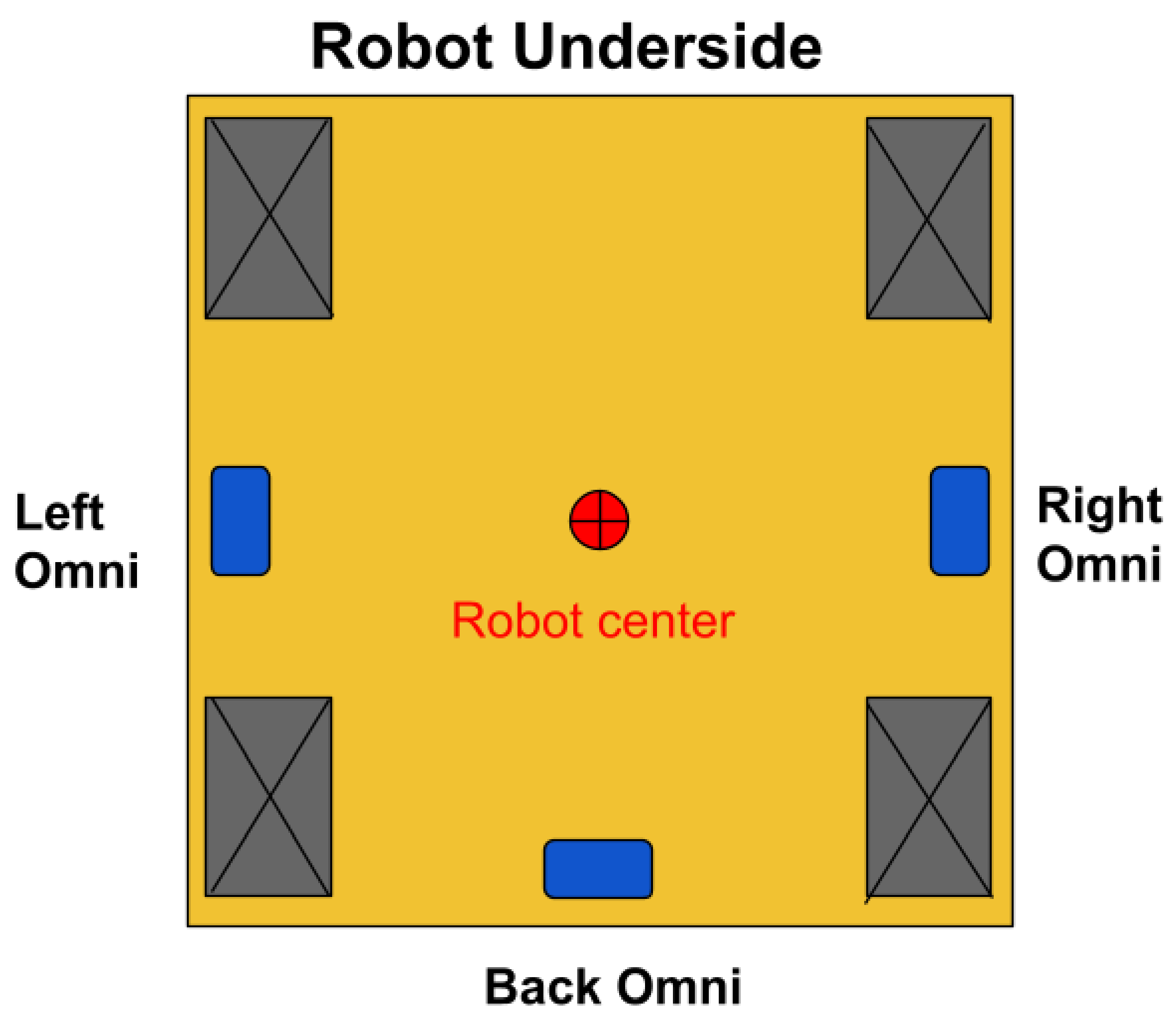
Deriving Trajectory and Coordinates Equation For Odometry System

It has been a long time interest of mine to create a control system for my school's robotics team that would ease programming by just having the programmer plot points in a coordinate system and the robot following it with high accuracy. It was until this past fall that I started working on this project, which was successfully implemented and prompted our team to achieve first place in the championship. This paper will focus on the mathematics behind odometry programming and derive the equations used to calculate the vector position, orientation, and velocity control.

WHAT IS ODOMETRY:

Odometry means using wheels to track the movement of a robot or ground vehicle. In this case, for the FTC competition, we are using encoders on omni-directional wheels mounted to the bottom of the robot. Odometry can be done in numerous ways, and has many meanings, however this will cover localization of an FTC and FRC robot. In the real world, Odometry like this is prone to inaccuracy over time so it is usually coupled with an external positioning system such as cameras or LIDAR but is not needed for the small application of the robot.

HOW IT IS SET UP:



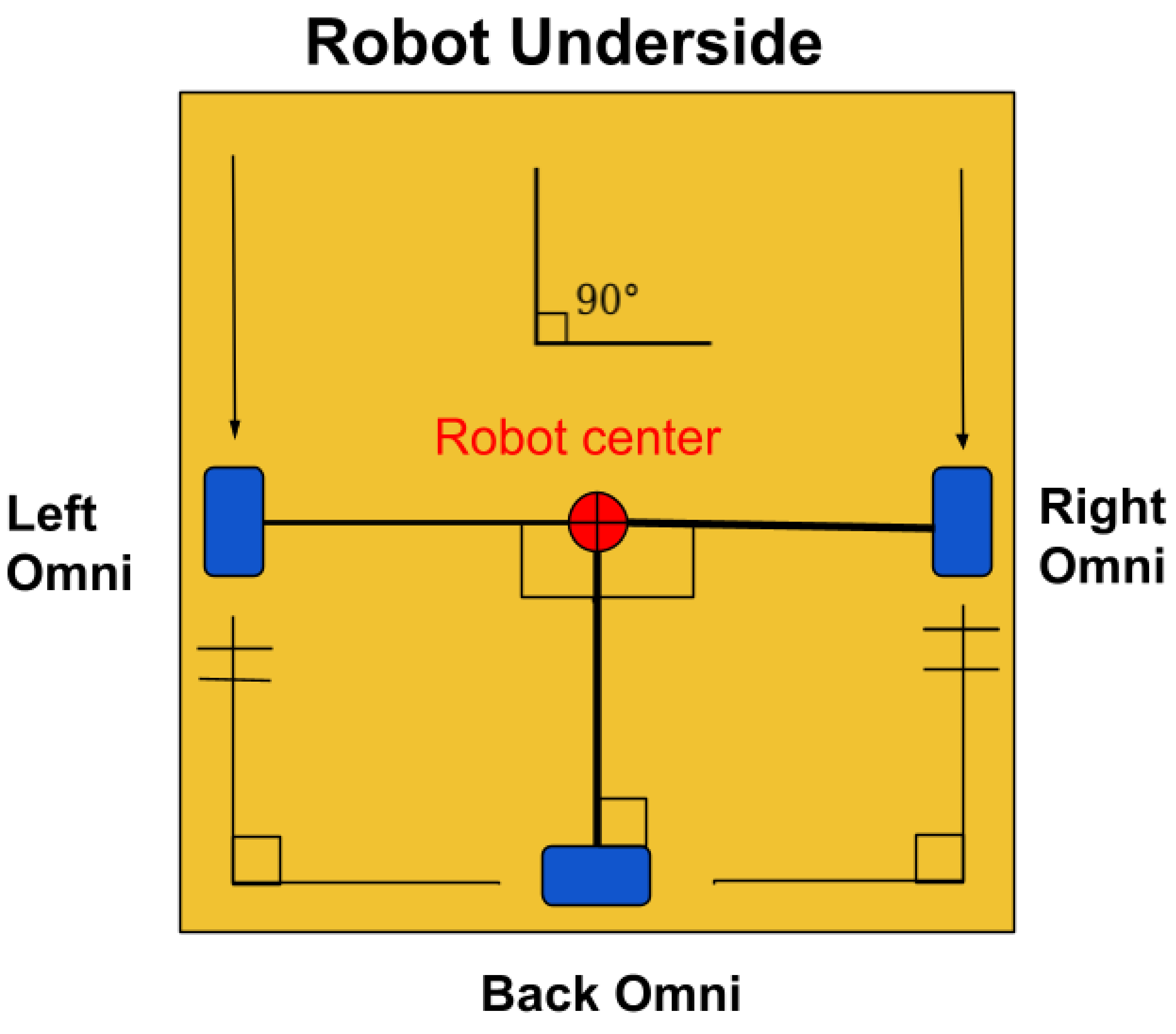
Above is a diagram showing how the omni-directional wheels(Omni's) are mounted on the robot (also known as dead wheels). Only two omins are needed if you have an alternative way of obtaining the robot's direction(heading), using an imu sensor as an example. The point is, it's needed to be able to track the robots movement in both X and Y axis, as well as its rotation. The boxes in the corners (that have X sign) represent the drive wheels but aren't important for odometry. The omnis don't have to be in the center of the robot. They just have to be arranged in a sort of T pattern. Adding a chunk to the side of the robot technically makes the omnis "off center", but does not affect odometry. Another consideration is that the further apart the omnis are mounted, the more accurate the system will be due to the higher resolution. This is because there is a chance that the measurements could be wrong due to wheel slippage and uneven terrains.

Core of the Odometry Algorithm:

At every instant where the robot position is wanted to be known, the odometer algorithm will calculate a "movement" vector (kind of like a line/arrow) that is its predicted change in position that updates from the last time it was called. The robot will provide live data on two coordinates:

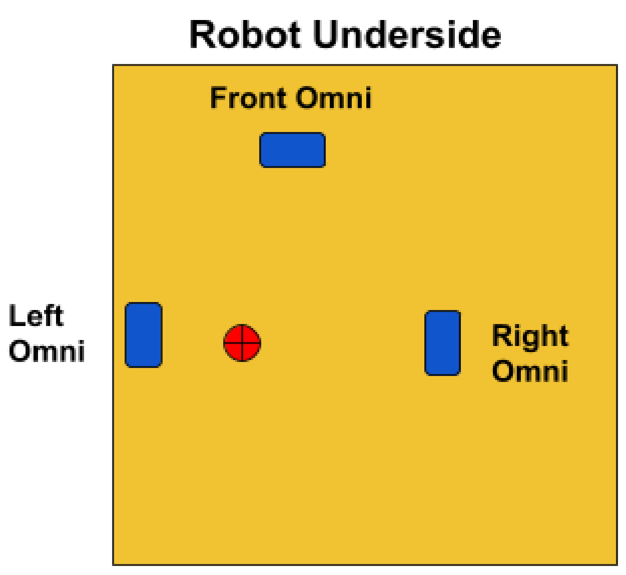
1. Position Coordinate (X,Y)
   a. The position coordinate states the exact coordinates of the robot on the playing field
2. Orientation Coordinate **(θ)**
   a. The orientation coordinate states the degree in radians of where the robot is facing relative to the playing field

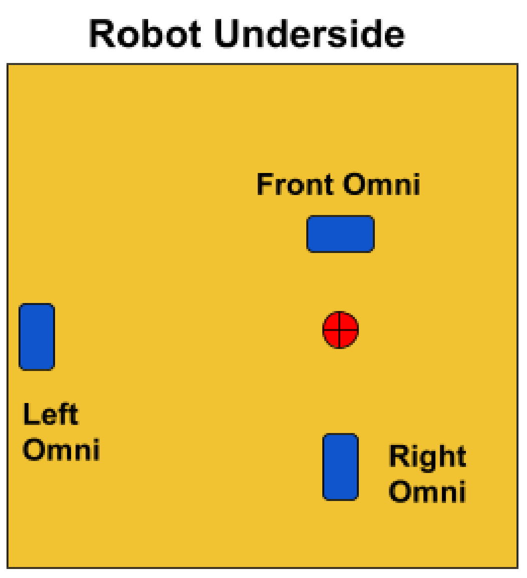
In order to derive the position and orientation coordinate, the robot's motion should be modeled as in Arc. Before completing this example, it is necessary to to understand rules for placement of the omni/dead wheels for the math to work properly. As shown above in Figure 1, all calculations will be done relative to the "Robot center" instead of tracking the whole moving along the playing field. It is important to know that the "Robot center" here happens to be the same as the tracking center. The tracking center does not necessarily have to be the robot center, but I decided to make the robot center the same as tracking center for weight distribution and achieving optimal distance apart to increase accuracy. If the robot center is not desired to be the same as the tracking center for whatever mechanical or choice based reasons, for the algorithm to succeed the following criteria needs to be met:

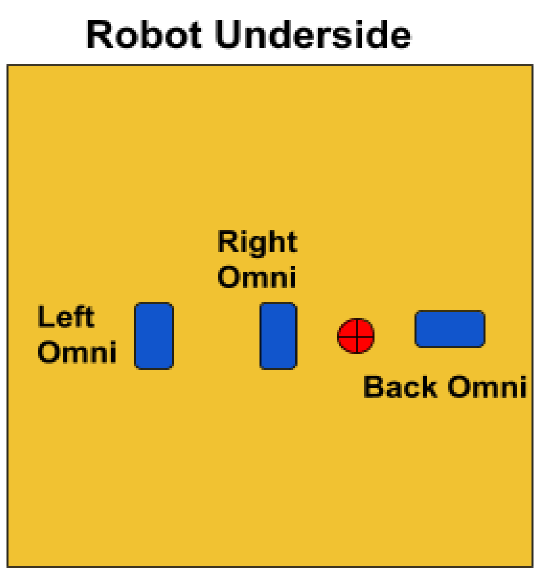


1. All Three omnis/deadwheels need to be perpendicular to the new desired tracking center
2. The Left and Right omni/deadwheel need to be parallel to one another
3. The Back omni/deadwheel needs to be perpendicular to the left and right omnis/deadwheels

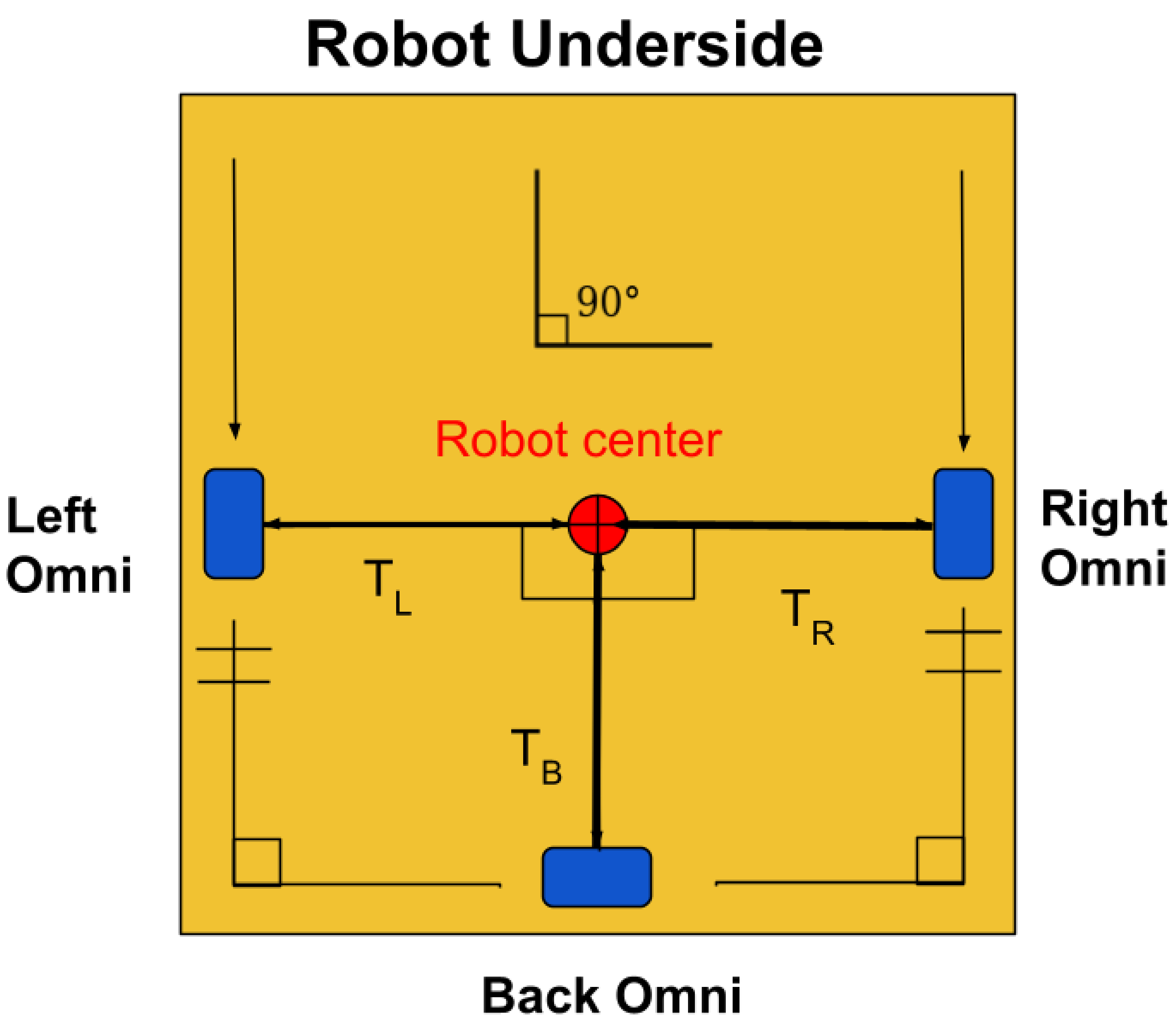
Other valid configuration examples can include but not limited to:







Here is the configuration I have on the robot:



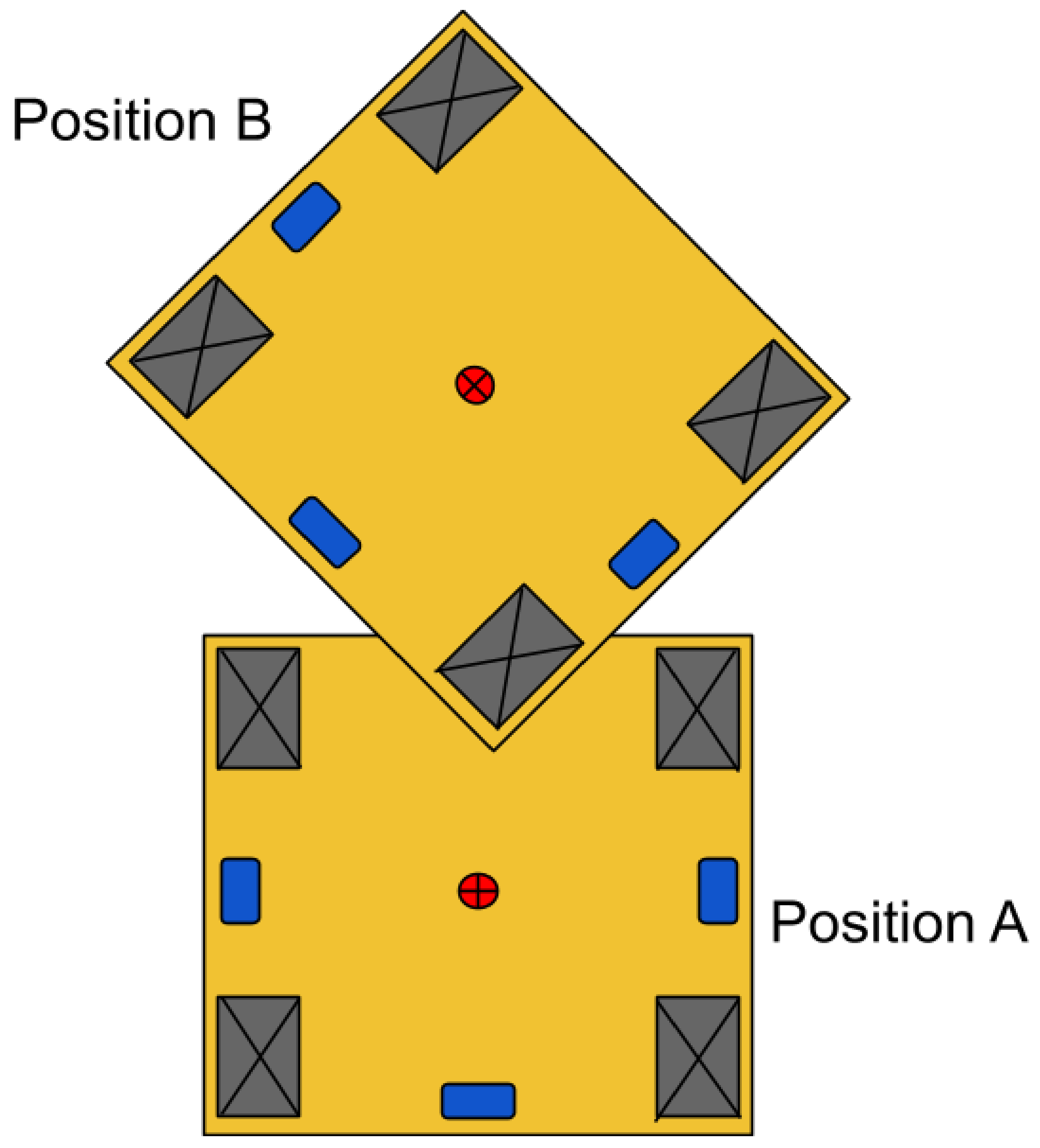
Constants:

- T_L is the distance from the left omni wheel to the center
  - T_L = 7.5 inches
- T_R is the distance from the right omni wheel to the center
  - T_R = 7.5 inches
- T_B is the distance from the back omni wheel to the center
  - T_B = 4 inches

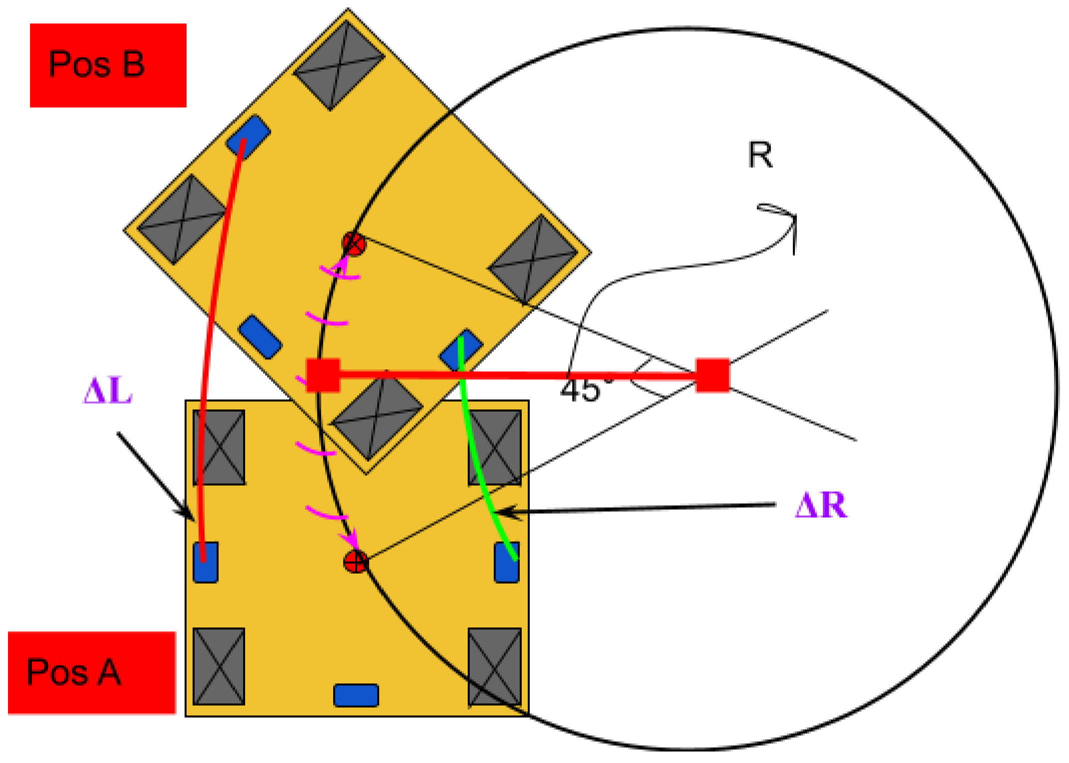
These three constants will be used to calculate the positions and orientation vectors.

<u>Tracking Robots movement in an Arc:</u>

Let's say the robot needs to move from Point A to point B in the diagram shown below.



If we draw a circle passing through both robot centers, it could be seen that the robot's path follows an Arc of 45 degrees.



The main arc in this process is the arc that both tracking sensors pass through. It is colored pink in the above figure. This is the arc that will mostly be tracked. There exists however, two other tracking arcs that are formed by this movement and need to be compensated for. The arc which the left tracking omni passes through is called the Left Arc **ΔL**, and the arc which the right tracking omni passes through is called the Right Arc, **ΔR**. In order to accomplish the delivery of the cone onto the medium junction (refer to diagram before) the left arc needs to be 20.8 inches in length and the right arc needs to be 9.1 inches in length. **ΔR** and **ΔL** represent the Arc lengths created from the initial position of right and left omni to the new position.

- **ΔL** = 20.8 inches
- **ΔR** = 9.1 inches

In order to retrieve the updated orientation of the bot, we are going to need to calculate the angle for the tracking centers arc. In order to calculate this angle, we need to use the Arc length formula:

S = r (**θ**)**,** where S is arc length, r is the radius, and **θ** is the arc angle in radians

All three arcs share the same arc angle because the arcs are concentric, meaning they share the same center of the circle but have a different radius. The criteria for configuration of the omni wheels was established so that this property is always available. This will ease the calculation process and make it better.

If we use the arc length formula, we can get that ΔL is:

$\Delta L = r\theta$, Generic form

r, the radius is equal to $R + T_L$

where R = the radius of the circle that contains arc from center of initial position to final position shown in figure above and T_L is the distance from the left omni wheel to the center .

$$\Delta L = (R + T_L) \times \theta \ ,$$

As shown above, the radius r of ΔL was substituted with $(R + T_L)$. This is because the radius of the Left arc is consistent of the circle's radius, R (the radius of the circle connecting center points of the two positions), plus the distance T_L , which is the distance from the center to the tracking omni. The movements all occur relative to the center point. The final part of the formula remains the same, where the radius is getting multiplied by the angle θ. Now here is the formula of ΔR :

$$\Delta R = r\theta$$

$$\Delta R = (R - T_R) \times \theta$$

The only difference between both formulas is the addition or subtraction of the length between omni and center. For the right arc we need to subtract the bigger circle radius from T_R because we are tracking the inner Arc.

Now we can manipulate both of our equations:

$$\Delta L = (R + T_L) \times \theta \qquad \Delta R = (R - T_R) \times \theta$$

We can divide the arc angle on both sides for both equations as following:

$$\frac{\Delta L}{\theta} = \frac{(R+T_L) \times\theta}{\theta} \qquad \frac{\Delta R}{\theta} = \frac{(R- T_R) \times\theta}{\theta}$$

Then we can isolate the circle's radius for both formulas,

$$\frac{\Delta L}{\theta} = (R + T_L) \qquad \frac{\Delta R}{\theta} = (R - T_R)$$

Now we can isolate the circle radius for both equations by subtracting T_L from Left arc equation and adding T_R on right arc equation:

$$\frac{\Delta L}{\theta} - T_L = (R) \qquad \frac{\Delta R}{\theta} + T_R = (R)$$

Since we don't know the circle's radius, both equations can now be combined. We set both equations equal to each other:

$$\frac{\Delta L}{\theta} - T_L = \frac{\Delta R}{\theta} + T_R$$

Multiply by θ on both sides:

$$\Delta L - T_L\theta = \Delta R - T_R\theta$$

Rearrange equation so that left and right arcs are on the left side:

$$\Delta L - \Delta R = T_R\theta + T_L\theta$$

Factor θ out:

$$\Delta L - \Delta R = \theta(T_R + T_L)$$

Isolate θ out by dividing left side by $(T_R + T_L)$:

Now the equation for calculating the instantaneous orientation of the robot is complete:

$$\frac{\Delta L - \Delta R}{(T_R + T_L)} = \theta$$

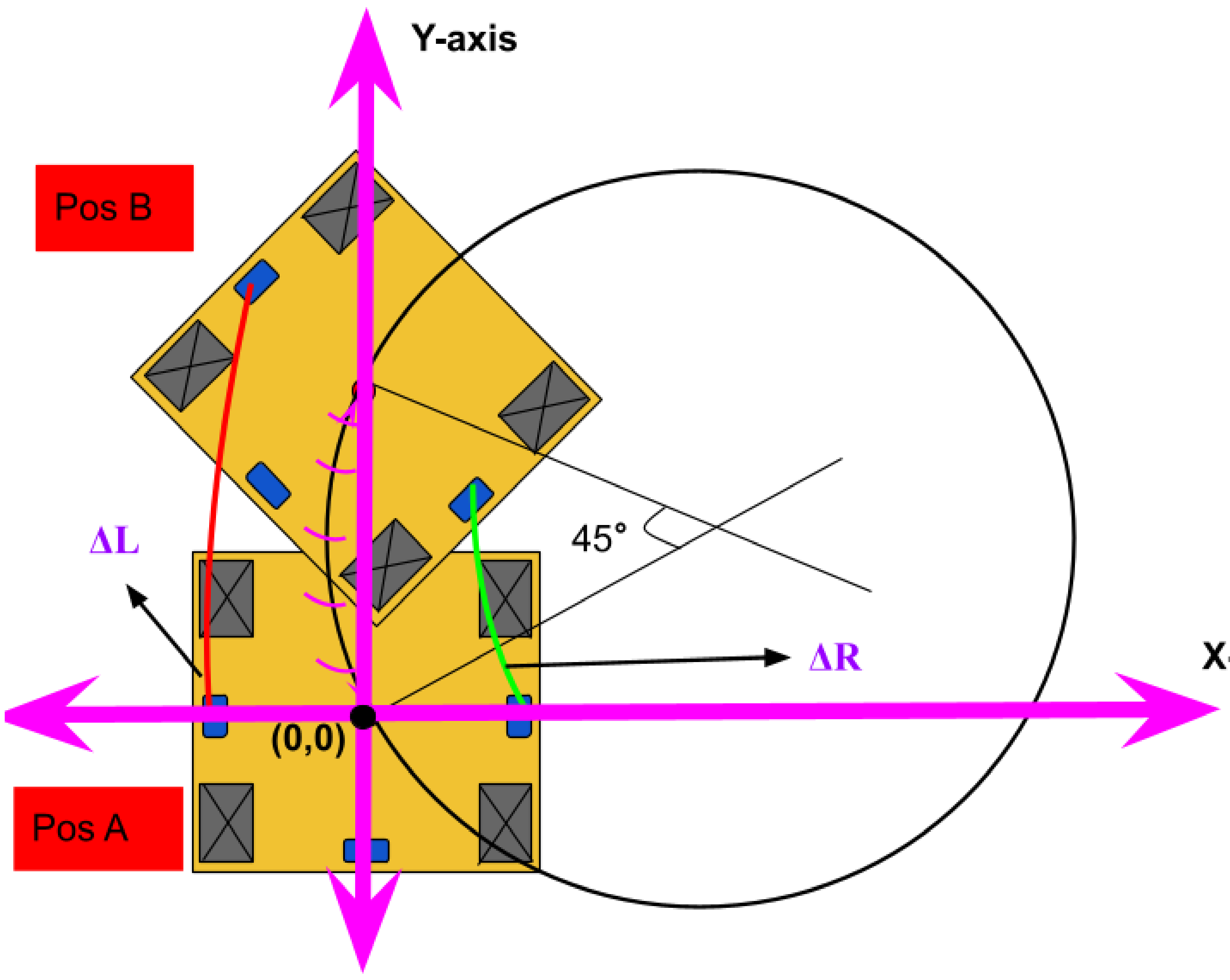
Plugging in the values for the above example, we can see we get the angle of 45 degrees:

$\theta = \frac{20.8 - 11.7}{7.5+7.5}$, note that the measurement of the arc is in inches

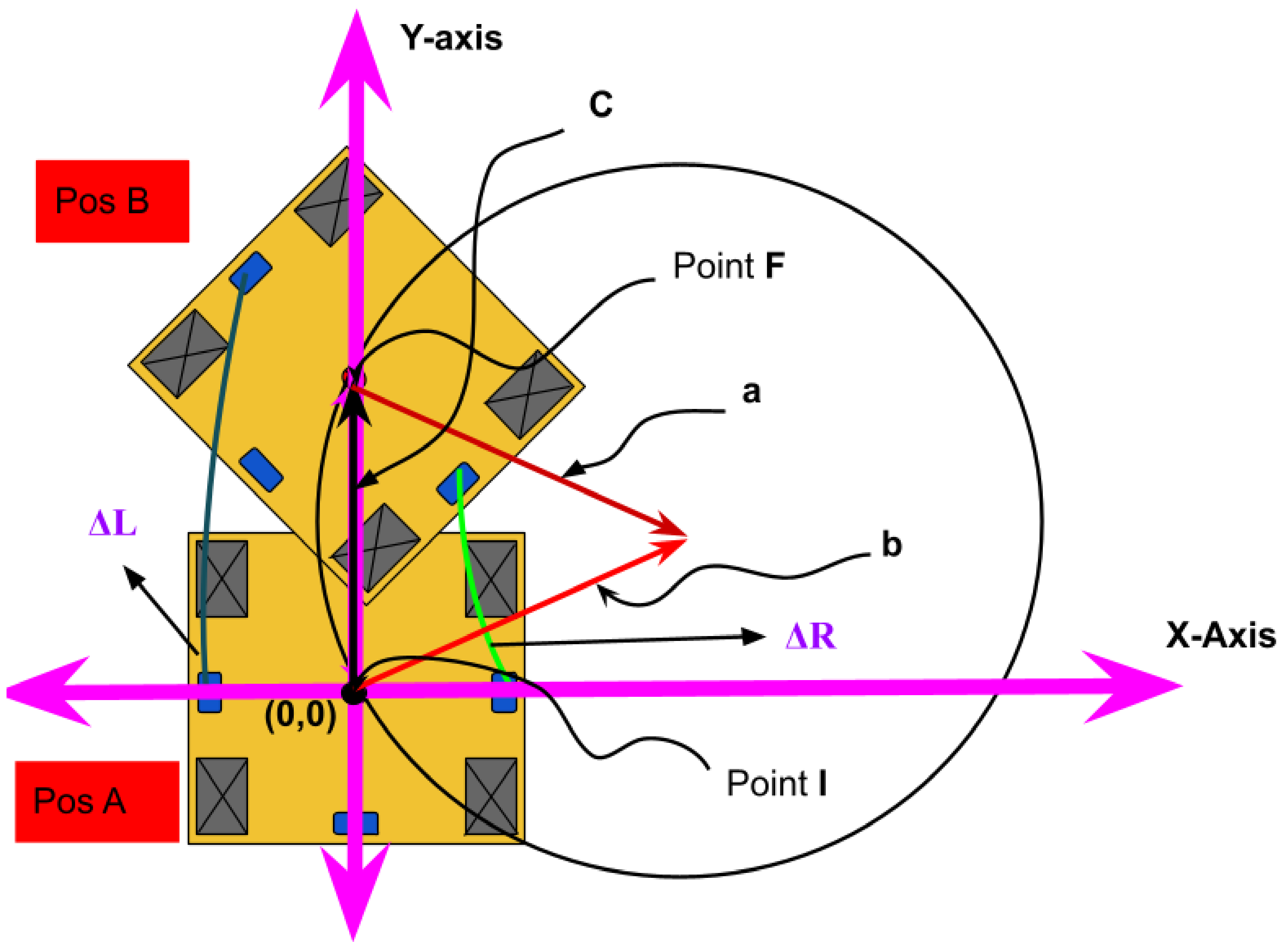
$$\theta = 0.78\,rad \approx 45\,degrees \rightarrow \qquad 0.78\,rad \cdot \frac{180}{\pi} \approx 45\,degrees$$

If we know how far each omni has moved, then the orientation of the robot can be found anywhere on the field using this equation.

The other component of this as alluded to in the beginning of the paper, is the position vector of the robot. When thinking about calculating the position of the robot, a coordinate system needs to be envisioned in order to reflect the robot's vertical and horizontal position. Since the user decides on the x and y axis placements on a graph, it is most intuitive to make the robots heading parallel to the Y- axis and align the robots tracking sensor to the coordinate system's origin. Refer to figure below to visualize this concept.



It will be assumed that the Y-axis is actually the straight line path from the initial robots tracking center (point I in figure below) to the final robot tracking center (point F in figure below).



The lengths between the two tracking centers (segment c in the above figure) will be on top of the Y-axis as shown above. Notice that in this diagram the distance traveled by the robot in the vertical direction forms an isosceles triangle of sides a, b, and c. This triangle is formed with the vertex being the center of the arc. This means that in order to find the y-coordinates of the robot all that is needed is to calculate the length of side c of the isosceles triangle. This will be achieved through using the law of cosine:

The law of cosine simply states that in a triangle a, b, and c, and Angles A, B, and C:

$$c^2 = a^2 + b^2 - 2ab(cos(C))$$

From the diagram, what we need to find is the length of c. The lengths of a, and b, are simply the arc's radius which we can

calculate with formula derived earlier for the heading angle:

$$\theta = \frac{\Delta L - \Delta R}{(T_R + T_L)}$$

Now that we know the orientation of the robot, the arc length formula can be used to calculate the arc's radius:

$$c = r\theta$$

It is known that the right arc (b) shares the same angle as the tracking center arc and the righ arcs radius (b) can be calculated with change in right tracking wheel ΔR:

$$\Delta R = r\theta$$

So the arcs radius is equal to:

$$r = \frac{\Delta R}{\theta}$$

Now that the right arc radius is known, in order to find the tracking center arcs radius all that needs to be done is to add the constant T_R which is the distance between the right tacking wheel ato the tracking center.

Therefore, tracking center radius denoted below as R is:

$$R = \frac{\Delta R}{\theta} + T_r$$

This will be both lengths a and b in the law of cosine equation, which would prompt s the ability to calculate the y coordinate when substituting for values of a and b into the equation:

$$c^2 = a^2 + b^2 - 2ab(cos(C))$$

After substitution and renaming variables, the equation would look like this:

$$y^2 = R^2 + R^2 - 2R^2 cos(\theta)$$

We now factor out the $2R^2$ from the equation:

$$y^2 = 2R^2(1 - cos(\theta))$$

This puts the equation in a position where it could be manipulated to reach a similar structure to the trig identity of the half angle formula. The half angle formula states:

$$sin(\frac{\theta}{2}) = \sqrt{\frac{1-cos(\theta)}{2}}$$

To reach this form we first need to factor a 2 from the $1 - cos(\theta)$:

$$y^2 = 4R^2(\frac{1-cos(\theta)}{2})$$

Taking the square root of both sides will make it very similar to the identity:

$$y = 2R(\sqrt{\frac{1-cos(\theta)}{2}})$$

This method was inspired by Tyler Vennes Control Engineering in FRC, cited in bibliography below.

$\sqrt{\frac{1-cos(\theta)}{2}}$ can be substituted with $sin(\frac{\theta}{2})$:

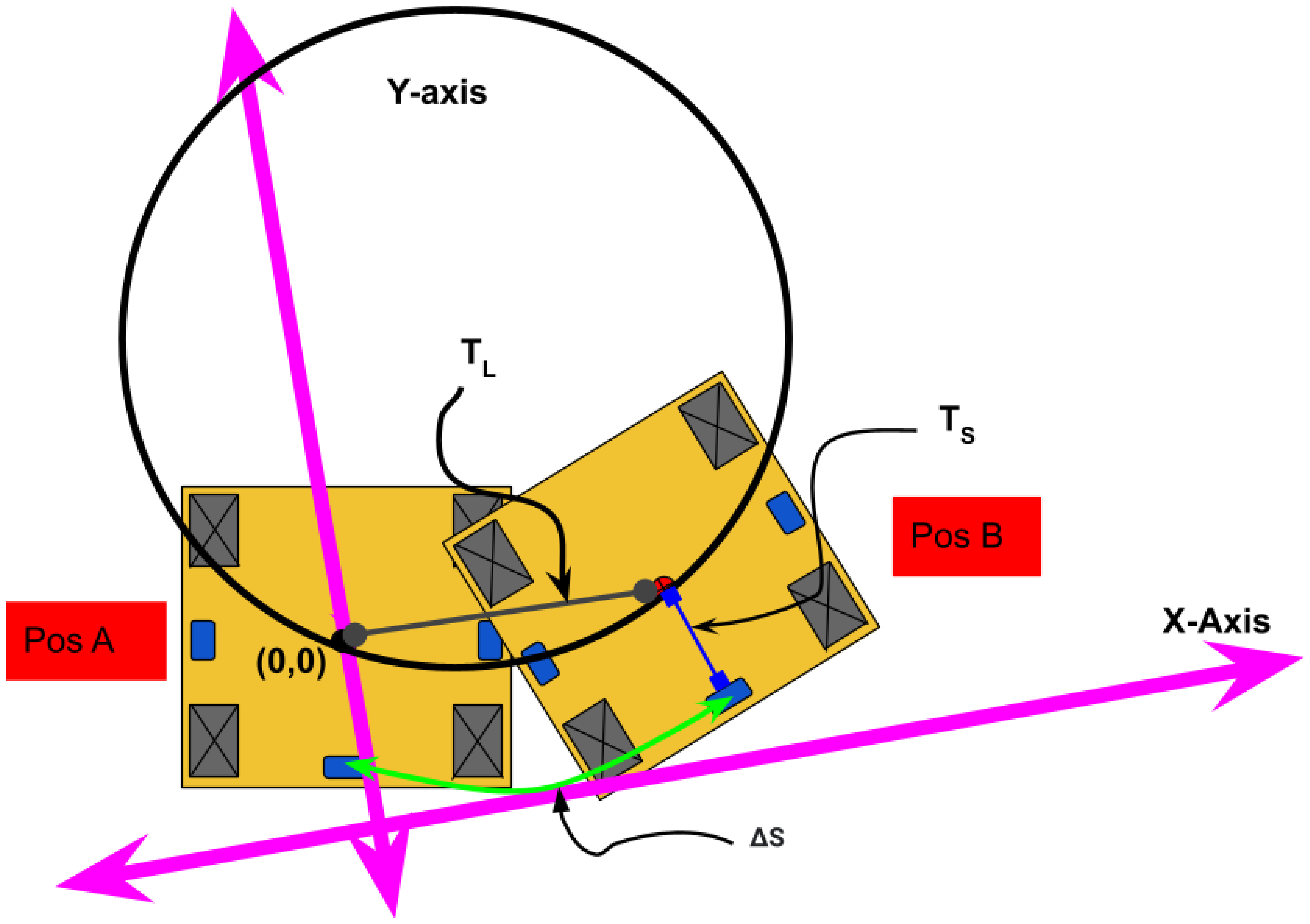
$$y = 2R(sin(\frac{\theta}{2}))$$

Coming back the the formula, the radius R can be replaced with $\frac{\Delta R}{\theta} + T_r$:

$$y = 2(\frac{\Delta R}{\theta} + T_r)(sin(\frac{\theta}{2}))$$

This is the y-coordinate equation of the vector.

Going back to the previous diagram, we can see that the shifted coordinate system and line c is parallel to the Y-axis. There isn't really any change in the X- position so the X vector would just be 0. In a scenario where the x-coordinate might deviate from the expected position, this would mean that the back tracking wheel would form another arc. To understand it visually refer to the diagram below which shows the final position of the robot with relation to change in position of backtracking wheel:



Now you can see that a new tracking arc ΔS is formed by the backtracking wheel where T_L is perpendicular to the Y-axis.

To calculate the X coordinate, we can calculate it the same way as for the Y coordinate but with a single minor change. Since the radius is now calculated by the arc formed by the backtracking wheel we will ΔS for the arc and T_S for compensating for the distance between the tracking center and the back omni.

Following the same steps for the y coordinate we can derive the x coordinate to be:

$$x = 2\left(\frac{\Delta S}{\theta} + T_S\right)\left(\sin\left(\frac{\theta}{2}\right)\right)$$

Thus, to sum things up the position equation of the robot can be written as:

$$d = 2\left(\sin\left(\frac{\theta}{2}\right)\right)\left[\left(\frac{\Delta S}{\theta} + T_S\right), \left(\frac{\Delta R}{\theta} + T_r\right)\right]$$

If we just know how far each tracking has moved (using a dead wheel encoder) we can find the position of the robot on the field since the position is simply the starting position plus the summation of all the position vectors up to that time. Instead of calculating the robot's relative position based on previous position, we can calculate it as an absolute quantity regardless of what movements the robot has done before, and because of this we can adjust for past errors and correct future movement.

Bibliography:

Odometry (2020). Game Manual 0. Retrieved April 1, 2023, from https://gm0.org/en/latest/docs/software/concepts/odometry.html#resources-for-odometry

Coordinate frames (2019). Coordinate Frames - Road Runner. Retrieved April 1, 2023, from https://acme-robotics.gitbook.io/road-runner/tour/coordinate-frame

Vennes, Tyler. (2017). *Controls Engineering in FRC - Tavsys.net*. Controls Engineering in the FIRST Robotics Competition Graduate-level control theory for high schoolers. Retrieved April 1, 2023, from https://file.tavsys.net/control/controls-engineering-in-frc.pdf