Applying High-School Mathematics to Robotics
Claire Zhang, Christina Guan
Polaris, Team 13-0535

Applying High-School Mathematics to Robotics

1 Abstract

We apply high-school level mathematics to make programming and grabbing objects in robotics much more efficient and reliable. Our robot has an arm consisting of two segments. The first segment can rotate 180 degrees, and the second can rotate 360. We have derived a system of equations that allows the arm to grab any object within a semicircle made by the arm. In the second portion of the paper, we calculate the width of the desired object and its length by using the ET sensor and applying trigonometry. This technique makes our program consistent, accurate, and reliable.

2 Introduction

Most programs are coded specifically for one task only, and they cannot be reused. Every year, a new program must be made from scratch. This is extremely tedious as well as time consuming. However, many missions involve performing the same fundamental motions, like lifting or reaching for an object. By applying simple high-school level mathematics such as geometry and trigonometry, we have created programs that can be reused year after year. Furthermore, in Botball we only have a 7-week time period to complete game objectives; when we can reuse programs, we can focus more time and energy on robot design. Our goal is to make our work process simple and efficient.

3 Grabbing Objects with a Two-Segmented Arm

Our objective is to bend the two segments, or arms, to reach a desired point, *(a,b),* shown in **Figure 1**. The locus of the arm lies on a coordinate plane. The first arm has a length of r_1 and rotates 180 degrees around (0, 0). The second arm, which extends from the endpoint of the first arm, has a length of r_2 and rotates 360 degrees. The radius of the large hemisphere is (r_1+r_2). Point *(a, b)* is an input of our program, and it must lie within this hemisphere. The point labeled *(x, y)* is the

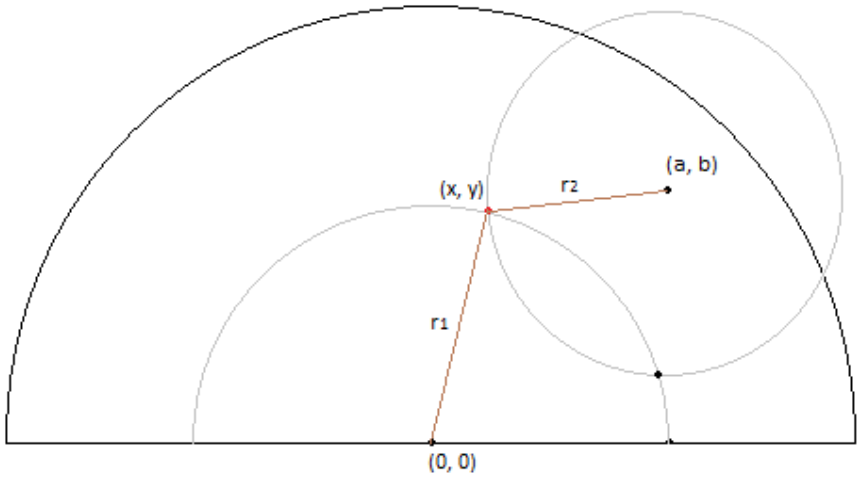

Figure 1: Model of our arm on a coordinate plane.

location of the joint of the arms. The first step is to solve for *(x, y).* After solving for *(x, y)*, we move each arm to the desired location to reach point *(a, b).*

In general, to reach a point in the three dimensional world, a coordinate *(x, y, z)* should be specified. In this paper, we assume that we have navigated and spun the robot to a position and angle such that the point *(a, b)* is within the reach of the arm. Therefore, the third dimenson, or z-axis, involves the movement of the actual robot. This simplifies our problem to a two-dimensional coordinate system.

3.1 Equations of Circles

To find the unknown point *(x, y)*, we will apply our knowledge of circles. The standard equation of a circle is $(x-h)^2+(y-k)^2=r^2$, where *(h, k)* is the center point of the circle. The radius is *r*, and *(x, y)* is any point on the circumference of the circle [2]. This is depicted in **Figure 2**.

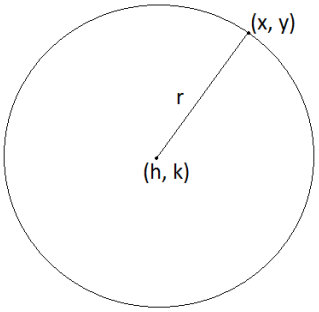

Figure 2: A standard circle.

The equation of our first circle with radius r_1 is $x^2+y^2={r_1}^2$.
The equation of our second circle with radius r_2 is $(x-a)^2+(y-b)^2={r_2}^2$. (These circles are shown in light grey in **Figure 1**.) We now have a solvable system of equations:

- $x^2+y^2={r_1}^2$
- $(x-a)^2+(y-b)^2={r_2}^2$

It is possible to solve these equations algebraically. However, the equation obtained is quartic and difficult to simplify. Therefore, we use a numerical method to find *(x, y)* with a program. This is accomplished within a loop, by plugging in values of *x* and finding the value that best fits the equation with the input *(a, b).*

3.2 Using a Program

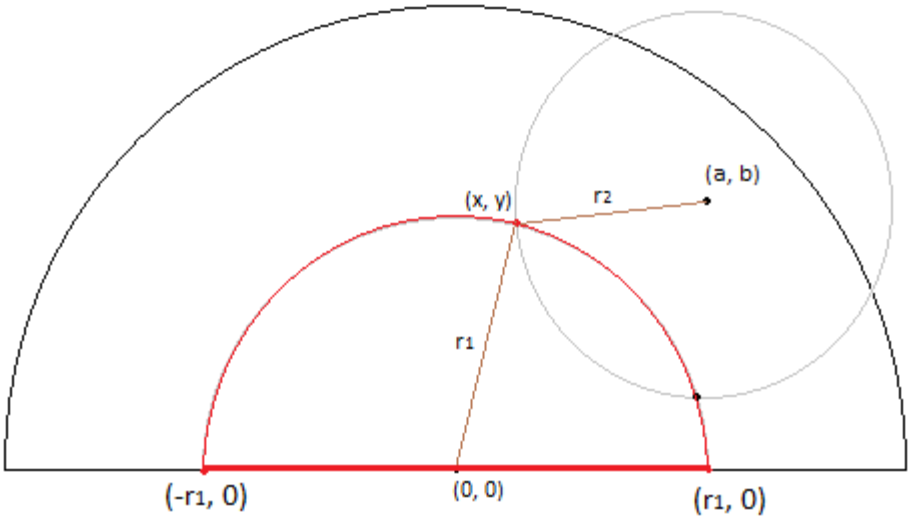

Figure 3: The domain of X.

Because there are infinite values of *x*, we must narrow this set by finding the domain of our equations. *(x, y)* must lie on the circumference of the circle with radius r_1, because the first arm can only rotate 180 degrees. Therefore, the domain is $-r_1 \leq x \leq r_1$. The

horizontal distance the arm covers is $2r_1$. This is highlighted in red in **Figure 3**.

Our program will proceed as follows:

- In a loop, we begin with an x-input of - r_1. To be accurate, we will have 200 iterations. Each consecutive input increases the value of *x* by ($2r_1$/200), the constant difference.
- Plug the x-value into the first equation $x^2 + y^2 = {r_1}^2$ in order to solve for y.
  - The equation must be rearranged into $y = \sqrt{({r_1}^2 - x^2)}$
- Plug the *(x, y)* pair obtained into the expression $(x - a)^2 + (y - b)^2$, and save the value into a variable p.
  - Remember that a and b are constants.
- Find the positive difference (absolute value) between ${r_2}^2$ and p and save this into a variable q**:** if q is great, then the current *(x, y)* is far away from the desired *(x, y).*
- We want to find the smallest value of q in order to find our *(x, y)* pair. If q =0, it means the corresponding *(x, y)* perfectly fits our second equation, $(x - a)^2 + (y - b)^2 = {r_2}^2$.
- Define a global variable d outside of the loop. The variable *q* is the difference calculated within this loop, while the variable *d* stores the smallest *q*-value encountered thus far.
- Continue with the loop and plug in the next *x*-value, which we find by adding the constant difference to the current *x*-value (described above).
- Find the *q*-value for that *x*. If this *q*-value is less than the *d*-value thus far, then save the lesser *q*-value into *d*, replacing the previous value. We also save the corresponding *(x, y)* pair.
- We exit the loop when $d = 0$ or when we have completed all 200 iterations of the *x*-value.
- The final *d*-value obtained will lead us to our desired *(x, y)* pair. We have now discovered the point where our two arms bend to reach *(a, b).*

3.3 Moving the Arms with Trigonometry

Once we have the desired *(x, y)*, we first need to move the r_1 arm to that point, and then the r_2 arm to *(a, b).* We achieve this by utilizing our knowledge of triangles and trigonometry.

We must find the values of $\angle A$ and $\angle B$. Once we have these angles, we rotate our arms by

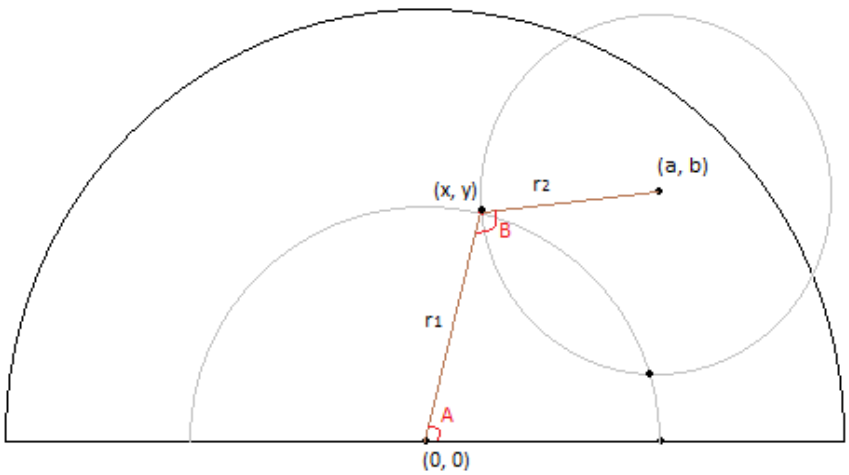

Figure 4: Angles A and B.

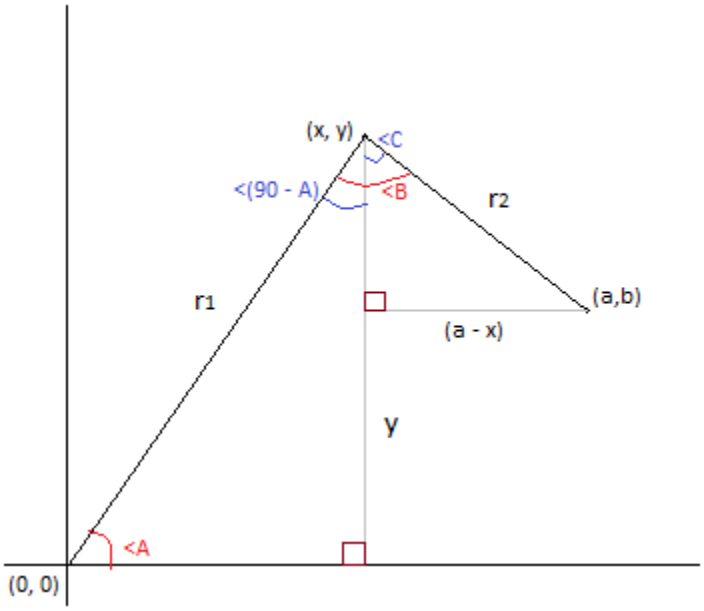

Figure 5: How to find angles A and B.

using the angles as a fraction of a full, 360 degree rotation.
We find $\angle A$ by making use of inverse trigonometric functions [1]:

- $\angle \boldsymbol{A} = \boldsymbol{\sin^{-1}} \frac{\boldsymbol{y}}{\boldsymbol{r_1}}$
- Save this value to a variable n.
- To translate n into a rotation, plug n into the equation $D = r(\frac{n}{360})$, where r is the circumference of the circle made by r_1.
- Rotate the $r_1$arm by D starting from the x-axis.

Finding $\angle \boldsymbol{B}$ is slightly more complicated. First, we draw a horizontal perpendicular line from *(a, b)* to length *y*. This forms a triangle with a base of length $(\boldsymbol{a} - \boldsymbol{x})$. If $\boldsymbol{a} < x$, then we take the absolute value of the difference to find the base.

The steps to find $\angle B$ are:

- Find $\angle \boldsymbol{C}$: $\angle \boldsymbol{C} = \boldsymbol{\sin^{-1}} \frac{(\boldsymbol{a}-\boldsymbol{x})}{\boldsymbol{r_2}}$
- Therefore, $\angle B = (90 - \angle A) + \angle C$
- Save $\angle B$ to *m.*
- Using $\boldsymbol{r_1}$ as the x-axis, rotate $\boldsymbol{r_2}$ using the equation $\boldsymbol{D} = \boldsymbol{r}(\frac{\boldsymbol{m}}{\boldsymbol{360}})$, where r is the circumference of the circle made by $\boldsymbol{r_2}$.

We have completed our movement of the arms to reach point *(a, b).*

4 Using Trigonometry to Find the Distance Away From the Object

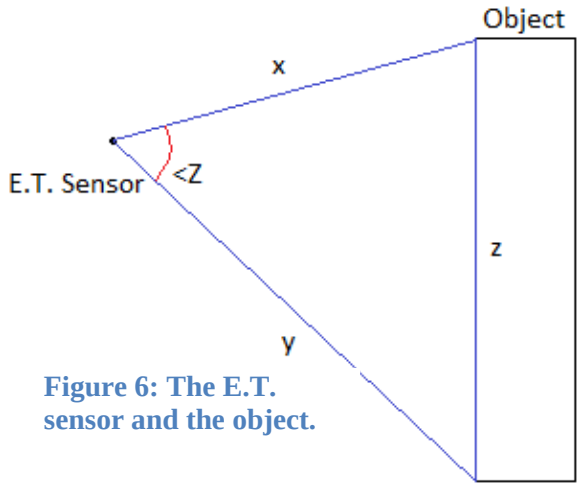

Figure 6: The E.T. sensor and the object.

Finding the Known Values

The E.T. sensor is a range distance sensor. It sends out a modulated frequency IR beam and measures the reflection to detect the distance from the sensor to an object. It returns an integer corresponding to the distance to the object. If no object is detected, a large value representing infinity is returned.

The robot spins with a mounted E.T. sensor to find the endpoints of the object. The first endpoint is the angle where the E.T. sensor first detects an object, or when the returned value dramatically decreases. The second endpoint is the angle where the sensor no longer senses an object, or when the value dramatically increases again. The detected values returned right before the value jumps or drops are saved into the variables x and y. These two variables are the distances to the endpoints of the object. The angle the robot turns between the two endpoints is $\angle Z$.

The known values x, y, and $\angle Z$ are shown in **Figure 6**. The width of the object, z, is the only unknown value.

4.1 The Law of Cosines

The law of cosines is $a^2 = b^2 + c^2 - 2bc\cos A$ [1]. By exchanging the variables, we calculate the length of the object, z:

- $z^2 = x^2 + y^2 - 2xy\cos(\angle Z)$

4.2 Benefits of Length Detection

Detecting the length of the object is useful for our navigation program, which creates a map of the game board. Previously, the function that detects blockages required bumping into the object, marking it on the map, and subsequently rerouting. Detecting the distance from the object and its length allows the program to completely avoid the object while also marking it as a blockage. This results in a faster and smarter navigation.

Knowing the length of the object is also useful when the robot must grab or move it. It allows the robot calculate how much it must open its arms or claws in order to pick it up.

5 Conclusion

Programming for robotics in the Botball game presents a real-world application of the mathematics we learn in the classroom. Most high school students understand these basic mathematical concepts, but they may not think of applying this knowledge to robotics. With circles and trigonometry, we can manipulate a jointed arm to reach any point within a semicircle. Using the E.T. sensor and trigonometry also allows us to find the width and distance from a remote object. These applications simplify our navigation and aid in obtaining game pieces. By drawing from our knowledge of mathematics, we have developed new concepts that make our programs accurate, consistent, and reusable year after year.

References

[1] Burger, Edward B., David J. Chard, Earlene J. Hall, Paul A. Kennedy, Steven J. Leinwand, Freddie L. Renfro, Tom W. Roby, Dale G. Seymour, and Bert K. Waits. *Algebra 2.* Austin, TX: Holt, Rinehart and Winston, 2008. Print.

[2] Rhoad, Richard, George Milauskas, and Robert Whipple. *Geometry for Enjoyment and Challenge*. Evanston, IL: McDougal, Littell, 1991. Print.