Design of an Arbitrary Path-Following Controller for a Non-Holonomic Mobile Platform

Deon G. Sabatta
Mobile Intelligent Autonomous Systems,
Council for Scientific and Industrial Research
Pretoria, South Africa
dsabatta@csir.co.za

***Abstract*— Before a robot may be classed as an autonomous system, it must be able to move unaided in an environment. In a known environment or with pre-recorded trajectories, this reduces to the problem of path following. In order to follow an arbitrarily defined path, we are required to calculate the system inputs to the platform in the form of rotational and translational velocities to keep the robot on this path.**

In this paper we derive an analytical controller for a non-holonomic mobile platform capable of following an arbitrarily defined smooth path. To derive an analytical controller, we require an analytical definition of the path which is not always available for arbitrary paths. To overcome this problem we divide the path up into circular and straight segments which are handled independently. The controller takes as input the first two derivatives of the arbitrary path at each point and calculates the desired forward and rotational velocities required for the platform to asymptotically track the path. The controller derived in this paper is implemented on the Seekur platform from Mobile Robots. Results showing the following of a pre-recorded path from differential GPS are discussed.

I. Introduction and Related Work

When controlling a robot along a desired trajectory, we are required to provide control inputs to the platform (usually in the form of forward and rotational velocity) to track the path. Some path planning algorithms [1], [2] directly provide the required forward and rotational velocities while others [3], [4] only provide the path. When only the desired path is provided, we are required to calculate the desired input velocities to track the path. For simple paths that have been defined analytically, this is normally easy to accomplish, however most path planning algorithms only provide paths as a set of points.

In theory, path following could be accomplished by using a set-point controller and constantly moving the reference along the path at a specified rate. When using a set-point controller, the forward velocity of the platform is related to the distance between the platform and the reference point. Therefore, path-following control achieved through moving the reference point along the path results in movement that is not smooth. To overcome this problem, we propose a path-following controller that maintains a fixed forward velocity, as in [5]. In [5], path following controllers are presented for simple straight and circular paths. In this paper, we extend this work to include the ability to follow arbitrarily defined paths by decomposing the path into simple segments.

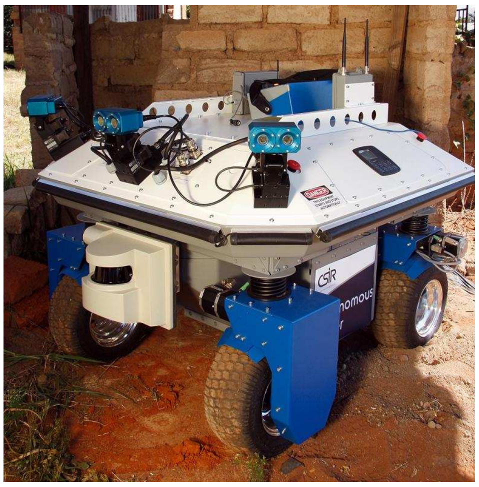


Fig. 1. The Seekur platform from Mobile Robots that the proposed controller was implemented on.

The dynamic control of wheeled platforms poses several problems due to the non-holonomic constraints imposed by the wheels. When holonomic constraints are present in a dynamic system, the total number of generalised coordinates or states may be reduced by the number of constraints. This is in general not true for non-holonomic constraints. The control of such platforms is therefore dependent on the exact choice of output equations. Several authors have examined controllers for the various configurations of wheeled mobile robots. In [6] and [7], controllers are derived for two wheel mobile robots and in [8] a dynamic model for a three wheeled cart is derived. While the system in [8] is shown to be controllable, no dynamic controller is derived.

In the following section we derive a simplified model for our platform and determine the control laws for following straight and circular paths. We then develop the framework for arbitrary path following in Section 3 and present the simulated and experimental results for our control methodology in Section 4. Finally in Section 5 we present our conclusions and future work.

II. Derivation of Control Laws

This section covers the derivation of the controllers as presented in [5] pertaining to our mobile platform. We begin this section by deriving a simplified model of our robotic platform. This model is then used to derive the path following controllers capable of following a few simple analytically defined paths. Thereafter, we derive the framework to permit arbitrary path following.

A. Simplified Model of the Platform

The Seekur platform (Figure 1) has four independently driven and steered wheels. As a result, the platform can move in any given direction and rotate about any given point. The platform has internal controllers controlling each of these wheel's orientations and velocities and allows the user to define the forward, translational and rotational velocities of the platform. Given the rolling constraints imposed by the wheels, certain configurations of motion first require the wheels to change orientation before movement may commence. This delay in motion results in movement of the platform that is not smooth. To overcome this, we consider only continuous curvature paths, i.e. paths that result from a smooth change in forward and rotational velocity. We also ignore any input to translational velocity.

With these assumptions in mind, and given that we can control the forward and rotational velocities independently, we arrive at the following simple kinematic model of the platform.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v\cos\phi \\ v\sin\phi \\ \omega \end{bmatrix}. \tag{1}$$

In this model, (x, y) represents the 2D location of the platform, ϕ its orientation and v and ω represent the forward and rotational velocities respectively.

To design a non-linear controller based on the concept of Input-Output linearisation, we require the system to be in the form,

$$\dot{\boldsymbol{\xi}} = f(\boldsymbol{\xi})\boldsymbol{\xi} + G(\boldsymbol{\xi})\boldsymbol{\Gamma}, \tag{2}$$

where $\boldsymbol{\xi}$ is the system state, $\boldsymbol{\Gamma}$ is the input to the system and $f(\boldsymbol{\xi})$ and $G(\boldsymbol{\xi})$ are the system matrices. To accomplish this we include the forward velocity of the platform into the system state and choose the forward acceleration of the platform as the second input. The final system in the form of equation (2) is,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{bmatrix} = \begin{bmatrix} \xi_4\cos\xi_3 \\ \xi_4\sin\xi_3 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}\boldsymbol{\Gamma}, \tag{3}$$

where $\boldsymbol{\Gamma} = [\omega, a]^{\mathrm{T}}$ represents the input angular velocity and forward acceleration to the system. The state vector $\boldsymbol{\xi}$ represents the system state, as in (1) augmented with the platform velocity.

B. Simple Path Following Controllers

The path following controller that is developed is based on similar ideas to those presented in [5]. Since we have two inputs to the system, we are required to choose two output equations. Since we are performing input-output linearisation. The choice of these output equations directly effects the outcome of the controller design.

To enable smooth path following along a continuous path, we choose as output equations, the shortest distance between some point, in the reference frame of the robot, and the desired path as well as the forward velocity of the robot. This allows us to maintain a constant forward velocity and minimise some error related to the distance from the path. Instinctively, we would like to choose this reference point as the centre of the platform; however, since, under our assumptions, the platform cannot move sidewards, this introduces a non-holonomic constraint at this point and as such the platform becomes uncontrollable. To counter this problem, we proceed as in [5] and choose, as a reference point, a point in front of the robot at a distance l, referred to as the look-ahead distance.

In order to derive controllers using these two output equations, we have to be able to calculate the shortest distance between the look-ahead point and the desired path as a function of state variables and path parameters. To do this we are required to specify the path analytically. We begin by calculating the controllers that keep the robot on a straight path or a circular path.

1) Straight Line Paths: The first controller that will be developed is for that of a straight line with equation,

$$Ax + By + C = 0. \tag{4}$$

We define the look-ahead point in the robot's frame of reference as the point l metres ahead of the centre of the platform. This point can be specified as,

$$p_{la} = (x_{la}, y_{la}) = (\xi_1 + l\cos\xi_3, \xi_2 + l\sin\xi_3). \tag{5}$$

The shortest distance between this point and the path specified in (4) may be shown to be,

$$d = \frac{Ax_{la} + By_{la} + C}{\sqrt{A^2 + B^2}}. \tag{6}$$

For the purposes of deriving this controller, we define the two output equations as,

$$h_1(\boldsymbol{\xi}) = \frac{A(\xi_1 + l\cos\xi_3) + B(\xi_2 + l\sin\xi_3) + C}{\sqrt{A^2 + B^2}}, \tag{7}$$

$$h_2(\boldsymbol{\xi}) = \xi_4. \tag{8}$$

Given that both[1] $L_{g_1}h_1 \neq 0$ and $L_{g_2}h_2 \neq 0$, the relative degrees of both outputs are 1 and the linearised system may be represented in the form,

$$\dot{\boldsymbol{z}} = A\boldsymbol{z} + \boldsymbol{E}\left(\boldsymbol{\alpha}(\boldsymbol{\xi}) + \boldsymbol{\rho}\boldsymbol{\Gamma}\right), \tag{9}$$

[1]Here $L_g h$ is the Lie derivative of h with respect to g defined as $L_g h \equiv \frac{\partial h}{\partial \boldsymbol{\xi}} g$ and $L_g^2 h = L_g(L_g h)$.

where, by definition,

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$
$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}, \qquad \boldsymbol{\rho} = \begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{bmatrix} \quad ,$$

and,

$$\begin{array}{ll} z_1 = h_1 & z_2 = h_2 \\ \alpha_1 = L_f h_1 & \alpha_2 = L_f h_2 = 0 \\ \rho_{11} = L_{g_1} h_1 & \rho_{12} = L_{g_2} h_1 = 0 \\ \rho_{21} = L_{g_1} h_2 = 0 & \rho_{22} = L_{g_2} h_2 \end{array}$$

Through an additional substitution of the form,

$$\boldsymbol{\tau} = \boldsymbol{\alpha} + \boldsymbol{\rho}\boldsymbol{\Gamma}, \tag{10}$$

equation (9) reduces to a true linear equation. (10) may then be rewritten as,

$$\boldsymbol{\Gamma} = \rho^{-1} (\boldsymbol{\tau} - \boldsymbol{\alpha}), \tag{11}$$

to calculate the required inputs to the platform.

With the substitution of (10), we see that the non-linear system transforms into two first-order linear systems. One controlling the minimum distance between the path and the look-ahead point and the second controlling the velocity. Since the platform already incorporates a controller to control the forward velocity, we need only derive a controller for the minimum distance to path. Since ρ_{12} and ρ_{21} are both zero, the controllers for the minimum distance to path and velocities decouple, meaning that we do not have to consider the input τ_2.

Using a simple proportional controller, the input τ_1 may be specified as, $\tau_1 = -k_1 z_1$. Incorporating this into the relation in (11), we arrive at an expression for Γ_1 of,

$$\Gamma_1 = \omega = -\frac{1}{\rho_{11}} (k_1 z_1 + \alpha_1), \tag{12}$$

where,

$$\begin{aligned} \alpha_1 &= \frac{\xi_4 (A \cos \xi_3 + B \sin \xi_3)}{\sqrt{A^2 + B^2}} \quad \text{and} \\ \rho_{11} &= -\frac{l (A \cos \xi_3 - B \sin \xi_3)}{\sqrt{A^2 + B^2}}. \end{aligned}$$

From the form of (12), we may see that the system is only controllable where the inverse of ρ_{11} is defined. That is,

$$A \cos \xi_3 - B \sin \xi_3 \neq 0 \quad \text{and} \quad l \neq 0$$

or equivalently that,

$$\frac{A}{B} \neq \tan \xi_3 \quad \text{and} \quad l \neq 0.$$

These inequalities are satisfied when the desired path is not at right angles to the current heading of the robot and the look-ahead point does not lie at the centre of the platform. The requirement on the heading of the platform arises due to the fact that no direction was specified with the path. The requirement $l \neq 0$, exists because of the non-holonomic constraints enforced through our definition of the system equations as mentioned previously.

2) Circular Paths: The derivation of the controller for the circular path is very similar to the case of the straight line controller. For this case, we derive a controller to enable the platform to follow a circular path specified by,

$$(x - x_c)^2 + (y - y_c)^2 - R^2 = 0, \tag{13}$$

where (x_c, y_c) specifies the centre of the circular path and R the radius.

For this case, the shortest distance between the look-ahead point and the path is calculated as,

$$d = \sqrt{(x_{la} - x_c)^2 + (y_{la} - y_c)^2} - R. \tag{14}$$

To simplify the derivation of the controller, we set $h_1(\boldsymbol{\xi})$ to,

$$h_1(\boldsymbol{\xi}) = (\xi_1 - x_c)^2 + (\xi_2 - y_c)^2 - R^2. \tag{15}$$

Even although this equation is not proportional to (14), it is still a valid error function[2] for the purposes of designing our controller. The second output equation $h_2(\boldsymbol{\xi})$ is left as,

$$h_2(\boldsymbol{\xi}) = \xi_4.$$

Once again, we may determine the relative degree of both outputs to be 1, yielding a linearised equation of the same form as (9). The control input for the rotational velocity may once again be expressed as,

$$\Gamma_1 = \omega = -\frac{1}{\rho_{11}} (k_1 z_1 + \alpha_1); \tag{16}$$

however, the values of α_1 and ρ_{11} are now replaced by,

$$\begin{aligned} \alpha_1 &= 2\xi_4 ((\xi_1 - x_c) \cos \xi_3 + (\xi_2 - y_c) \sin \xi_3 + l) \quad \text{and} \\ \rho_{11} &= 2l ((\xi_2 - y_c) \cos \xi_3 - (\xi_1 - x_c) \sin \xi_3). \end{aligned}$$

Once again we require $\rho_{11} \neq 0$ to ensure that the control input is finite. This is ensured provided that,

$$(\xi_2 - y_c) \cos \xi_3 \neq (\xi_1 - x_c) \sin \xi_3 \quad \text{and} \quad l \neq 0.$$

The first inequality above can be rewritten as,

$$\frac{\xi_2 - y_c}{\xi_1 - x_c} = \frac{\sin \xi_3}{\cos \xi_3} = \tan \xi_3,$$

once again requiring that the heading of the robot is not at right angles to the desired path for the same reason as in the straight path case.

III. Arbitrary Path-Following Framework

In the previous section, we derived two path-following controllers, one for straight paths and the other for circular paths. In this section, we develop a method whereby an arbitrary path may be expressed in terms of straight- and circular-paths for the purposes of controlling the platform.

The input to the arbitrary-path follower takes the form of a collection of points in the robot reference frame together with a recommended path velocity at that point. At each time step, the nearest point in the path to the robot is identified as (x_i, y_i). A local straight or circular path is then calculated

[2] Both functions evaluate to zero at the same locations and have derivatives of the same sign over corresponding intervals.

that passes through this point and matches the local derivative information of the arbitrary path. This path information is then fed to the controller to enable path following.

To calculate the parameters of this path segment, we begin by determining the first two derivatives in x and y with respect to the point index of the points specifying the arbitrary path. These derivatives are calculated using the 3-point centre difference formulae as follows[3],

$$\mathrm{d}x = \frac{1}{2}(-x_{i-1} + x_{i+1}), \tag{17}$$

$$\mathrm{d}^2x = x_{i-1} - 2x_i + x_{i+1}, \tag{18}$$

where x_i represents the x value of the closest point to the robot. x_{i-1} and x_{i+1} denote the neighbouring points of the closest point. The same relations may be used to calculate the derivatives of the y coordinates of the input path. Derivatives with respect to coordinate indices, as opposed to each other, are used to prevent the derivatives from becoming ill conditioned when the path lies on one axis only[4].

Given the derivatives of the path coordinates, we may calculate the curvature of the path to determine whether it is a straight or circular path. From to [9], curvature is defined as,

$$\kappa = \left| \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{\frac{3}{2}}} \right|. \tag{19}$$

This value represents the inverse of the Radius of Curvature (RoC) of the path. The inverse quantity is used as the RoC becomes infinite for a straight path. If this value is near zero, we may attempt to fit a straight path through the given points, while, when this value lies away from 0, we will fit a circular path.

A. Straight Line Segments

To calculate the parameters of a straight line path of the form (4) we proceed as follows. The derivative of the line in (4) may be calculated as,

$$\frac{\mathrm{d}y}{\mathrm{d}x} = -\frac{A}{B}.$$

From this we may choose the values of A and B as,

$$A = \mathrm{d}y \qquad \text{and} \qquad B = -\mathrm{d}x.$$

We can then solve for C by substituting the point of interest (x_i, y_i) into (4) to obtain,

$$C = \mathrm{d}x\ y_i - \mathrm{d}y\ x_i.$$

In summary, given the closest point to the robot and the path derivatives at that point, the coefficients of the straight path defined in (4) may be calculated as,

$$A = \mathrm{d}y, \tag{20}$$

$$B = -\mathrm{d}x, \tag{21}$$

$$C = \mathrm{d}x\ y_i - \mathrm{d}y\ x_i. \tag{22}$$

[3]When at the beginning or end of the path, the forward and backward difference formulae would have to be used.

[4]In this case the derivative $\frac{\mathrm{d}y}{\mathrm{d}x}$ could become infinite.

B. Circular Path Segments

In order to calculate the parameters for a circular path, we need to know in which direction the circle is turning. The relation for curvature in (19) is unsigned, however, if we remove the absolute value we obtain a signed curvature that indicates the direction of rotation. Signed curvature is defined, using the path derivatives, as,

$$\tilde{\kappa} = \frac{\mathrm{d}x\ \mathrm{d}^2y - \mathrm{d}y\ \mathrm{d}^2x}{(\mathrm{d}x^2 + \mathrm{d}y^2)^{\frac{3}{2}}}. \tag{23}$$

The signed curvature is positive when the unit tangent rotates in a counter-clockwise direction and negative when it rotates clockwise.

If we consider the case where $\tilde{\kappa} > 0$, the unit tangent rotates in a counter-clockwise direction meaning that the circular path is turning to the left and hence the centre of the circle lies $90°$ to the left of the tangent vector. The unit tangent vector for the path may be calculated from the path derivatives as,

$$\hat{\boldsymbol{t}} = \frac{\mathrm{d}x\ \hat{\boldsymbol{x}} + \mathrm{d}y\ \hat{\boldsymbol{y}}}{\sqrt{\mathrm{d}x^2 + \mathrm{d}y^2}},$$

where $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ represent the x and y unit vectors respectively. Since the radius vector lies $90°$ to the left of the tangent vector, we apply a $90°$ rotation matrix to $\hat{\boldsymbol{t}}$ to calculate $\hat{\boldsymbol{r}}$ as,

$$\hat{\boldsymbol{r}} = R_{90°}\hat{\boldsymbol{t}} = \frac{-\mathrm{d}y\ \hat{\boldsymbol{x}} + \mathrm{d}x\ \hat{\boldsymbol{y}}}{\sqrt{\mathrm{d}x^2 + \mathrm{d}y^2}}. \tag{24}$$

The centre of the circular path then lies at a distance, $R = \frac{1}{|\tilde{\kappa}|}$, away from (x_i, y_i) along the radius vector $\hat{\boldsymbol{r}}$. The centre point of the circle may then be calculated as,

$$x_c = x_i - R\ \mathrm{d}y \left(\mathrm{d}x^2 + \mathrm{d}y^2\right)^{-\frac{1}{2}},$$

$$y_c = y_i + R\ \mathrm{d}x \left(\mathrm{d}x^2 + \mathrm{d}y^2\right)^{-\frac{1}{2}}.$$

For the case, $\tilde{\kappa} < 0$, the required rotation applied to the unit tangent to obtain the unit radius vector becomes $-90°$. This is equivalent to using the transpose of the rotation matrix in (24). The effect of this is to alter the sign of R in the equations used to calculate the circle centre. These two cases may be combined by including the sign of $\tilde{\kappa}$ in the computation of x_c and y_c or simply by replacing R with $\frac{1}{\tilde{\kappa}}$. After doing this, the resulting parameters for a circular path are found as,

$$R = \left|\frac{1}{\tilde{\kappa}}\right|, \tag{25}$$

$$x_c = x_i - \frac{\mathrm{d}y}{\tilde{\kappa}} \left(\mathrm{d}x^2 + \mathrm{d}y^2\right)^{-\frac{1}{2}}, \tag{26}$$

$$y_c = y_i + \frac{\mathrm{d}x}{\tilde{\kappa}} \left(\mathrm{d}x^2 + \mathrm{d}y^2\right)^{-\frac{1}{2}}. \tag{27}$$

IV. Experimental Results

A. Simulated Results

The simulated results were calculated using a Simulink implementation of the simplified platform kinematics and controller. No platform dynamics were included in the simulation allowing for instantaneous changes in platform velocity.

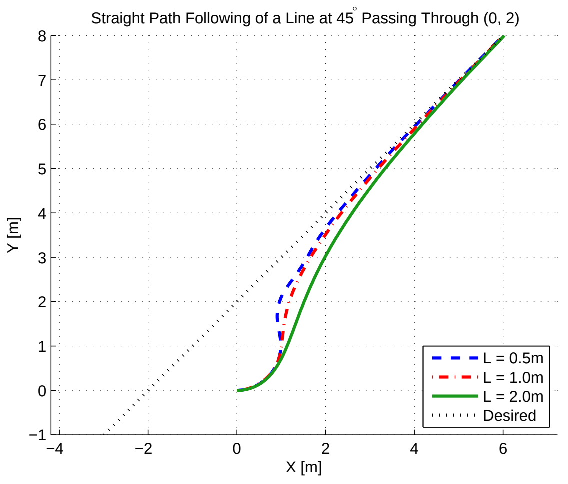


Fig. 2. Plot of the simulated straight line following controller with various look-ahead distances.

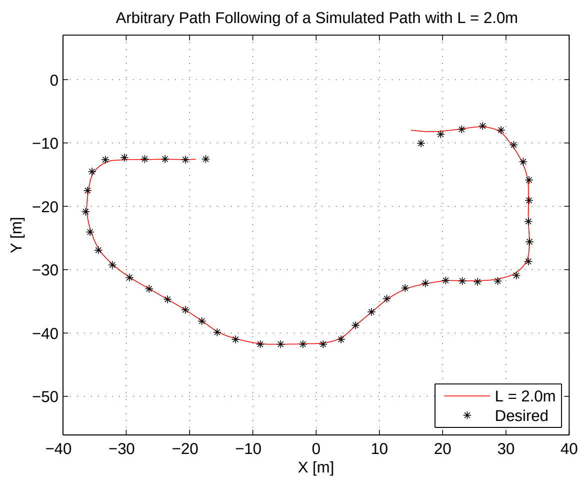


Fig. 4. Plot of the simulated arbitrary path following framework on a simulated path around the parking lot.

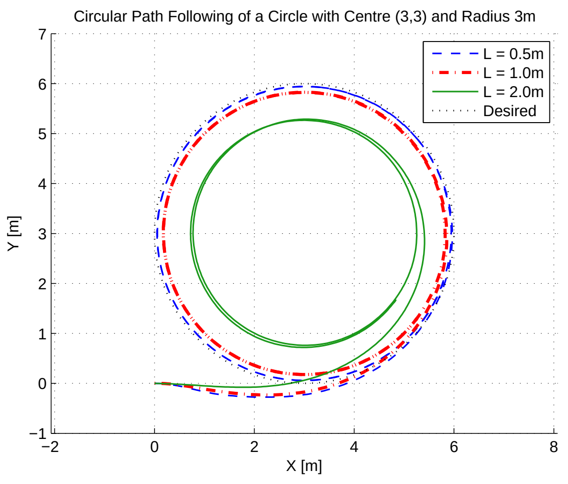


Fig. 3. Plot of the simulated circular path following controller with various look-ahead distances.

1) Simple Path Following: In this section we present the simulated results for the straight and circular path following controllers. For the straight path following controller, the robot was made to follow a straight path at $45°$ passing through the origin. The robots initial state was $[2, 0, 0]^{\mathrm{T}}$. The simulated results are shown for look-ahead distances of 0.5m, 1.0m and 2.0m in Figure 2.

These paths were followed with a forward velocity of $v = 0.5\mathrm{m/s}$ and a proportional gain of $k = 0.3$. From these results we can see that a larger look-ahead distance results in smoother paths but longer settling times.

For the circular controller, we start with the platform at the origin and attempt to navigate a circular path of radius 3m positioned at (3,3). The result of this simulation is shown in Figure 3.

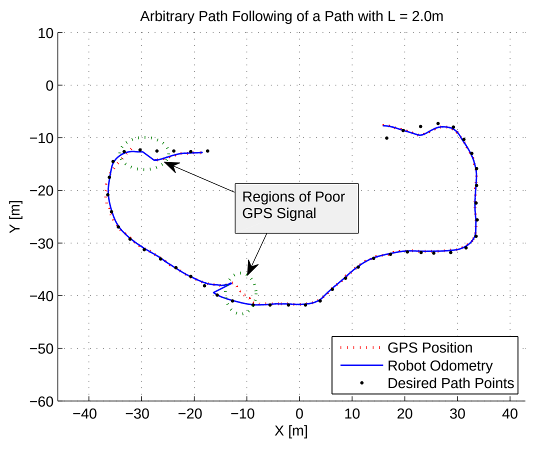
These paths were also followed with a forward velocity of $v = 0.5\mathrm{m/s}$ and a proportional gain of $k = 0.3$. From these results we can see that increasing the look-ahead distance causes the actual path of the platform to diverge from the desired path. In the limit, a look ahead distance of 3.0m for this example would result in the platform remaining in the centre of the circle and rotating on the spot.

The performance of the arbitrary path following controller was tested in simulation by implementing a controller to follow a sample path captured by DGPS in the parking lot behind building 17a on the CSIR campus. The locations indicated in the plot of Figure 4 represent the ENU offset of the path relative to the DGPS base station. The controller was simulated with a look-ahead distance of 2.0m and a starting position slightly off the path to demonstrate the ability of the controller to navigate onto the path. The result is presented in Figure 4.

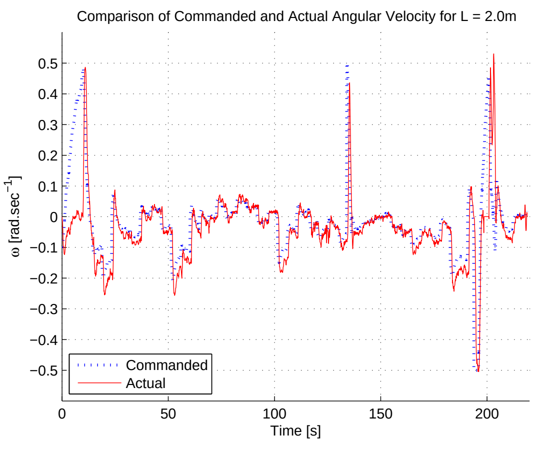
2) Influence of Look-Ahead Distance l on Path Following: From the examples above, we may infer that the trade-off regarding the selection of look-ahead distances is smoothness of path vs. accuracy of path. It is also worth mentioning that the deviation from the path is related to the ratio of look-ahead distance to path radius. Since the angular rotation is limited and the forward velocity is fixed, this also imposes a restriction on the smallest radius of curvature that the robot may follow. With this in mind, it may be pertinent to relate the look-ahead distance and desired forward velocity of the platform to the radius of curvature so that as the radius of curvature decreases, the robot slows down and the look-ahead distance is reduced. This would allow for more accurate path following at lower speeds on tighter circles. However, for paths of meaningful scale, the tracking performance is reasonable and it becomes evident that the impact of the look-ahead distance is diminished.

B. *Experimental Results*

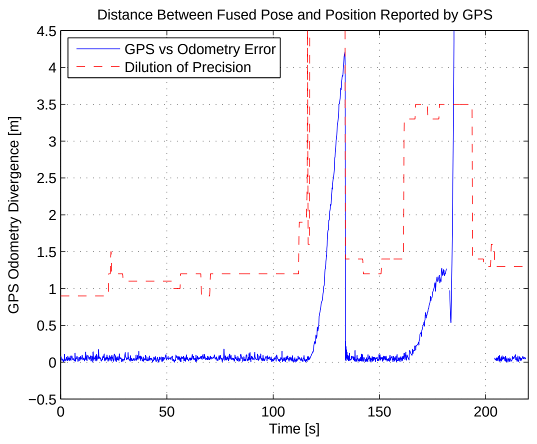
In this section we discuss the experimental results of the arbitrary path following controller. We attempt to follow a path recorded in the parking lot of building 17a. This path was recorded using the DGPS heading unit on the platform, then converted to the ENU coordinate frame and resampled



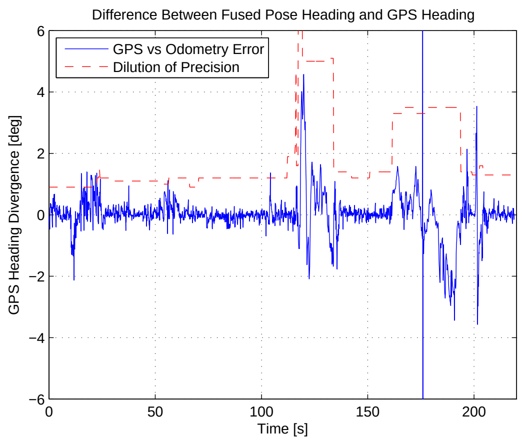
(a) Arbitrary path following with a look-ahead distance $L = 2.0\mathrm{m}$.



(b) Comparison of commanded and actual rotational velocities.



(c) Distance between fused pose location and DGPS location showing GPS DOP.



(d) Distance between fused pose heading and DGPS heading showing GPS DOP.

Fig. 5. Experimental results pertaining to the arbitrary path following controller.

so that the points along the path are not too close. This is necessary to prevent excessive jerkiness of the path during following as the controller continually adjusts the path.

The results of the arbitrary-path following controller are presented in Figure IV-A.2. Figure 5(a) shows the fused robot odometry together with the position returned by the GPS unit. Initially, the path following algorithm performs satisfactorily as the number of visible satellites is large and the signal quality is good. As the robot moves towards the end of the path we see two anomalies. The first at around (-10, -40) where the GPS signal quality was reduced due to the presence of an urban canyon. This anomaly occurred at around 120s as may be seen in Figures 5(c) and 5(d). The DOP at this point rose above 2.0 and the position solution of the GPS was deemed inaccurate. The robot continued to follow the path by extrapolating the last position using internal odometry. When the GPS signal was reacquired, the error in the odometry was recovered and the internal pose of the robot was set to the GPS location. The controller then recovered steering the platform back onto the path. As no absolute ground truth is available, it is difficult to decide which input was more accurate. While it would appear that the platform odometry estimate diverged from the path, this is only because the controller strives to keep this estimated position on the path. The second anomaly present in the GPS signal occurred from around 160s to 195s where the GPS signal was lost for a significantly longer time.

Figure 5(b) shows the differences between the commanded and actual rotational velocities and Figures 5(c) and 5(d) show the errors between the fused estimates and the GPS positions and heading respectively. Figures 5(c) and 5(d) also show the HDOP superimposed on the results. In these figures, loss of signal on the GPS is indicated by missing data in the plots.

V. Conclusions

This paper presents a method whereby complex arbitrary paths may be decomposed into simpler sub-components for which non-linear controllers may be designed. The performance of this method when tracking an arbitrary paths has been shown to be acceptable with most of the error attributed to problems with GPS signal quality. These problems could be mitigated through the use of a statistical fusion framework to improve the results.

One problem with the above approach is the inability to specify direction together with the path information. As a result, the robot will follow the path in whichever direction is closest to its current heading. To overcome this problem, an additional controller would have to be developed to ensure that the platform heading was within 90° of the desired path heading at the start of the path.

REFERENCES

[1] J. Borenstein and Y. Koren. The vector field histogram – fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991.

[2] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4:23–33, 1997.

[3] O. Khatib and S. Quinlan. Elastic bands: Connecting, path planning and control. In *IEEE International Conference on Robotics and Automation*, May 1993.

[4] R. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 2–7, 1989.

[5] N. Sarkar, X. Yun, and V. Kumar. Control of mechanical systems with rolling constraints: Application to dynamic control of mobile robots. *International Journal of Robotics Research*, 13(1):55–69, 1994.

[6] C. Canudas de Wit and R. Roskam. Path following of a 2-dof wheeled mobile robot under path and input torque constraints. In *IEEE International Conference on Robotics and Automation*, pages 1142–1147, 1991.

[7] C. Samson and K. Ait-Abderrahim. Feedback control of a non-holonomic wheeled cart in cartesian space. In *IEEE International Conference on Robotics and Automation*, pages 1136–1141, 1991.

[8] B. d'Andrea Novel, G. Bastin, and G. Campion. Modelling and control of non-holonomic wheeled mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 1130–1135, 1991.

[9] Salas, E. Hille, and G. J. Etgen. *Calculus: One and Several Variables*. Wiley, 8th edition, 1999.