|  |  |  |  |
| --- | --- | --- | --- |
| [The Rossum Project](http://rossum.sourceforge.net/) / [Papers](http://rossum.sourceforge.net/papers/) / [Calculations](http://rossum.sourceforge.net/papers/CalculationsForRobotics/) / Circle Path | | | Revision 1.2  11 January 2006 |
| **A Path Following a Circular Arc**  **To a Point at a Specified Range and Bearing**  by G.W. Lucas (gwlucas@users.sourceforge.net) | | | |
| **Contents** | | | |
|  | [1](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm#_Introduction). | Introduction | |
|  | [2.](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm#_A_review_of_trigonometry) | A review of trigonometry | |
|  | [3.](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm#_Developing_calculations_for_the%20rob) | Developing calculations for the robot’s trajectory | |
|  | [4.](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm#_From_geometry_to_motion) | Geometry in motion | |
|  | [5](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm#_Trigonometry_at_Work). | Trigonometry at work | |
|  | [6.](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm#_Results) | Results | |
|  | [7.](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm#_Taking_a_step_backwards) | Taking a step backwards | |
|  | [8.](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePath.htm#_Conclusion) | Conclusion | |
|  |  |  |  |

**Introduction**

In this discussion, we wish to develop a calculation that describes the path of a mobile robot or vehicle given certain requirements. The robot is expected to travel from its initial position to a goal point located at a specified range and bearing. Initially, the robot is not facing its goal position directly, but must turn as it moves toward its target. We might encounter a problem of this type when processing data from a directional sensor such as a sonar device, when navigating a pre-defined floor plan, or responding to user input. To develop at calculation that meets these requirements, we will use a path that follows the arc of a circle.

Figure 1 shows an example of a situation where a path based on a circular arc would be appropriate. In this case, a robot that is already in motion detects a target and turns to approach it. The combination of both forward (linear) and rotational motion results in a path that follows a curve. What kind of curve? It depends on what we choose when we program the robot. If we specify that the linear and rotational velocities remain constant, the robot’s path will lie along the arc of a circle. It we specify that these values can change during the robot’s movement, then the robot’s path may follow a more complicated kind of curve. For this article, we will make the simpler choice. We will stipulate constant values for linear and rotational velocity and develop a path for a robot or a vehicle that follows a circular arc from a starting position to an arbitrary goal

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | |  | |  |  | |
| **Figure 1 – A moving robot detects a target at time T=0 and turns to intercept** |

Here it is useful to remember that the choice of a circle as the basis for the path is arbitrary. There are many cases where a circle does, indeed, represent the true path of a wheel-based drive system, but there are many others where it does not. Indeed, a robot using a differential steering system might simply stop, pivot in place to face the goal point, and then move directly forward. The circular path does, however, provide useful building blocks for the more general solutions to be discussed in future articles. Furthermore, the use of a circular trajectory enables the robot to follow a path without stopping and turning, a feature which may be advantageous aesthetically or operationally. Finally, it is often easy to translate the intent to follow a circular path into control inputs for an electromechanical drive system.

Table 1 provides definitions for the variables we will use in this discussion. Time is given in seconds. The specific choice of units of distance is unimportant as long as a consistent metric is used throughout. We will always treat angles as given in radians measured in a counterclockwise direction from the x-axis (some of the results depend on this treatment).

.

**Table 1 – Variables and specifications**

|  |  |
| --- | --- |
| **Variable** | **Description** |
|  | Specified initial position of robot |
|  | Specified initial orientation of robot |
| s | Specified speed of robot |
|  | Polar coordinates giving, respectively, the range and bearing to the goal position. May be specified or computed based on |
|  | Cartesian coordinates of goal position. May be specified or computed based on |
|  | Time: variable time, specified initial time, computed final time for robot’s trajectory. |
|  | Computed angle subtending the circular arc of the robot’s path. Also total change in orientation for robot when it completes the arc. |
|  | Computed radius of the circle that includes the robot’s path. |
|  | Computed center of the circle that includes the robot’s path. |
|  | Computed length for arc of circle traveled by robot or vehicle |
|  | Computed angular velocity for robot throughout its trajectory, given in radians per second. |
|  | Derived function giving orientation of robot or vehicle as a function of time. |
|  | Derived functions giving position of the robot or vehicle as a function of time. |

**A review of trigonometry**

|  |
| --- |
|  |
| **Figure 2 – Relating a triangle to a circle** |

At this point in our discussion, a quick review of a concept from trigonometry is in order. Figure 2 illustrates an important relationship between a triangle and a circle. Consider a circle of radius centered on the origin. Given a triangle with hypotenuse of length and one endpoint of the hypotenuse positioned at the origin, we can locate a point on the circle based on an angle by using the basic trigonometry functions sine and cosine where

                                                     [1]

As the angle is allowed to vary between zero and  radians (360 degrees), the endpoint of the hypotenuse traces path along the circle. When the angle equals zero, the endpoint lies on the positive x-axis. When the angle equals, the endpoint lies on the positive y-axis, etc.

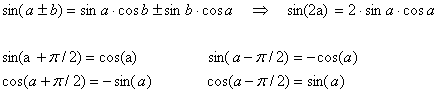
We will use this relationship between the circle and a triangle to perform double duty in the discussion below. First, it provides a way to convert between polar coordinates and Cartesian coordinates. In other words, it provides a method to compute a position relative to some reference point given a range and bearing. Second, it allows us to develop an equation that describes a circle as a function of angle. Once we have that, the only thing we need to model the robot’s path is a relationship that gives us angle as a function of time. By combining these two ideas, we can predict the robot’s time-dependent position as it follows a path along a circular arc. Thus the relationship illustrated by Figure 2 provides the key to developing the calculation we seek for this problem.

Equation [1] provides a way of converting polar coordinates to Cartesian coordinates. Its opposite conversion is given by



The angle is calculated by applying the inverse tangent (arc tangent) function to the quotient of the y and x-coordinates. For computational purposes, we have to pay attention to the special case where the x-coordinate is zero or very close to zero. Most major computer programming environments provide a function called ATAN2 that takes x and y values as separate arguments and handles the special cases.

In these notes, we will also use the following trigonometric identities:



**Developing calculations for the robot’s trajectory**

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | |  | |  |  | |
| **Figure 3 – Trajectory based on an arc of a circle** |

Figure 3 illustrates the trajectory that we will develop in this section. Note that, for now, we do not care about the final orientation of the robot, but simply wish to find a way to steer it to its goal point. Clearly, if the goal were directly in front or in back of the robot, the simplest path to that point would be a straight line. Assume that it is not and that we wish to develop a path based on a circular arc. It is worth noting that, as a practical matter when writing programs for robotics, some of the equations developed below include division-by-zero problems when the goal is directly in front of the robot. Appropriate logic should be included to handle the special case.

If the goal point is behind the robot, the shortest path to the point would be for the robot to travel backwards. We will examine this case later, but for now, assume that we wish for the robot to always move to its front. If the goal is behind the robot, it will travel to that point by looping around and going the long way.  Finally, we need to make one more assumption. Assume that the robot’s motion is always directly forward and that its orientation changes at a fixed rate so that it is tangent to the circle at all points of the trajectory. To construct the circle, we need to find the value for its radius, which is given by. Based on our assumption of tangency, the center of the circle will be located*, in the robot’s initial frame of reference*, at the position given by the Cartesian coordinates. The robot path follows the arc of the circle subtended by angle.

By definition of a circle, the robot’s distance from the center of the circle remains constant. Using this information, we can construct an isosceles triangle as shown in Figure 3 with two sides of length  and one of length, the measured length to the target.  To solve for the radius of the circle we use the Law of Sines from trigonometry to obtain:

                                      [2]

From inspection, we see that the sum of angles *R* and  form a right angle ( when expressed as radians). Because the sum of the angles of a triangle is 180 degrees ( when expressed as radians) we find:



and



thus

                                              [3]

This result proves useful in practical applications so often that I will restate it for emphasis.

*The angle subtended by the robot’s path along a circular arc is twice that of its initial bearing to the goal point.*

Substituting [3] into [2] and applying the appropriate trigonometric identities, we have



thus

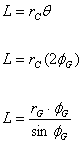
                                      [4]

The value  for the radius of the circle is essentially the *turn radius* for the robot’s motion. Note that this value is signed and can be negative. We will exploit this fact below.

When the target is directly in front of the robot, the angle of the bearing is zero and the radius of the circle approaches infinity. That is consistent with our observation that, when the goal point is directly in front of the robot, its path will follow a straight line. So, assuming that the path is not a straight line, let’s consider how the sign of the radius relates to the direction of the turn. If the value for  is positive, it indicates that the robot is turning left. If the value is negative, it indicates that the robot is turning right.

Knowing the magnitude of the turn radius is particularly useful because it may indicate that our robot or vehicle would encounter real-world limitations in performance. Naturally, vehicles with Ackerman steering have a fixed minimum radius that they can execute in a turn. There is also the possibility that the robot may tip over or at least lose traction and spin out if it tries to execute too tight a turn at too high a speed. When a vehicle travels in a circular path, its direction is constantly changing.  This change in direction requires a constant acceleration inward toward the center of the circle. This value, the *centripetal acceleration*, is given by the relationship (a result from elementary physics). As the radius becomes small, the acceleration value becomes large enough to be a problem. If you know something about the stability of your robot, you can use this information to determine if it is operating within safe parameters. You may, for example, decide that the robot needs to slow down in order to keep the acceleration value from becoming too large. Of course, you have to be careful. The force due to braking while the robot is following a circular path is orthogonal to the direction of the centripetal acceleration. As such, it increases the magnitude of the vector sum of the forces acting on the robot's wheels. So braking during the turn could increase the total force acting on the wheels to the point where the robot loses traction. Driving a robot is a bit like driving an automobile... the appropriate time to hit the brakes is *before* your enter a turn.

Recall that by definition an angle expressed in radians is related to the length of the circular arc it subtends. Thus, the length of the path that the robot travels, *L*, is simply the product of the angle, *given in radians*, and the circular radius, which can be expressed variously as:

                                                   [5]

Before we apply equation [5], a certain amount of bookkeeping is required. The bearing provided as an input to this calculation could be in any form. If it came from a sensor or user data entry, it might very well be in the range 0 to 360 degrees which, of course, we would convert to a value in the range. Mathematically, any representation of an angle is correct. For practical calculations, however, it is useful to treat the angle as a *signed* value in the range  (-180° to 180°). This restriction makes sense because we are interested in only one particular representation of the robot’s trajectory: the shortest path in the forward direction. Also note that by coercing our input to this particular range, we will ensure that the sign of angle is the same as that of and, therefore, the value of the path length *L* is positive.

Finally, we might wonder about the robot’s final orientation when it completes its transit between the two points. Simple geometry shows that the change in orientation is just the angle  subtended by the arc of the circular path. This result is useful so often that it too warrants being restated for emphasis:

*If a body follows a trajectory defined by an arc of a circle, its change in orientation is equal to the angle that subtends that arc. This observation is true for all points on the body.*

Note that this rule applies in cases such as the differential and Ackerman steering systems where the displacement of the robot is coupled to its change in orientation. It would not apply to a system such as the omni-directional steering mentioned in earlier articles where the displacement and orientation are independent of each other.

**Geometry in Motion**

So far we’ve considered the geometry of a circular path for a robot or vehicle. Now we turn our attention to its motion. The essential difference between a problem in geometry and a study of motion is the idea of time. Constructing a path that lies on the arc of a circle is a matter of pure geometry. It has no relation to time. The path that we specify today will still be the same path tomorrow. A trajectory, however, is a different matter. Each point in a trajectory is associated with a time and, perhaps, a set of velocity components. So the next step in developing model for the motion of a robot or vehicle along a path lying on the arc of a circle is finding a way to match each point in that path with a time.

Let’s begin our consideration of motion by determining how long it will take the robot to travel from its initial position to its goal. The robot’s speed is supplied as a specification for the problem.  We computed the length of the path that it travels in equation [5] above. So the time required for the robot to travel from its initial point at time to its final point at time  is easily computed by dividing the length of its path by the speed at which it travels:



We’ve already established that as the robot follows the circular path, its orientation changes at a steady rate by angle. Given that information, we can compute its rotational velocity as



Note that in the event of a right turn, we need to assigna negative value.

Our knowledge of the rotational velocity leads directly to a function defining the robot’s orientation as a function of time.

                                        [6]

**Trigonometry at Work**

At this point, we are ready to derive compact functions giving the robot’s position as a function of time. In this article, we will do so using trigonometry. It is actually easier to find these functions using calculus, but one of our goals is to stick to elementary mathematics. Therefore we will use trigonometry and leave the calculus to a separate discussion (see [Supplemental Article](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CirclePathWithCalc.htm)).

Figure 4 illustrates the general approach. First, taking the robot’s initial orientation  and the circular radius, we use equation [1] to find the position of the center of the circle. With that, we can again use [1] to find an equation that gives the robot’s position on that circle as a function of its orientation. And since we have shown how orientation can be computed on the basis of time, we can use that knowledge to find a calculation for position as a function of time.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | |  | |  |  | |
| **Figure 4 – Relation between initial point and circle.** |

Let’s derive a set of functions for the robot’s position based on the ideas given above. Recall that we have already established the following parameters describing the robot’s behavior: the distance it travels, its turn radius, its speed (given as a specification), its rotational velocity, and the time required for its transit. Refer to Figure 4. If the robot is turning to the left, then the center of the circle is at a right angle to the left of initial orientation, or. Applying equation [1], we find the coordinates for center of the circle to be:



If the robot is turning to the right, then the center of the circle is 90 degrees to the right of its orientation, or. Recall that when the robot is turning to the right, the value computed for the turn radius  is negative. So we have:



Regardless of the sign of, we can apply trigonometric identities to both calculations for the coordinates of the circle’s center leading to the following statements

                                    [7]

Figure 4 shows the initial position of the robot with an orientation of  radians. We can think of its position as being given in polar coordinates measured from the center of the circle. The bearing from the robot’s initial position to the circle’s center point is just 180 degrees off from the bearing from the circle’s center point to the robot’s initial position. So, for a left turn, we have the robot’s position given as a function of time as



**Results**

As in the derivation for equation [7], a symmetrical expression will hold for the right turn, with both equations simplifying to:

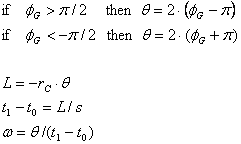
                     [8]

With this result, we have all the calculations we need to track the position of our robot or vehicle as it follows a path defined by an arc of a circle.

**Taking a step backwards**

So far, we have ignored the possibility that it might be better for the robot to travel backwards rather than forward. For example, if the goal is slightly behind the robot, it might be better for it to stop and back up a bit rather than maintaining its forward motion and making a nearly 360 degree turn. Of course, moving backwards will not always be appropriate. The choice of whether or not to do so depends on the application.

If we elect to set the robot on a backward trajectory, the basic geometry shown in Figure 3 still applies. We do not change the value of the turn radius, but we do adjust  to reflect the fact that the robot will be traveling a shorter distance in the opposite direction.



**Conclusion**

The circular trajectory described in this document has several limitations.  It does not allow you to specify the final orientation of the robot and does not consider the effects of acceleration. A more ambitious technique, which does address these issues, will be discussed in the next article in this series (see [A Path Based on Third-Degree Polynomials](http://rossum.sourceforge.net/papers/CalculationsForRobotics/CubicPath.htm)).

Nevertheless, there are a substantial number of small, mobile robots that operate using little more than the calculations described in this webpage. The key is to recognize the limits of the methods presented here and to combine that knowledge with data from real-world measurements and trials. With the right combination of theory and practice, this simple model of a trajectory based on an arc of a circle can lead to a robot capable of navigating successfully in a variety of environments.

**Acknowledgements**

I am very much indebted to Sarah Burton and Mitch Berkson, both of whom offered suggestions and corrections that substantially improved the quality of this article. Thank you both for your time and effort.

I also gratefully acknowledge the support of the research and development division of [Sonalysts, Inc](http://www.sonalysts.com/), for the use of computational resources in the creation of this material.

And, as always, I extend my thanks to [SourceForge](http://sourceforge.net/) for hosting The Rossum Project website and for their deep commitment to open-source software in general.