufficient to quickly correct a robot that has (say) ventured too far into the black, further and continued lingering in the black results in a continuously increasing steering correction. Effectively, the continuous error summation constitutes integration – an application of Calculus.

(2) At every cycle, the algorithm monitors how much the light sensor's reading has changed from its most recent values, and tries to anticipate whether the steering correction being calculated might actually overshoot the intended balance. Effectively, the continuous monitoring of the change in light reading constitutes an instantaneous derivative – also an application of Calculus.

Both of these applications of Calculus introduce further parameters that need to be tuned to achieve optimum performance. Unfortunately, typical middle-school students are even less prepared to understand the meaning of these tunable parameters than they are to understand proportionality concepts. When we recognize the middle schoolers' limited mathematical development as a design constraint, strategies that would otherwise be excellent in an industrial setting are seen to be counter-productive to the STEM-mentoring goal of the challenge.

A Discretized Line-Follower

Since the Two-State algorithm is easily understood but too "dumb", and the Proportionality and PID algorithms are too advanced for the given mathematical constraints, it would seem reasonable to try to expand the Two-State algorithm into an N-State algorithm, with adjustable $N > 2$. In one sense, this might be seen as a Proportionality algorithm, but discretized into a finite number of states, the number of which would be adjustable and would depend on the individual students' ability and willingness to cope with a growing number of states. Such an algorithm is described in the next section.

The Sorting Hat

Over a few years, a discretized N-state line following algorithm has been arrived at and employed to great success at SMS. This success is judged not so much on robot performance, but primarily by the ability of SMS students to understand the algorithm, to knowledgeably customize it according to their own experimental results, and to expand it to whatever degree their individual ambition allows. The algorithm also provides certain mentoring opportunities as we shall see below.

As shown in Figure 4, the algorithm involves dividing the anticipated range of light sensor readings into N possible states. For each of the N states, a heading and speed are pre-chosen for the algorithm to assign to the robot. In each program cycle, a reading is taken from the light sensor, it is determined which of the N states this light reading calls for, the appropriate state's headings and speed are executed, and the cycle is repeated.

The calculation of which bin number results from the particular light sensor reading involves linear algebraic manipulation of several quantities including the instantaneous light reading as well as the anticipated sensor minimum and maximum values. The latter need to have been pre-

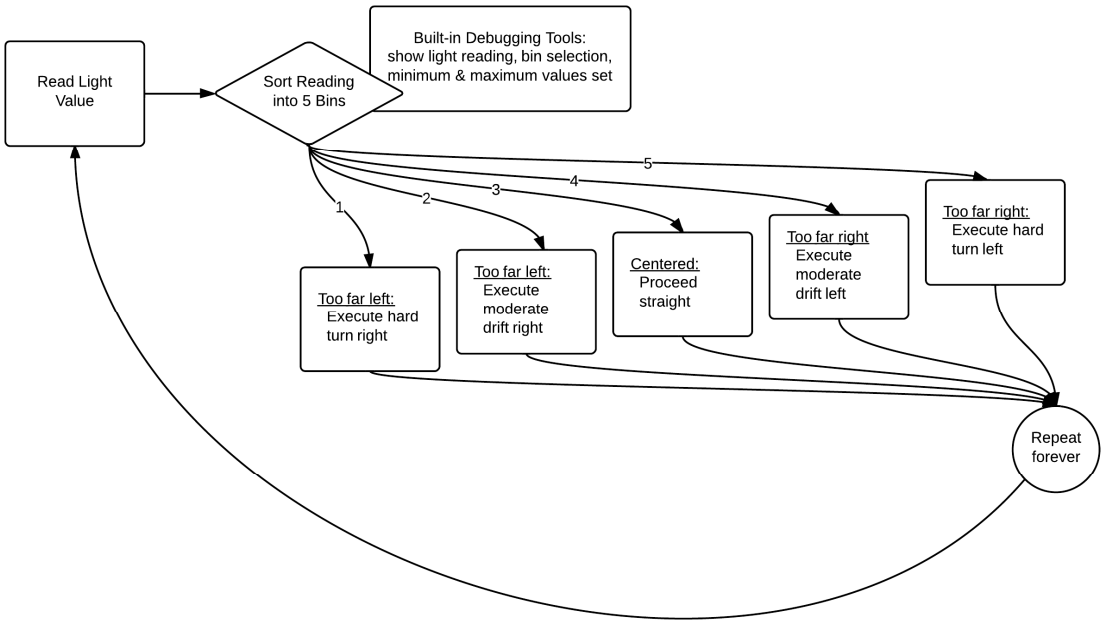

Figure 4. Discretized N-State Algorithm for line follower for N=5.

calibrated by the students and fed into the algorithm. The overall bin calculation is perhaps at the high cusp of what is understandable by middle school students. The math involved is not made easier to understand by the clumsy, bulky way it is presented by the NXT graphical programming interface. However two things help to make the calculation process understandable:

(1) The newer release of the NXT programming application allows the entire bin calculation process to be collapsed into a single "My Block", so that the lengthy and visually intimidating string of clumsy mathematical operation blocks need not detract from the flow of the main algorithm's purpose (see Figure 5);

(2) The concept of binning can be presented to students by relating it to a convenient and timely metaphor borrowed from the Harry Potter series of books.[5] In this way, the collapsed block is presented as a "Sorting Hat", in homage to the way students at Hogwarts get sorted into School Houses. Within the mindset of Harry Potter fans, these School Houses exhibit their own ranking order of increasing desirability, just as the current algorithm must rank its bins in order of increasing brightness). The ability to attach a customized graphic to the MyBlock solidifies the metaphor.

Despite the mathematical details' being thus visually subsumed into a MyBlock, it is still possible for intrepid students to expand the MyBlock to examine its functioning. Less daring students can nevertheless appreciate the Sorting Hat Block's purpose in concrete terms even before their ratio reasoning skills have developed enough to fully deconstruct it.

It is tempting to classify the present Sorting Hat as an intermediate between the Two-State algorithm (described above) and the Proportionality algorithm. After all, in the limit of $N \rightarrow \infty$ one could reasonably expect the N-State sorting Hat and the Proportionality algorithm to behave identically. However, such a comparison does not tell the whole story and this insight has the potential to provide multiple opportunities for mentoring discussions. Firstly, one must recognize that even the so-called Proportionality algorithm, when actually implemented, is itself ultimately discretized owing to the limits of the light sensor, whose output range typically spans about 20 integer values under actual competition conditions. Secondly, the comfort with which students brazenly tune their individual N states' steering responses frequently results in a decidedly non-linear response overall. (Just because one state corresponds to a light reading that

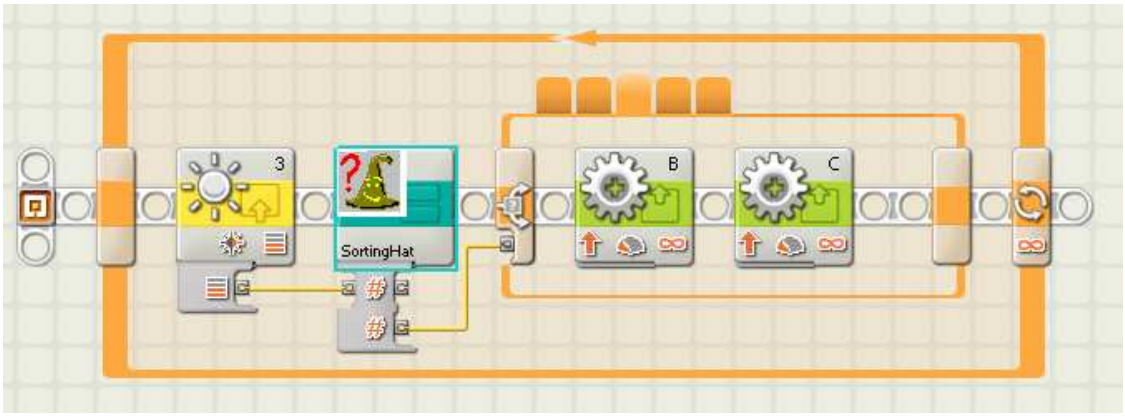

Figure 5. "The Sorting Hat," an implementation of an N-state line following algorithm as it appears in the NXT Graphical programming interface. The five rolodex-style tabs designate five different steering states.

is twice as bright does not mean that the student will choose a steering response that is twice as strong). By comparison, the corresponding non-linear response function (if one could be found to match the students' choices) would add an even greater algebraic complexity to any code attempting to represent the robot's steering response as a continuous function, linear or otherwise. In any case, it is not clear that the light sensor is itself a linear device.

Lessons Learned

In assessing the success of a competition-driven design event for STEM mentoring, successful performance is secondary to the students' ability to understand and comfortably tinker with the technology at hand. As we have seen, this aspect is constrained by the students' limited mathematical development, and these limits must be recognized as design constraints by the mentor. Today's students also display attitudinal factors that successful mentoring must address.

One of these attitudinal factors that was discovered is the new relationship that exists between students and programming. Early student comments revealed that students of the Internet Age primarily consider programs as something to be downloaded, not written. It has proven to require a significant paradigm shift to convince students that the programming for novel competition events is not readily available online, and must be created from scratch. Actually, this particular competition recognizes a "prestige ranking" of its events in which the Line Following event is considered the lowest. Presumably this low ranking tacitly acknowledges (without necessarily condoning) the reality that of all the events, this one is the easiest for which to find some pre-written code online.

Tools such as the Sorting Hat algorithm have proven to constitute a useful stepping stone toward the paradigm of program-writing, inasmuch as the students are presented with a tool that clearly can be more useful than the easily-downloaded Two-State code, but nevertheless requires that the students understand and tailor the program to their own robot design in order to benefit from it. One might argue that simply giving students a tool such the Sorting Hat still falls short of the absolute ideal of requiring students to build their entire code completely from scratch. As we have described with the line-following metaphor of mentoring, students with easy access to downloadable code will likely resort to downloading something they do not understand if the balance of challenge and assistance strays too far from some nominal ratio, just like their straying robots. Viewed within this context, we deem the Sorting Hat to be the preferred alternative. Compared to downloading, it presents a greater likelihood that the students will strive to meaningfully tailor their programming and to problem-solve based on a solid conceptual understanding.

Concrete visual triggers can be powerful tools for ushering students toward that understanding. So for example, it has proven useful to have the Sorting Hat's N outcome options laid out horizontally in the graphical programming interface, so that each option's position along the diagram corresponds visually with the robot's degree of centeredness about the line's edge. (See Figure 5). Thus the option displayed under the leftmost tab of the diagram corresponds to the robot's having drifted too far left; the option displayed under the rightmost tab corresponds to the robot's having drifted too far right, etc. Interestingly, and perhaps related to this concrete visual trigger, students presented with the choice of how many N steering states to implement

overwhelmingly choose an odd-numbered N, because they can then explicitly include a perfectly centered, straight-forward-pointing state at the center of the diagram with all other states arranged symmetrically around it. Empirically, the evenness or oddness of N seems to have no detectable effect on the quality of the robot's performance, but the explicit inclusion of a centered straight-ahead option in odd-numbered-N schemes seems more in tune with the students' intuition.

Another advantage of the students' being provided with a common engine such as the Sorting Hat algorithm is that it allows a mentor to demonstrate the use of explicit debugging tools such as the preemptive inclusion of a display-to-screen function to continuously communicate to the operator the instantaneous values of various dynamical variables. When a robot misbehaves inexplicably, it is of immense value for debugging the problem to be shown (say) what light level the sensor is currently reading, what numbered option the algorithm is choosing, etc. Certainly such feedback makes the mentor's life easier when trying to help a frustrated student make sense of the misbehavior. More importantly, it also serves to model important problem-solving strategies to the student. Unfortunately, beginning programmers do not typically bother to build in such debugging feedback mechanisms. And without experiencing the benefit of such debugging mechanisms, students might never appreciate their usefulness. However, by surreptitiously providing built-in debugging tools, mentors have the opportunity to let students experience the benefit of something they would not otherwise have the patience or appreciation to include themselves.

Conclusion

Middle school students in STEM outreach programs such as robotics clubs don't just benefit from a mentor's technical skill set. There are other skills that potential mentors can contribute that are at least equally valuable. The students' on-going cognitive development gives them the need to understand things in visual and/or concrete terms. Being able to formulate adaptive strategies, such as the ones discussed here, can provide scaffolding for understanding elements of formal reasoning that would otherwise be inaccessible to these students. Showing students how and why to number their various program revisions can foster organizational skills. Productive mentoring can be as simple as a volunteer's taking the initiative to bring in some electrical tape to draw a line course, turning an abstract challenge into a concrete reality.

References

1. Club Contact person: Barbara Livengood, Gifted Coordinator, Salina South Middle School, 2040 South 4th Street Salina KS, 67401.

2. Western Kansas Lego Robotics Competition, Contact: Adams P., Maseberg J., Smith M., Science and Mathematics Education Institute, Fort Hays State University, 600 Park Street, Hays, Kansas 67601.

3. "NXT Programs" (2013). Available WWW: http://www.nxtprograms.com/index.html.

4. See for example Astrom K., Hagglund T. (2006) *Advanced PID Control*, ISA, The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC. 1-5561-7942-1.

5. Rowlings, J. K. (1997) *Harry Potter and the Philosopher's Stone*, Arthur Levine, New York City. 0-7475-3269-9.

Bibliographical Information

ZAJAC, RICHARD is a professor of physics at Kansas State University – Salina, in the Arts, Sciences & Business Department at 2310 Centennial Road, Salina KS 67401.