Path Tracking for a Miniature Robot

By

Martin Lundgren

Excerpt from Master's thesis 2003
Supervisor: Thomas Hellström

Department of Computing Science
Umeå University
Sweden

1 Path Tracking

Path tracking is the process concerned with how to determine speed and steering settings at each instant of time in order for the robot to follow a certain path. A path consists of a set of points representing the positional coordinates of a particular route. Often when implementing a path tracking algorithm, one also have to implement a path recording unit responsible for saving all the coordinates that constitutes the path. A human operator then has the possibility to manually steer the robot along some track while the path recording unit saves the information about the path. The path tracking algorithm also has to handle unplanned positional or orientation deviations from the path. Such deviations can be caused by odometric errors of some kind, or by new obstacles occurring on the path that must be avoided.

There are many different types of path tracking algorithms available today. I have chosen to implement and evaluate three of the them on the Khepera robot; follow-the-carrot, pure pursuit and vector pursuit. The first two methods has been around for quite a while now, while vector pursuit or screw tracking as it's also called is relatively new on the scene. The big difference between these methods is that vector pursuit uses information about orientation at the look-ahead point, while the others don't.

1.1 Follow-the-carrot

This algorithm is based on a very simple idea. Obtain a goal-point, then aim the vehicle towards that point [10]. Figure 7 describes the basics behind this method.

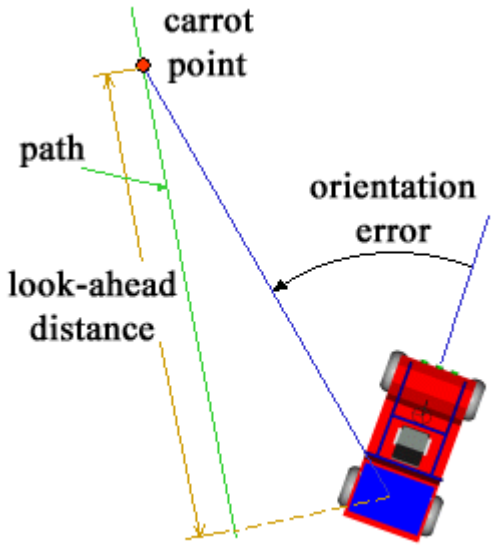

Fig 7. Follow-the-carrot

A line is drawn from the centre of the vehicle coordinate system perpendicular to the path.

The carrot point, or goal point, is then defined to be the point on the path a look-ahead distance away from the intersection point of this line. The most important parameter is the orientation error, defined to be the angle between current vehicle heading and the line drawn from the centre of the vehicle coordinate system to the carrot point. A proportional control law then aims at minimizing the orientation error between the vehicle and the carrot point. An orientation error of zero means the vehicle is pointing exactly towards the carrot point. The magnitude of a turn φ is decided by:

$$\varphi = k_p * e_o$$

where k_p is the proportional gain and e_o is the orientation error. One also hade the possibility of increasing the accuracy of the controller, perhaps adding integrative or derivative functions to it [11]. However, simple proportional controllers are still the most frequently used in this algorithm.

Although the follow-the-carrot approach is easy to understand and very simple to implement, it has a couple of major drawbacks. Firstly, the vehicle has a tendency to naturally cut corners. This happens because the vehicle immediately tries to turn towards each new carrot point. Another drawback with this path tracking technique is that the vehicle could oscillate about the path, particularly in the case of small look-ahead distances or at higher speeds. There are however modifications that can be made to the algorithm to increase its efficiency and accuracy. The selection of steering angle can be modified to be based on both the positional error displacement perpendicular to the path and the orientation error [11]. Yet the disadvantages of this method are generally to high, making it useless for implementation in new projects that require good tracking ability. Still it is very suitable for educational purposes, or for comparison with other tracking algorithms.

1.2 Pure Pursuit

The concept of the pure pursuit approach is to calculate the curvature that will take the vehicle from its current position to a goal position [3]. The goal point is determined in the same manner as for the follow-the-carrot algorithm. A circle is then defined in such a way that it passes through both the goal point and the current vehicle position. Finally a control algorithm chooses a steering angle in relation to this circle. In fact, the robot vehicle changes its curvature by repeatedly fitting circular arcs of this kind, always pushing the goal point forward.

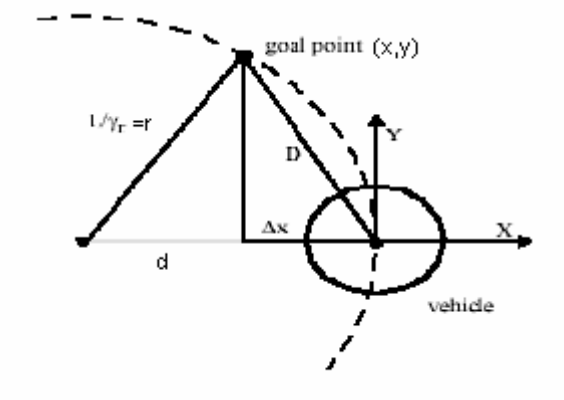

Fig 8. Pure Pursuit approach

It is important to note that the description of the pure pursuit algorithm in figure 8 is shown in vehicle coordinates. The vehicle coordinate system is defined where the y-axis is in the forward direction of the vehicle, the z-axis is down and the x-axis forms a right-handed coordinate system. Therefore all coordinates used must first be transformed to vehicle coordinates in order for the algorithm to work properly. Luckily it is pretty straight forward to converts coordinates located in one system into its representation in another system [9]. Let (xr,yr) be the current position of the robot, and (xg,yg) the goal point to be converted into vehicle coordinates. Then

$$xg_v = (xg - xr)\cos(\Phi) + (yg-yr)\sin(\Phi)$$
$$yg_v = -(xg - xr)\sin(\Phi) + (yg-yr)\cos(\Phi)$$

where (xgv,ygv) is the goal point in vehicle coordinates and Φ is the current vehicle heading. In the figure above D is defined to be the distance between current vehicle position and the goal point. Δx is the x offset of the goal point from the origin, and $1/\gamma_r$ is the radius of the circle that goes through the centre of the vehicle and the goal point. The required curvature of the vehicle is computed by:

$$\gamma_r = 2\Delta x/D^2$$

The derivation of this formula is based on just two simple equations [5]:

$$1)\ x^2 + y^2 = D^2$$
$$2)\ x + d = r$$

(x,y) being the coordinates of the goal point in figure 8 above. The first equation is a result of applying Pythagoras' theorem on the smaller right triangle in the same figure, and the second equation comes from summing the line segments on the x-axis. The following derivation is pretty straightforward and should not be that difficult to understand:

$$d = r - x$$
$$(r - x)^2 + y^2 = r^2$$
$$r^2 - 2rx + x^2 + y^2 = r^2$$
$$2rx = D^2$$
$$r = D^2/2x$$
$$\gamma_r = 2x/D^2$$

The last step comes from the mathematical fact that a circle has a constant curvature which is inversely proportional to its radius.

The name *pure pursuit* comes from our way of describing this method. We paint a picture in our minds of the vehicle chasing this goal point on the path a defined distance in front of it - it is always *in pursue* of the point. This can also be compared to the way humans drive their cars. We usually look at some distance in front of the car, and pursue that spot.

Assuming that the points that constitutes the path and the successive positions of the robot vehicle belongs to the same coordinate system, the pure pursuit algorithm can be described by these simple steps:

1. Obtain current position of the vehicle
2. Find the goal point:
   2.1. Calculate the point on the path closest to the vehicle (xc, yc)
   2.2. Compute a certain look-ahead distance D
   2.3. Obtain goal point by moving distance D up the path from point (xc,yc)
3. Transform goal point to vehicle coordinates
4. Compute desired curvature of the vehicle $\gamma = 2\Delta x/D^2$
5. Move vehicle towards goal point with the desired curvature
6. Obtain new position and go to point 2

The pure pursuit technique shows better results than the follow-the-carrot method described earlier. One improvement is less oscillations in case of large-scale positional and heading errors, another is improved path tracking accuracy at curves. Because of the advantages this method is more frequently used in real world applications than the follow-the-carrot algorithm. Other optimized methods may show better tracking results in some areas, but overall this is yet the best algorithm available today.

1.3 Vector Pursuit

Vector pursuit is a new path tracking method that uses the theory of screws first introduced by Sir Robert S. Ball in 1900. Screw theory can be used to represent the motion of any rigid body in relation to a given coordinate system, thus making it useful in path tracking applications. Any instantaneous motion can be described as a rotation about a line in space with an associated pitch. Screw control was developed in an attempt to not only have the vehicle arrive at the goal point, but also to arrive with the correct orientation and curvature. The methods described earlier, follow-the-carrot and pure pursuit, does not use the orientation at the look-ahead point. However vector pursuit uses both the location and the orientation of the look-ahead, giving it an advantage over the other algorithms as it ensures that the vehicle arrives at the current goal point with the proper steering angle.

A screw consists of a centreline and a pitch defined in a given coordinate system. The centreline can be defined by using Plücker line coordinates. A line can be represented using only two points represented by the vectors $\underline{\mathbf{r}}_1$ and $\underline{\mathbf{r}}_2$. This line can also be defined as a unit vector $\underline{\mathbf{S}}$, in the direction of the line and a moment vector $\underline{\mathbf{S}}_0$, of the line about the origin. This representation can be seen in figure 9.

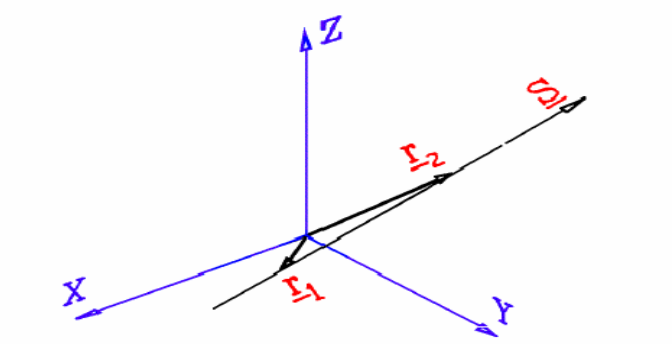

Fig 9. A line represented by two vectors

Based on the Plücker line representation we then have:

$\underline{\mathbf{S}} = \underline{\mathbf{r}}_2 - \underline{\mathbf{r}}_1 / | \underline{\mathbf{r}}_2 - \underline{\mathbf{r}}_1 |$

$\underline{\mathbf{S}}_0 = \underline{\mathbf{r}}_1 \times \underline{\mathbf{S}}$

where the vectors ($\underline{\mathbf{S}}$; $\underline{\mathbf{S}}_0$) are the Plücker line coordinates of this line. Then by defining $\underline{\mathbf{S}} = [L, M, N]^T$ and $\underline{\mathbf{S}}_0 = [P, Q, R]^T$, with $\underline{\mathbf{r}}_1 = [x_1, y_1, z_1]$ and $\underline{\mathbf{r}}_2 = [x_2, y_2, z_2]$ we have:

$$L = \frac{x_2 - x_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 (z_2 - z_1)^2}}$$

$$M = \frac{y_2 - y_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 (z_2 - z_1)^2}}$$

$$N = \frac{z_2 - z_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 (z_2 - z_1)^2}}$$

and

$$P = y_1 N - z_1 M$$
$$Q = z_1 L - x_1 N$$
$$R = x_1 M - y_1 L$$

As explained earlier any instantaneous motion of a rigid body can be described as a rotation about a line in space with an associated pitch. Figure 10 shows such a body rotating with an angular velocity ω, about a screw $\underline{\mathbf{S}}$ defined by its centreline ($\underline{\mathbf{S}}$; $\underline{\mathbf{S}}_0$). The pitch of the screw is also shown in the figure.

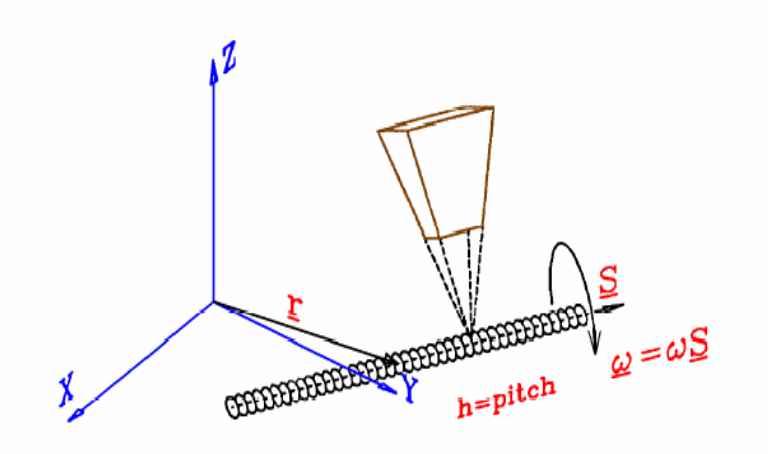

Fig 10. Instantaneous motion about a screw

The velocity of a rigid body depends on both the velocity due to the rotation and the translational velocity due to the pith of the screw. This velocity is represented as:

$$\omega\underline{\$} = (\omega\underline{\mathbf{S}} \; ; \; \omega\underline{\mathbf{S}}_{oh}) \qquad (4.3.1)$$

where

$$\underline{\mathbf{S}}_{oh} = \underline{\mathbf{S}}_o + h\underline{\mathbf{S}} = \underline{\mathbf{r}} \times \underline{\mathbf{S}} + h\underline{\mathbf{S}}$$

and $\underline{\mathbf{r}}$ is any vector from the origin to the centreline of the screw.

The vector pursuit method developed by Wit (2001) calculates two instantaneous screws. The first screw accounts for the translation from the current vehicle position to the look-ahead point, while the second screw represents the rotation from the current vehicle orientation to the orientation at the look-ahead point. These two screws are then added together to form the desired instantaneous motion of the vehicle. This information is then used for calculating the desired turning radius that will take the vehicle from its current position to the goal point.

Pure translation of a rigid body is defined as the motion about a screw with an infinite pitch, and therefore (4.3.1) becomes $v\underline{\$} = (0 \; ; \; v\underline{\mathbf{S}})$. Pure rotation on the other hand is defined by the motion about a screw with a pitch equal to zero, and (4.3.1) reduces to $\omega\underline{\$} = (\omega\underline{\mathbf{S}} \; ; \; \omega\underline{\mathbf{S}}_o)$. Then depending on whether the nonholonomic constraints of the vehicle are initially ignored or not, the screws with respect to translation and rotation are calculated and summed up to form the desired instantaneous motion of the vehicle.

The results of the testing performed by Wit [3] both on real vehicles as well as in simulation shows comparable tracking results to the other methods described earlier. Vector pursuit showed particularly good skills in handling jogs appearing suddenly in the middle of the path. This means the method is doing rather good jumping from a small error in position and orientation to fairly large errors. Similar situations can occur in the real world when the robot vehicle detects sudden obstacles appearing on the path, forcing the vehicle to drive around the obstacle and then return to the path. It can also occur if there is noise in the position estimations, which can happen if the vehicle was using GPS techniques for localizing. In other test cases vector pursuit shows comparable results to the pure pursuit algorithm. This is not so surprising, because both methods calculates a desired radius of curvature and the centre point of rotation when tracking a curved path.

1.4 The look-ahead distance

Common to all these methods is that they use a look-ahead point, which is a point on the path a certain distance L away from the orthogonal projection of the vehicles position on the path. Path tracking techniques that uses this look-ahead point are called geometric algorithms. Changing the look-ahead distance can have a significant effect on the performance of the algorithm. Increasing the value L tends to reduce the number of oscillations, thus ensure a smooth tracking of the path. However this will also cause the vehicle to cut corners, severely reducing the tracking precision of curvy paths. The effects of changing this parameter will in

the end be a question of in what context you want the tracker to provide good results. There are two problems that need to be considered:

I. Regaining a path
II. Maintaining the path

The first problem occurs when the vehicle is way off the path, thus having large positional and orientation errors, and is trying to return to this path. Under this circumstances it is pretty obvious what the effects will be of changing the look-ahead distance. A larger value causes the vehicle to converge more smoothly and with less oscillations. On the other side, once back on the path a large value will lead to worse tracking, especially in the case of paths containing very sharp curves. So a difficult trade-off must often be made when choosing the look-ahead distance for a particular path tracking algorithm. Do I expect the vehicle to encounter obstacles while in tracking mode, causing large positional errors? Or is it more likely the path will be obstacle free but rather curvy? These are questions that must be considered before implementing any systems of this kind.

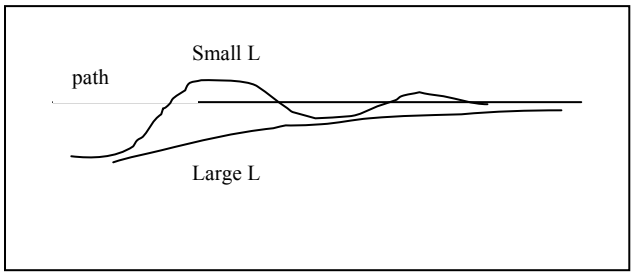

Fig 11. The effects of having a small look-ahead distance contra a large for problem 1

Another factor that must be considered when choosing look-ahead distance L is the vehicle speed. An increase in vehicle speed would also require the distance L to be increased. The reason for this is that higher speeds requires the vehicle to start turning at an earlier stage. Generally the vehicle always starts turning before the curve at every look-ahead distance larger than zero. This is a positive quality that compensates for the time needed in the execution phase of the turn. Good tracking ability at high speeds is of great importance, and therefore a high priority for real world robot vehicle applications today. As everyone knows, time and money goes hand in hand. Tracking a path in five minutes is better than to do it in ten, and doing so while not loosing accuracy is even better.

References

[1] Ronald C. Arkin; Behaviour-Based Robotics, The MIT Press, Cambridge, 1998

[2] G W. Lucas; A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators, The Rossum Project, 2001

[3] J S. Wit; Vector Pursuit Path Tracking for Autonomous Ground vehicles, Ph.D thesis, University of Florida, 2000

[4] T. Hellström; Autonomous Navigation for Forest Machines, Department of Computing Science, Umeå University, Umeå Sweden, 2002

[5] R C. Coulter; Implementation of the Pure Pursuit Path Tracking Algorithm, Robotics Institute, Carnegie Mellon University, January, 1992

[6] G. Dudek, M. Jenkin; Computational Principles of Mobile Robotics, Cambridge University Press, New York, 2000

[7] E. Pärt-Enander, P. Isaksson; Användarhandledning för Matlab 4.2, Uppsala Universitet, 1997

[8] K-team S.A; Khepera User Manual version 5.0, Lausanne, 1998

[9] Foley, van Dam; Computer Graphics - Principles and Practice, Secod Edition, USA, 1999

[10] "Unmanned vehicles: University of Florida" at: http://www.me.ufl.edu/~webber/web1/pages/research_areas/vehcile_control.htm

[11] "Control laws: PID based control equations" at: http://abrobotics.tripod.com/ControlLaws/PID_ControlLaws.htm

[12] "msdn:Microsoft" at http://msdn.microsoft.com/library/default.asp?url=/