A Practical Application of Mathematics in Botball: Trigonometric Calculation of Robotic Positioning Errors
Jonathan Malouf and Ian Poorman
DeWitt Perry Middle School
culpd@cfbisd.edu

A Practical Application of Mathematics in Botball: Trigonometric Calculation of Robotic Positioning Errors

1 Introduction

In playing Botball, you may find that the robot you are working with may not do what you want it to do. The BEMF encoders are an excellent way to control a robot; however, they only work well if you set the robot up correctly and build the robot in a solid manner. As is often the case, the robot will have a mechanical issue, such as mismatched motors, or a Lego piece rubbing against a wheel. In this case, the best solution is to fix the physical problem. If there are no physical or mechanical issues, the most important thing is properly setting up the robot in the starting box. If the robot is not correctly set up in the starting box, the robot will not follow the path you have programmed it to follow.

If your robot does not follow the correct path it will not score the desired amount of points and will, worst case scenario, fall off the table and break. This paper will show the reader how to calculate the amount of error that you introduce by setting you robot up incorrectly and ways in which to minimize that error.

2 Basic Trigonometry

Before we could start with our investigations, we needed a basic understanding of trigonometric concepts. In this paper, we will not go into extreme details; we will just cover simple right triangles. Here are some basic trigonometric definitions:

Sine- (abbv. sin) the ratio of the opposite side to the hypotenuse of a right-angled triangle.

Sin A = opposite side/hypotenuse

Tangent- (abbv. tan) the ratio of the opposite to the adjacent side of a right-angled triangle.

Tan A = opposite side/ adjacent side

Cosine- (abbv. cos) the ratio of the length (in a right triangle) of the side adjacent to an acute angle to the length of the hypotenuse.

Cos A = adjacent side/hypotenuse

Here is another, much simpler version the information above:

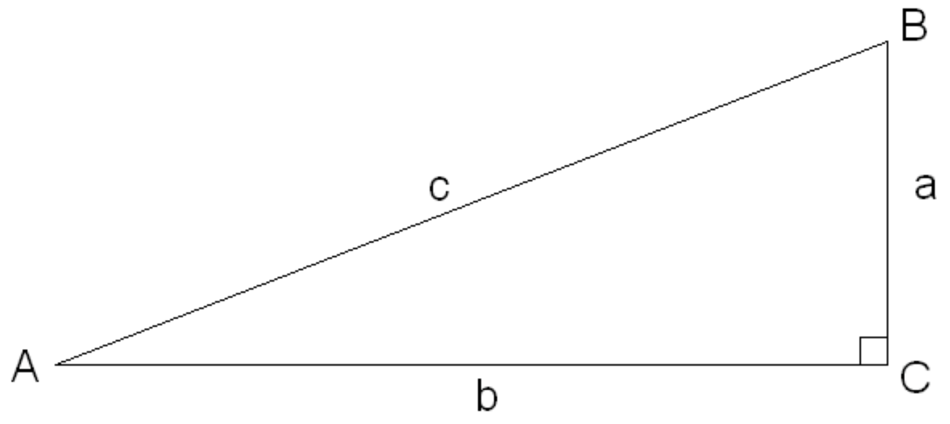
- SOH- Sine- opposite/ hypotenuse

-CAH- Cosine- adjacent/ hypotenuse

-TOA- Tangent- opposite/ adjacent

Now, take a look at this diagram.

(Fig. 1)



This is a picture of a triangle that could represent the path of a robot that has gone off course. The line segment b represents the intended path for the robot to take. C represents the target for the robot. The line segment c represents the actual path that the robot took. Angle A is the number of degrees from line segment b off it is. Line segment a is the distance off the robot is from its target.

Let's say the robot takes off at point A and heads for point C, but winds up at point B. The distance between point A and C is 10 meters, and you calculate that angle A is 30 degrees. From this information, we can determine the distance between point B and C. To do this, we will use the definitions given above. We know the angle A and we know the length of the side adjacent to the angle and we want to find the side opposite of the angle. We look at our definitions and see that the tangent function fits what we need. We setup the problem and it looks something like this:

Tan 30 = opposite/10 meters

We look up the tangent of 30 degrees and find that it is .5773. We can now rewrite the functions as:

.5773 = opposite/ 10 meters

This is now a simple algebra problem, by multiplying both sides by 10 meters we get:

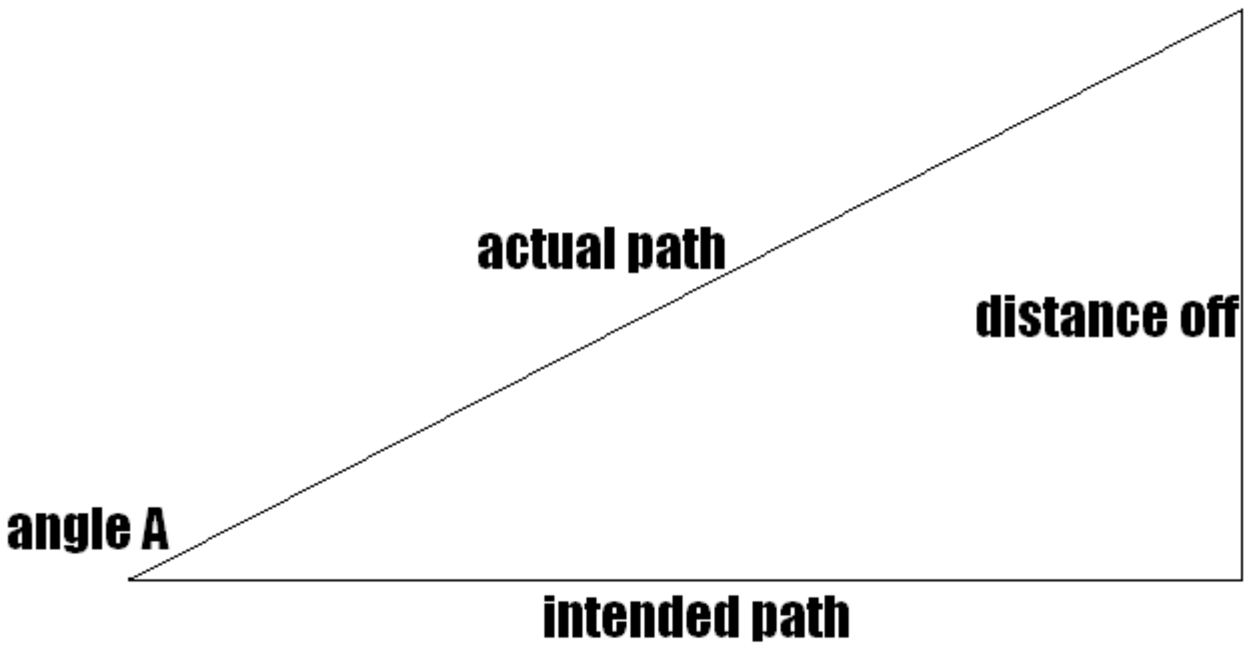
opposite side = 5.773 meters

This means our robot will be an amazing 5.8 meters off of its intended target! Of course we do not use such large distances in Botball but this illustrates the point.

2 Going the Opposite Direction

Take a look at the picture below-

(Fig. 2)



Let's say that you set the robot down on the table. You turn on the robot and the program. The robot is programmed to go forward 20 inches. However, for some reason, the robot's path is skewed off to the side because you set the robot up incorrectly. When it gets to its eventual destination, it is actually 10 inches off. You want to know the measure of degrees off it is. This is very easy. You already have all the information you need to determine the number of degrees off it is, so all you need is a formula. We will divide the distance off (10 inches) by the intended distance (20 inches). This gives you a quotient of 0.5. Then you take the 0.5, and multiply it by tangent to the negative first (arctan). This gives you angle A.

tan A = 10 inches/ 20 inches

tan A = .5

arctan .5 = A

26.35 = A

This means we set our robot up with an angle of 26.35 degrees from normal!

But what can you do with all of this information? Of what use is it to you? The next section will tell you how.

3 Real World Examples

In most cases, you would have a protractor and ruler to measure the degree or distance off you are. But what if your robot isn't on a relatively small field like a Botball table? What if, instead, your robot was in a situation like the DARPA Grand Challenge (but had no sensors)? If it got just a little off of its desired course, it could go for miles in the wrong direction, eventually winding up at a location far from the original destination.

Another example would be if the robot was off by just a little bit. If you test it on a small scale, it might not seem to make a difference, but in a larger run, it would be off by a completely different amount. A robot that is off by one degree will eventually be far off from its original target.

4 Practical Ways to Prevent Problems

There are many things that you can do that will help the robot follow a straight path. These solutions may not always work, but they are usually very effective. There are three main ways to help the robot go straight. One, the template, ensures that the robot is set up correctly. Another one is the use of sensors. They are a very good way to realign your robot mid-way through the game. The other way is to use the landscape to realign yourself.

A template is basically a piece of material (paper, cardboard, etc.) that will keep your robot aligned and in the same spot every time that you or your team uses it. Using a template can be an effective way to align your robot. You bring the template up to the table when you setup the robot and use it to align the robot, the template is then removed from the table.

You can use the terrain to align the robot during the game. You can use the terrain to your advantage, because it will mean that you don't have to spend extra time learning how to program the robot to use sensors, or spend a week learning how to use a new sensor. It will also give you a better chance of being in perfect position, because if it messes up it will probably be able to get back into position and finish the game, by using the landscape. To use the landscape to your advantage, you have to have your robot run into a fixed object, like PVC pipe. For example, this last year one of our teams used the terrain by having a flat front on their robot, the robot would drive into the PVC borders, the flat front of the robot would cause it to align itself parallel to the PVC, it would run into multiple pipes to give it a good idea where it was. It was able to drive over the entire field, and as the very last thing, come back and get two balls all the way across the table just in time before time ran out. It got them everytime by using the PVC pipe to align itself.

Some other tools you can use that are also extremely helpful are the sensors. The basic ones are the infrared, the sonar, the touch sensors, and the camera. The infrared sensor is a distance finder that is somewhat reliable but in the end will be extremely useful to teams that need the robot to know how far an object is to travel that distance. The sonar sensor is also a distance finder but, instead of using pulses of infrared light, the sonar uses pulses of sound that are out of human hearing limits to find object, just like bats or dolphins. The camera is a camera of course, but it can also tell your robot the distance to certain objects based on their size.

5 Conclusion

Robots are dumb. They will only do exactly what you tell them to do. Although it may appear that the robot is not doing what you told it to, it is. That is why it is so important to always make sure that you do everything you can to ensure that your robot does what you tell it to do. The most important thing when working with something that can do exactly what you tell it to do is to be as specific as possible and also help your robot know where it is on the table.

6 References

[1] http://www.google.com/search?hl=en&q=define%3A+sin

[2] http://en.wikipedia.org/wiki/Cosine

[3] http://en.wikipedia.org/wiki/Law_of_cosines

[4] http://www.webmath.com/cgi-bin/rtri.cgi?c=28&e=45&b=20&d=%3F&a=20

[5] http://www.clarku.edu/~djoyce/trig/