Follow the Past - a Path Tracking Algorithm for Autonomous Forest Vehicles

April, 2004

Thomas Hellström, **thomash@cs.umu.se**

Ola Ringdahl, **ringdahl@cs.umu.se**

UMINF 04.11
ISSN-0348-0542

Umeå University
Department of Computing Science
SE-901 87 Umeå
Sweden

Contents

Abstract

A number of algorithms for path tracking are described in the robotics literature. Traditional algorithms like Pure Pursuit [Cou92] and Follow the Carrot [Bar01] use position information to compute steering commands that make a vehicle approximately follow a predefined path. These algorithms are well known to cut corners since they do not explicitly take into account the actual curvature of the path. In this paper we present a novel algorithm that uses recorded steering commands to overcome this problem. The algorithm is constructed within the behavioural paradigm, and is divided into three separate behaviours, each one responsible for one aspect of the path tracking task. The algorithm is implemented in a simulator for forest machines and the results are compared with the Pure Pursuit and the Follow the Carrot algorithms. The results show a significant improvement in performance, both for ideal noise free position data, and also for position data with added simulated noise.

1 Introduction

The *Follow the Past* algorithm has been developed as part of the project *Autonomous Navigation for Forest Machines* [Hel02] at Umeå University. The goal of this project is to develop a path tracking forest vehicle. A human driver drives the path once while the computer system records position, velocity, orientation and steering angle at every moment. In this way we know exactly how the vehicle moved at a specific point, and can use this information each time the vehicle autonomously travels along the path. Traditional algorithms like *Follow the Carrot* [Bar01] and *Pure Pursuit* [Cou92] use position information only, and sometimes run into problems that can be avoided by taking into account the additional information from the human driver. If the vehicle gets off course, for example when avoiding an obstacle or because of noise in the positioning sensors, the *Follow the Past* algorithm steers like the driver, plus an additional angle based on the distance to the path. This report describes the derivation of the algorithm and presents simulated runs and comparisons with traditional path tracking algorithms. All simulations are run on a dedicated simulator developed by Ringdahl [Rin03].

2 Follow the Past algorithm

During the manual driving of the path, the orientation and steering angle are stored together with the position at every moment. All this information is used by the *Follow the Past* method, which is composed of three independent behaviours:

ϕ_β:	Turn towards the recorded orientation.
ϕ_γ:	Mimic the recorded steering angle.
ϕ_α:	Move towards the path.

Each behaviour suggest a steering angle and is reactive, i.e operates on the current input values; orientation, steering angle and shortest distance to the path. ϕ_α uses recorded closest positions (x', y') and actual position (x, y) as inputs. ϕ_β uses recorded orientation θ' and actual orientation θ as inputs. ϕ_γ uses the recorded steering angle ϕ' as inputs. The three behaviours are fused into one action, the commanded steering angle ϕ_t, as shown in Figure 1. The figure also shows an obstacle avoidance system and how it is integrated with the path tracking.

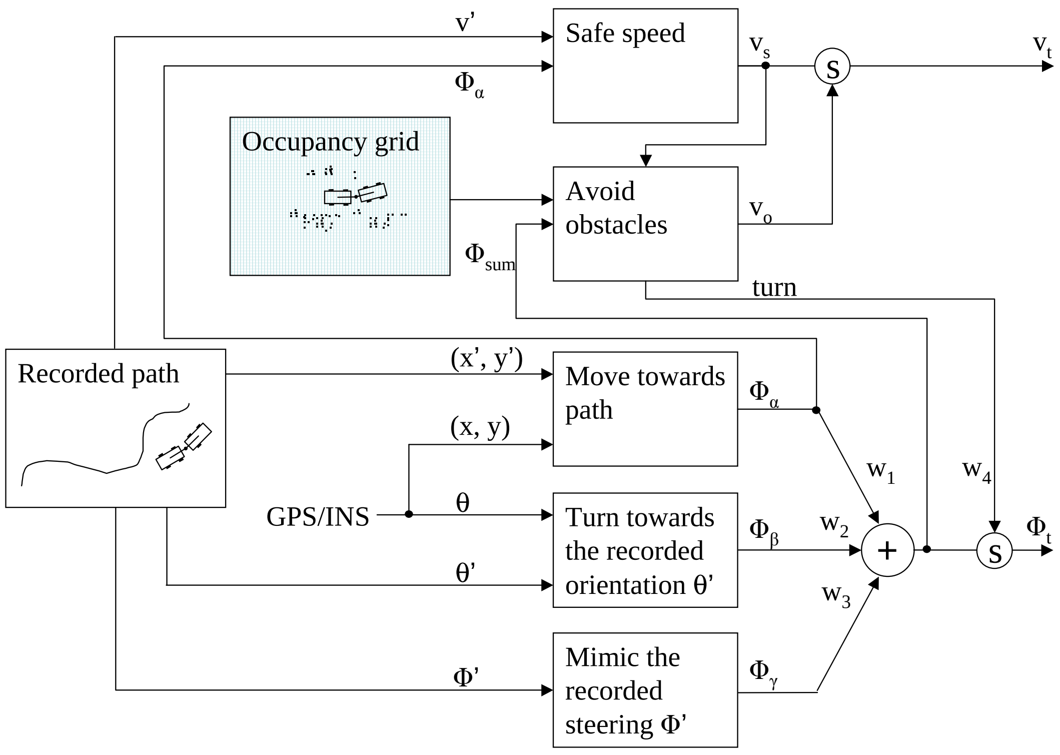


Figure 1: Path tracking with reactive control of steering angle ϕ_t and speed v_t

2.1 Definitions

Since the forest vehicle uses articulated steering, the definition of heading, orientation and steering angle may not be obvious to the reader. In this paper we use the following definitions:

The steering angle ϕ is defined as the angle between the front and rear sections of the vehicle, as shown in the left part of Figure 2. It is convenient to identify half of this angle, i.e. the angle between the baseline and each vehicle section, as shown in the right part of Figure 2.

Heading η is defined as the direction of the front part of the vehicle, defined in the global coordinate system as shown in Figure 3.

The orientation θ of the vehicle, defined in the global coordinate system as shown in Figure 3, is the direction in which the vehicle would travel if the steering angle was zero and the baseline was maintained, as the dashed vehicle in Figure 3. It can be computed as the heading of the vehicle minus half the steering angle, i.e. $\theta = \eta - \frac{\phi}{2}$.

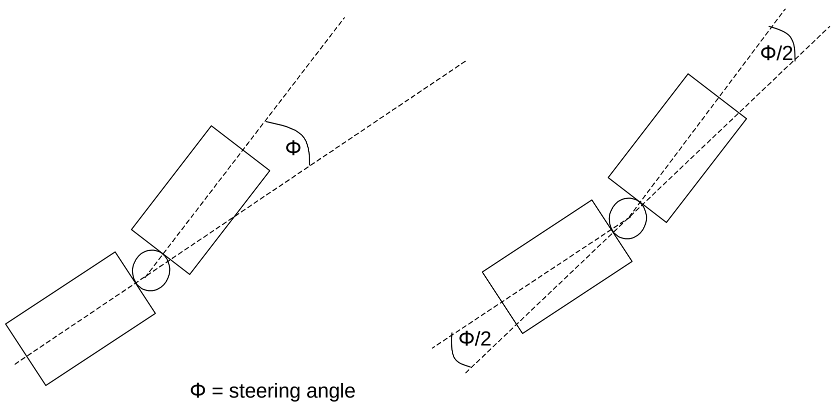


Figure 2: Definition of the steering angle ϕ for a forest machine with articulated steering.

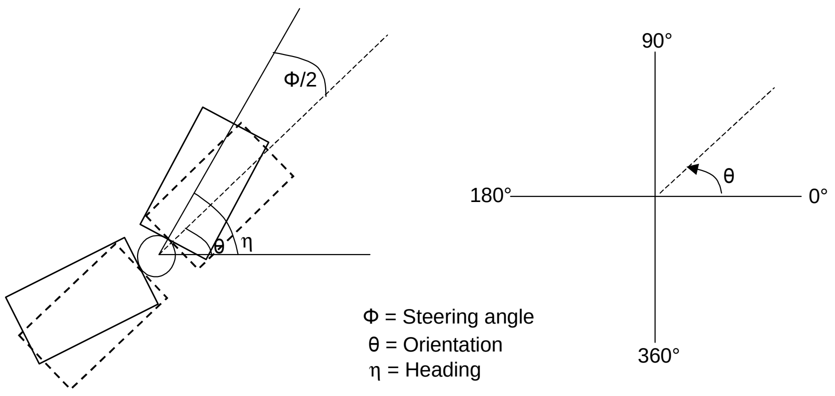


Figure 3: Definition of the vehicle's orientation θ, and heading η.

2.2 ϕ_β : Turn towards the recorded orientation

The angle θ' is defined as the recorded orientation at the closest point on the recorded path. This point is called the *path point*. ϕ_β is computed as the difference between the current orientation θ and the recorded orientation θ' (refer to Figure 4):

$$\phi_\beta = \theta' - \theta. \tag{2.1}$$

Since the behaviour ϕ_β only is concerned with the error in orientation, we may imagine the vehicle following a virtual path parallel to the recorded path and passing through the steering joint of the vehicle as shown in Figure 4.

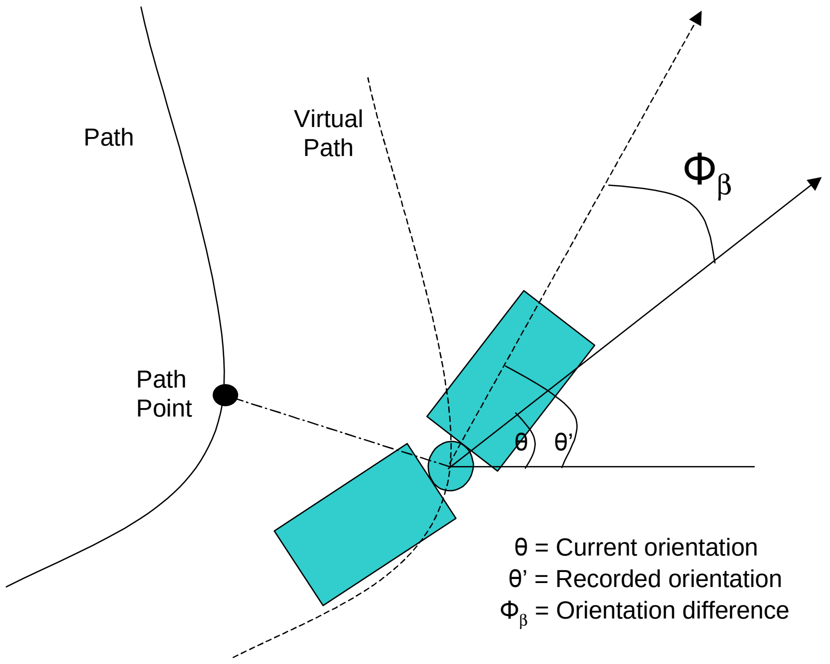


Figure 4: The vehicle follows a virtual path.

2.3 ϕ_γ: Mimic the recorded steering angle

This behaviour simply returns the recorded steering angle ϕ' at the the closest point on the recorded path (refer to Figure 5).

$$\phi_\gamma = \phi' \tag{2.2}$$

By using the recorded steering angle, the curvature of the path is automatically included in the final steering command. This is a great advantage compared to methods like *Pure pursuit* [Cou92] and *Follow the Carrot* [Bar01].

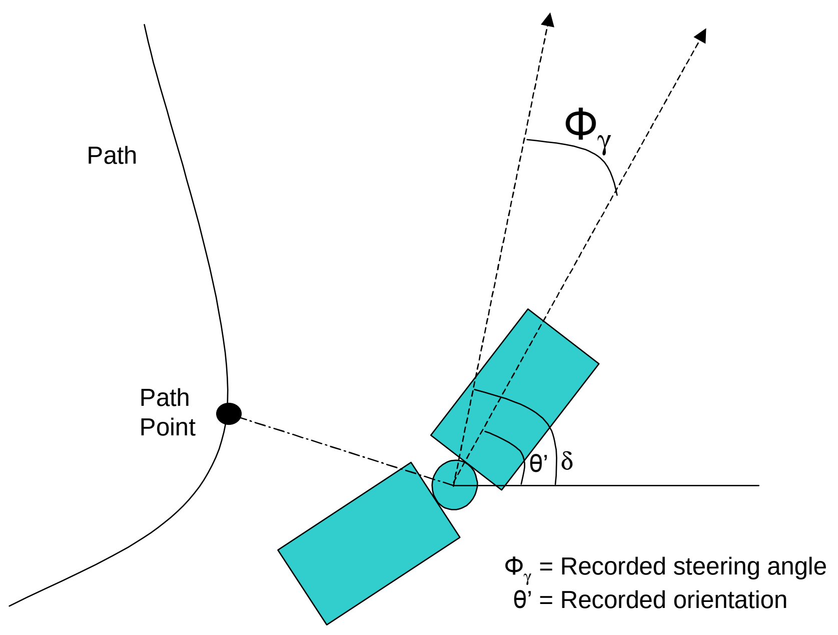

Path
Path Point
Φ_γ
θ'
δ
Φ_γ = Recorded steering angle
θ' = Recorded orientation

Figure 5: Calculation of ϕ_γ.

2.4 ϕ_α: Move towards the path

This behaviour is responsible for bringing the vehicle back to the path if the vehicle for some reason gets outside the path. This can be caused by noise in the position signals, or by the obstacle avoidance system, which is a separate behaviour not described in this report.

ϕ_α can be implemented in many ways. We have implemented two different methods as described below.

2.4.1 Method one: directly proportional to the distance to the path

The simplest way is to calculate a value, directly proportional to the distance between the vehicle and the closest point on the recorded path (refer to Figure 6). This is done by multiplying the distance to the path with a constant:

$$\phi_\alpha = k * d \tag{2.3}$$

with the condition:

$$|\phi_\alpha| \leq 90^\circ$$

where

k:	a constant $[deg/m]$
d:	the signed perpendicular distance between the vehicle and the path

Note that d is negative or positive depending on which side of the path the vehicle is. A typical value for k in the current application is 0.07. To ensure that ϕ_α doses not become too large when the distance to the path is large, we need a constraint. The two behaviours ϕ_β and ϕ_γ see to that the vehicle is parallel to the path, and therefore the angle ϕ_α must be within $[-90°, 90°]$, i.e $|\phi_\alpha| \leq 90°$.

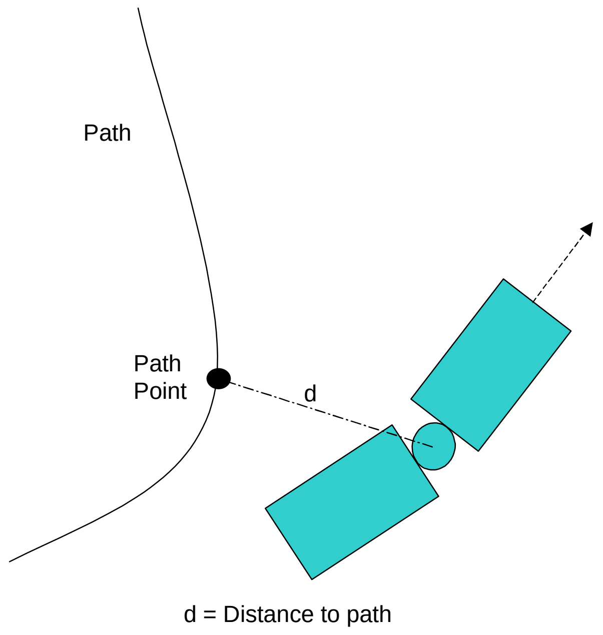


Figure 6: Calculation of ϕ_α in Method one.

2.4.2 Method two: use a Look Ahead Point

With Method one, the calculation of the angle ϕ_α is linear. This may result in oscillations about the path when the vehicle is close to it. Reducing the value for k will result in a slow response for large d, i.e. the vehicle will take longer time to reach the path. Method two overcomes this problem by using a *Look Ahead Point* and calculate the angle between the vehicle and the Look Ahead Point, see Figure 7. The algorithm for computing ϕ_α is:

1. Determine the closest point on the recorded path (i.e. the path point).
2. Compute a *Look Ahead Point* at a *Look Ahead Distance* ℓ from the path point, in the direction δ in Figure 7. This angle is the sum of the recorded orientation θ' and steering angle ϕ' at the path point.
3. Calculate a *Look Ahead Angle* ψ, defined as the angle of the vector between the vehicle's current coordinates and the Look Ahead Point, in global coordinates.

4. Compute ϕ_α as the difference between the Look Ahead Angle and the angle δ.

With the notation in Figure 7 we get:

$$\phi_\alpha = \psi - \delta \tag{2.4}$$

$$where$$

$$\delta = \phi' + \theta' = \phi_\beta + \phi' + \theta \tag{2.5}$$

where

ψ:	is the Look Ahead Angle
θ':	is the recorded orientation at path point p
θ:	is the current orientation
ϕ':	is the recorded steering angle at path point p

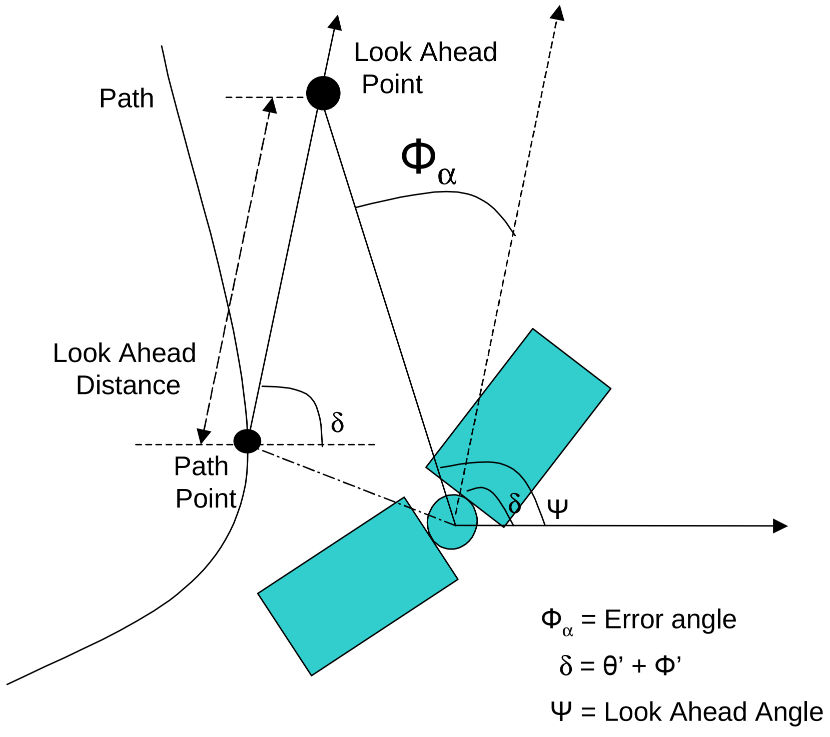


Figure 7: Calculation of ϕ_α in Method two.

2.5 Command fusion

The three behaviours ϕ_α, ϕ_β and ϕ_γ all return a suggested steering angle. These three values are fused into one value ϕ_t by simple addition as shown in Figure 1. I.e.:

$$\phi_t = \phi_\beta + \phi_\gamma + \phi_\alpha. \tag{2.6}$$

For Method One above, this expression will be:

$$\phi_t = \theta' - \theta + \phi' + k * d \tag{2.7}$$

where

θ'	is the recorded orientation at the path point p
θ	is the current orientation
ϕ'	is the recorded steering angle at the path point p
k	is a constant
d	is the shortest signed distance between the vehicle and the path point p

The path point p is the closest point from the steering joint of the vehicle to the recorded path, as illustrated in Figure 7.

For Method Two above, the expression for the fused ϕ_t will be (refer to Figure 8 and Equations 2.2, 2.4 and 2.5):

$$\begin{aligned}\phi_t &= \phi_\alpha + \phi_\beta + \phi_\gamma = \\ &= \psi - \delta + \phi_\beta + \phi_\gamma = \\ &= \psi - (\phi_\beta + \phi' + \theta) + \phi_\beta + \phi' = \\ &= \psi - \theta\end{aligned} \tag{2.8}$$

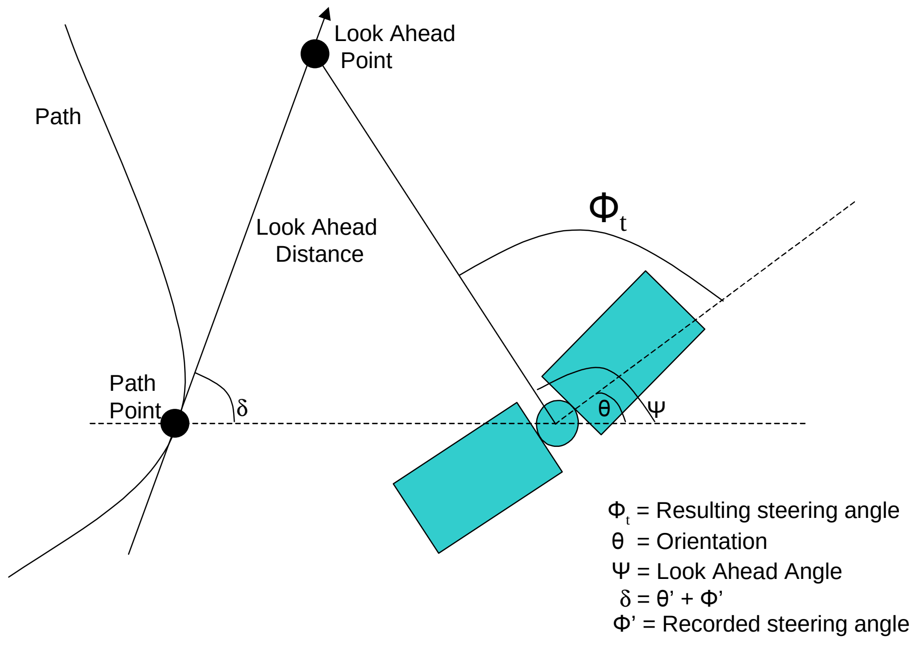
where

ψ	is the Look Ahead Angle
δ	is the angle from Equation 2.5
θ'	is the recorded orientation at the path point p
θ	is the current orientation
ϕ'	is the recorded steering angle at the path point p

2.6 Finding the nearest point on the path

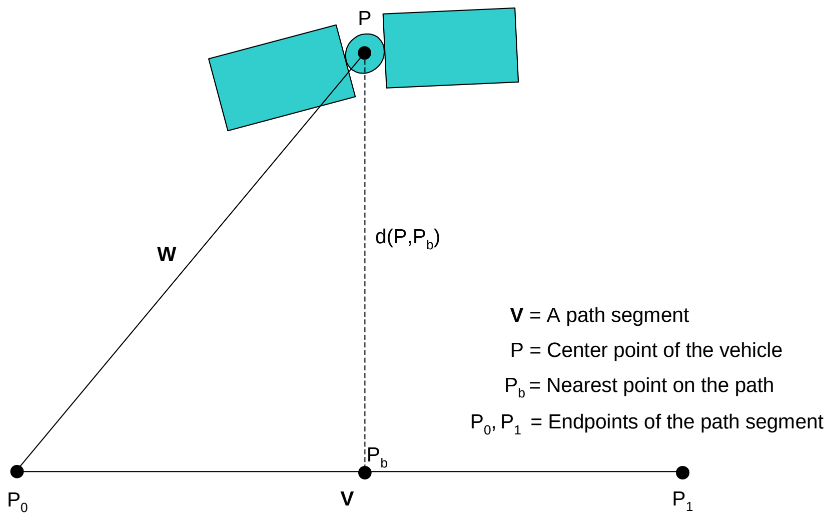
In order to calculate the location of the Look Ahead Point, the nearest point on the path (denoted the path point) must first be determined, see Figure 9. To calculate this, the distance between a point and a linear segment has to be calculated. An algorithm from GeometryAlgorithms.com [sof03] is used for this:

```
distance( Point $P$, Segment $P_0 : P_1$ )
{
    $\mathbf{v} = P_1 - P_0$
    $\mathbf{w} = P - P_0$
    if $((c_1 = \mathbf{w} \bullet \mathbf{v}) <= 0)$  //the point is to the left of P0
        return $d(P, P_0)$
```



Look Ahead Point
Path
Look Ahead Distance
Φ_t
Path Point
δ
θ
Ψ
Φ_t = Resulting steering angle
θ = Orientation
Ψ = Look Ahead Angle
$\delta = \theta' + \Phi'$
Φ' = Recorded steering angle

Figure 8: Command fusion for Method two



P
W
$d(P,P_b)$
V = A path segment
P = Center point of the vehicle
P_b = Nearest point on the path
P_0, P_1 = Endpoints of the path segment
P_b
P_0
V
P_1

Figure 9: Distance between the vehicle's center point P and the closest point on the path, P_b.

```
    if (($c_2 = \mathbf{v} \bullet \mathbf{v}$) <= $c_1$)  //the point is to the right of P1
        return $d(P, P_1)$
    $b = c_1/c_2$
    $P_b = P_0 + b\mathbf{v}$
    return $d(P, P_b)$  //the point is on the segment
}
```

where

P	is the center point of the vehicle
P_0, P_1	is the endpoints of a path segment
P_b	is the point on the path closest to the vehicle
$d(P_1, P_2)$	is the distance between two points

To decide which segment of the path the vehicle is closest to, loop through all segments and select the segment closest to the vehicle. In order to speed up the algorithm, the search begins a few segments behind the last selected segment and continues to a few segments in front of it. In this way "search window" is created around the previous path point. The probability that the closest point on the path will be near the closest point a moment ago, is very high. Normally the new path point should be just in front of the old one if the vehicle moves forward along the path. If the position given by the GPS contains a lot of noise, this window has to be larger since the previous path point may be incorrect. But even in this case, the vehicle should still be close to that position and thereby the nearest point on the path will be close to the correct one.

Since a segment is finite, the vehicle may be outside the perpendicular lines from the endpoints of the segment, as illustrated in Figure 10. In this case, the closest point is one of the endpoints and we must determine which one. One way to do this, however not a very efficient one, is to compute both distances and use the shortest. We must, however, still determine that the vehicle is actually outside the segment. To combine these two computations, we consider the angles between the segment $\overrightarrow{P_0P_1}$ and the vectors $\overrightarrow{P_0P}$ and $\overrightarrow{P_1P}$, i.e. from the segment endpoints to the vehicle's center point, see Figure 9. If either of these angles are 90°, the closest point on the path, P_b, is the corresponding endpoint. If the angle is not 90°, the point lies on either side of the endpoint, depending on whether the angle is acute ($< 90°$) or obtuse ($> 90°$). These conditions are easily tested by computing the dot product of the vectors involved. The sign of this product determines if the distance should be computed to one of the points P_0 or P_1, or as the perpendicular distance to the segment itself. In a situation such as in Figure 10, the algorithm will not find any segment that the vehicle is closest to, and then we have to use the closest endpoint instead.

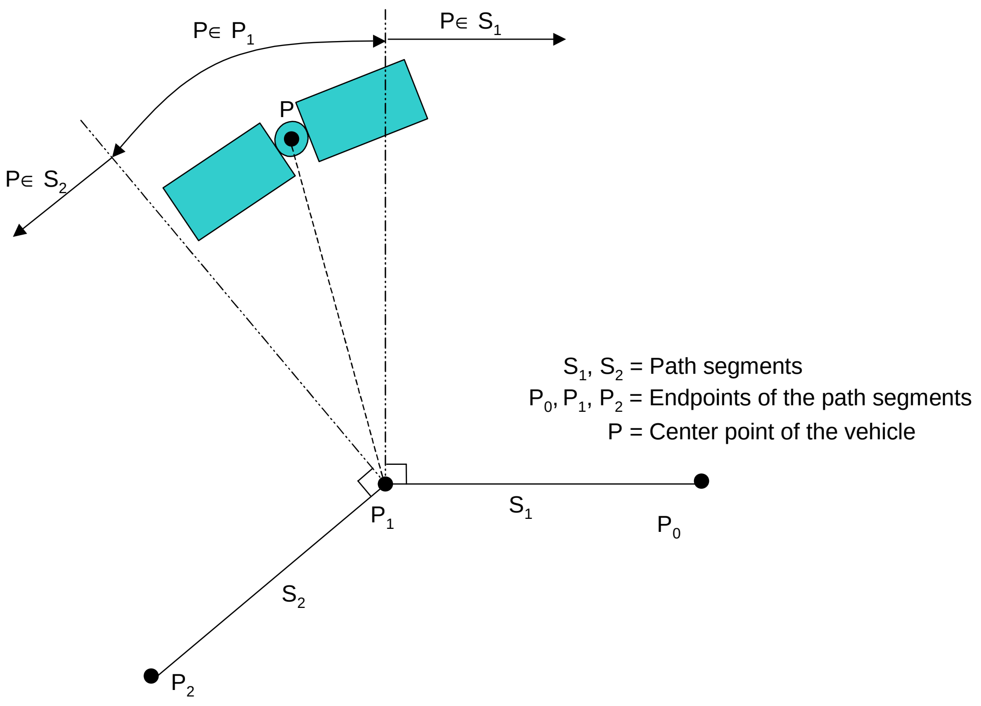

P∈ P1
P∈ S1
P
P∈ S2
S1, S2 = Path segments
P0, P1, P2 = Endpoints of the path segments
P = Center point of the vehicle
P1
S1
P0
S2
P2

Figure 10: Determination of closest point when the vehicle is between two segments.

3 Testing procedure

The developed Follow the Past algorithm is implemented in a simulator, and will be compared to the already existing implementations of the Follow the Carrot [Bar01] and Pure Pursuit [Cou92] methods. The vehicle's path tracking is presented graphically and with numerical error measures consisting of the vehicle's mean and maximum deviation from the path.

Where nothing else is stated, a Look Ahead Distance ℓ of 12 meters and k-value 0.07 are used. These values are determined experimentally and effects the vehicle's ability to quickly return to the path, for example when avoiding obstacles.

4 Results

As shown in Figure 11, the vehicle is able to follow a path perfectly with the *Follow the Past* method with no deviation from the path. In this test, the vehicle starts on the path and no noise in the positioning sensors is assumed. As a reference, Figures 12 and 13 show how the vehicle behaves when using the *Follow the Carrot* and *Pure Pursuit* methods respectively under the same conditions. For this path, the vehicle has to use maximum steering angle in order to follow it. This makes it difficult for both *Follow the Carrot* and *Pure Pursuit* to follow, as they tend to "cut corners" instead of following the path along a very curved path. Under ideal conditions, this shortcoming is solved by using the Follow the Past algorithm.

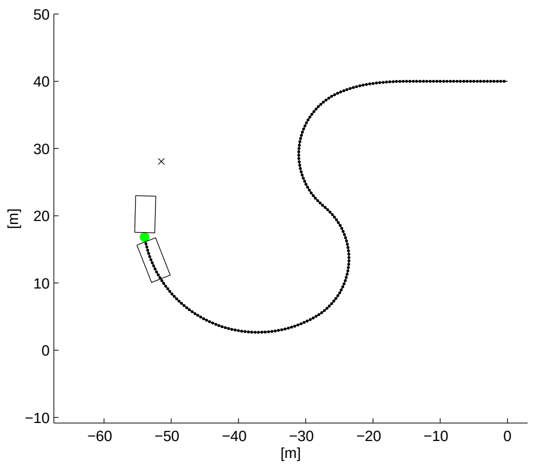


Figure 11: The vehicle follows the path perfectly with the Follow The Past method

4.1 Comparison between the two different methods to compute ϕ_α

In Sections 2.4.1 and 2.4.2, two methods to compute the behaviour ϕ_α, responsible for moving the vehicle towards the path, were introduced. Both methods are based on their response to the distance between the vehicle and the nearest point on the path. As shown in Figure 14, the difference between the two ϕ_α methods is not very large. With the current settings ($k = 0.07, \ell = 12m$), Method one turns slightly sharper towards the path while Method two has a more soft approach, but the difference can be eliminated with different settings for the k-value and the Look Ahead Distance ℓ. The effect of applying a smaller k or a larger ℓ respectively, is that the vehicle will turn slower towards the path, as can be see in Figure 15. As will be shown in the next section, $k = 0.03$ and $\psi = 30$ gives better performance with a noisy position signal, but they are clearly not optimal for a quick return to the path.

4.1.1 Noise

In real life, position information is seldom perfect. A GPS signal is, for example, subject to both systematic and purely random disturbances. Therefore, the algorithm's behaviour when the position information is not completely reliable has to be tested. In order to do this, the performance of the vehicle is observed when a

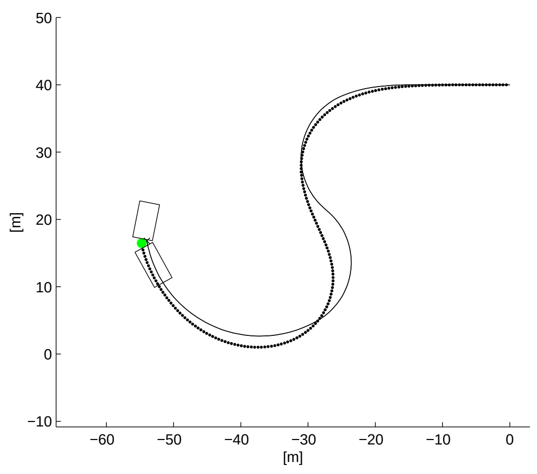


Figure 12: With the Follow the Carrot Method, the vehicle diverges at most 3.2 meters from the path. The average distance error is 0.99 meter.

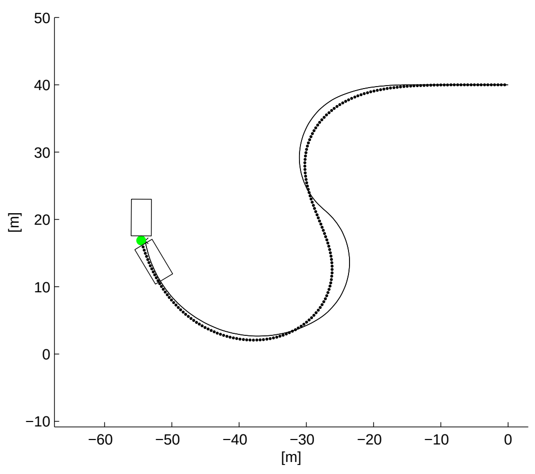
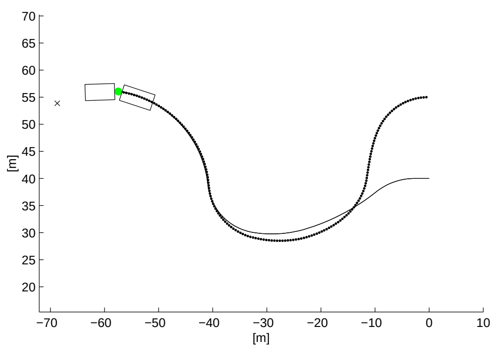
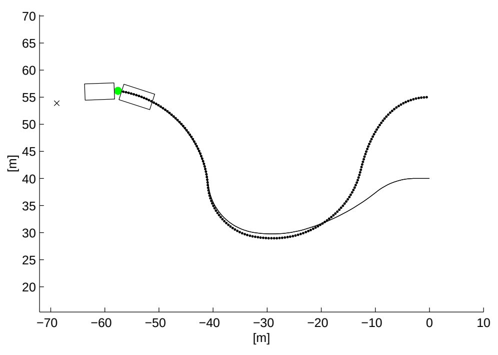


Figure 13: With the Pure Pursuit method the vehicle diverges at most 2.7 meters from the path. The average distance error is 0.76 meter.



(a) Method one, $k = 0.07$



(b) Method two, $\ell = 12m$

Figure 14: Method one turns slightly sharper than Method two towards the path.

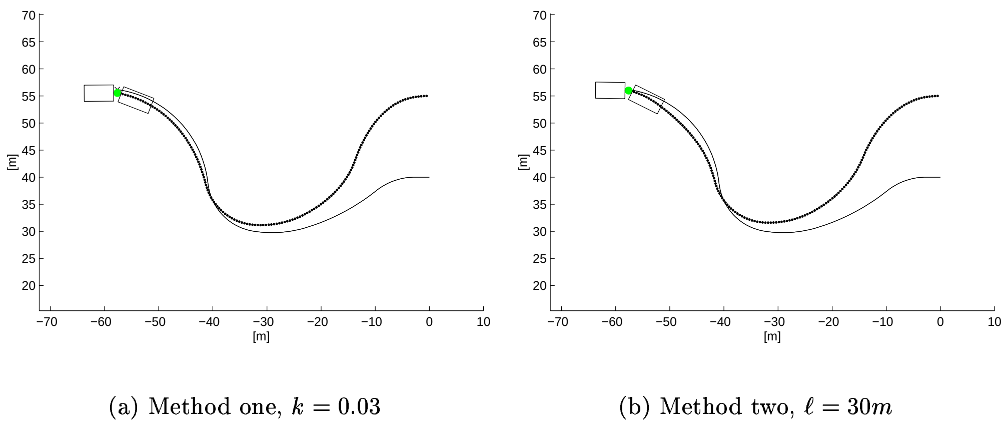


Figure 15: The vehicle turns slower towards the path with smaller k-value or larger Look Ahead Distance ℓ respectively.

normally distributed noise $\varepsilon \in N(\mu, \sigma)$ is added to the position data. Furthermore, since the noise from a GPS does not follow a normal distribution with zero mean value, a sine function is added to the mean value of the noise, thus simulating a periodic drift in the GPS signal. This means that the position (x, y) will contain a noise component with an amplitude of at most σ and a standard deviation σ. For reasons of briefness, noise components will in this report be described by specifying the σ only. The following equations are used:

$$\mu(t) = sin(\frac{2\pi t}{T}) * \sigma \tag{4.1}$$

$$x'(t) = x(t) + \varepsilon_x(t) \tag{4.2}$$

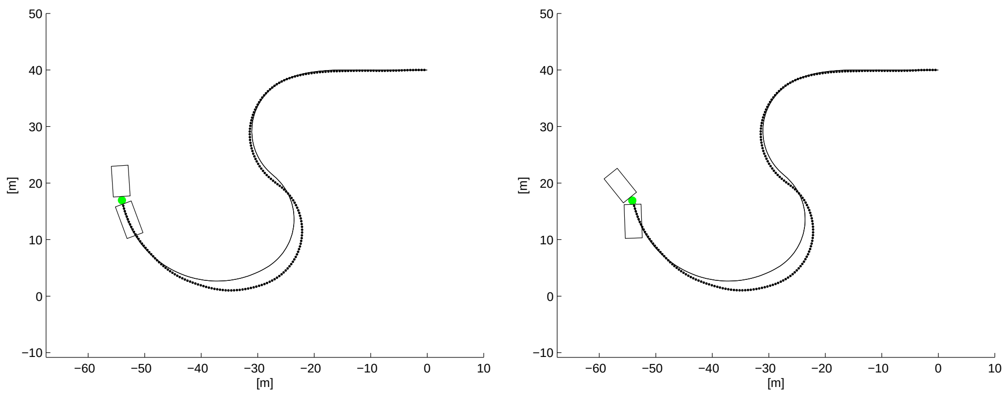
$$y'(t) = y(t) + \varepsilon_y(t) \tag{4.3}$$

where

$\varepsilon_x(t), \varepsilon_y(t) \in N(\mu, \sigma)$

$\mu(t)$:	the mean value of the normally distributed noise at time t
t:	time since start
T:	the periodicity of the noise (value used: 20s)
σ:	the standard deviation of the noise
$x'(t), y'(t)$:	the believed position of the vehicle at time t
$x(t), y(t)$:	the actual position of the vehicle at time t

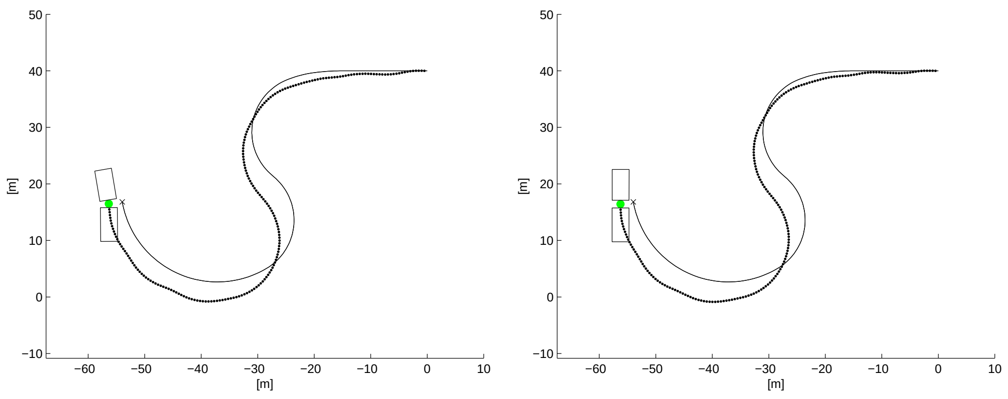
As can be seen in Figure 16, when adding noise with standard deviation $\sigma = 1m$, the *Follow the Past* method still performs better than both the Follow the Carrot and Pure Pursuit methods do without noise. For comparison, the performance of Follow the Carrot and Pure Pursuit without any noise is shown in Figures 12 and 13. There is no significant difference between the two ϕ_α versions of the *Follow the Past* method at this noise level.



(a) Method one (b) Method two

Figure 16: Both methods works well with small noise. The noise standard deviation σ is here one meter. The maximum deviation from the path is about 2.5 meters.

When noise with standard deviation $\sigma = 5m$ is added, the vehicle deviates at most just over four meters from the path, see Figure 17, which is just slightly worse than the Follow the Carrot method working without noise. The performance of the two ϕ_α methods, though poor, still do not differ much from each other.



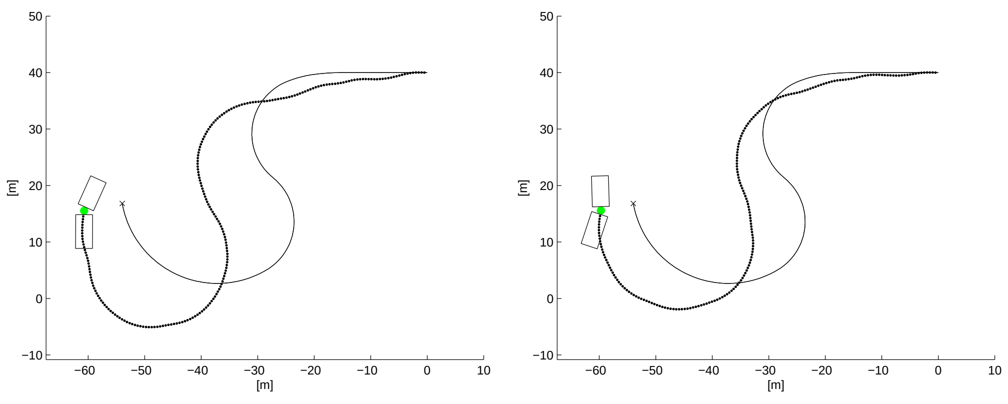
(a) Method one (b) Method two

Figure 17: With standard deviation $\sigma = 5m$, the performance degrades but is similar for the two methods. The maximum deviation from the path is just above four meters.

With standard deviation $\sigma = 9m$, Method one starts to perform notably poorer than Method two, see Figure 18, which would suggest that it is not as tolerant to noise as Method two. This is confirmed when adding a noise with standard deviation

$\sigma = 35m$. Method one brings the vehicle completely off course, see Figure 19. The cause of this error is discussed further in Section 4.2.1. Figure 20 shows that Method two has no problems when dealing with high noise levels.

In a forest environment with lots of trees and other obstacles very close to the vehicle, a GPS delivering a position with an error of several meters can not be tolerated. The situation above is a therefore bit extreme, the maximum position error during simulation is about 138m. Still, the path tracking algorithm should be able to handle this kind of noise without becoming unstable.



(a) Method one (b) Method two

Figure 18: With standard deviation $\sigma = 9m$, Method one performs worse than Method two. The max and mean distance to path for Method one in this example are 13 and 6.3 meters. For Method two, the values are 9.6 and 4.2 meters respectively.

The performance of the two methods is highly dependent of the value of the constant k and Look Ahead Distance ℓ respectively. In the results presented above, the values 0.07 and 12m are used. Different values for k and ℓ at different noise levels have been tested, to see how the algorithms react. Figure 21 shows the behaviour of the two algorithms with different settings for the k-value and Look Ahead Distance respectively, and at different noise levels. Table 1 shows the lowest values of the max and mean distance to the path at different noise levels. The tests were done on the path in Figure 23(a). Note that the difference in performance between the two methods are not very large when tuned correctly. Another interesting observation is that the higher the noise, the smaller value for k gives the best performance. This gives rise to a conflict between behaviours. Normally, when a vehicle gets off course, for example when avoiding an obstacle, we want to get back to the path as fast as possible. This is done by applying a large k-value. But if we have a large k-value, we get problems when the position is noisy. The problem illustrated in Figure 19, when the vehicle gets completely off course, can occur even at lower

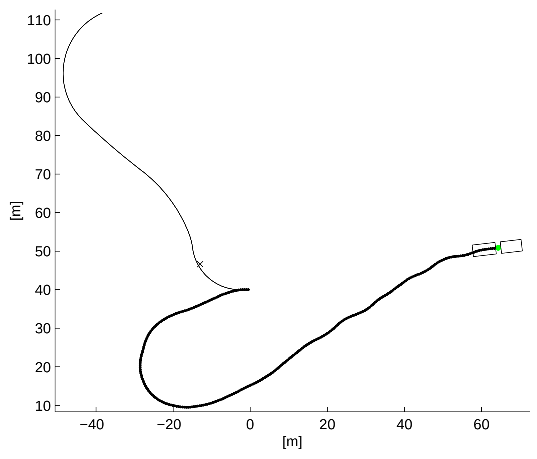


Figure 19: Method one does not work well with large noise. Noise with standard deviation $\sigma = 35m$ is applied in this example.

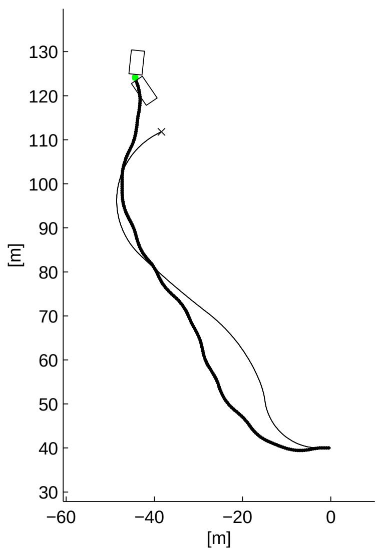
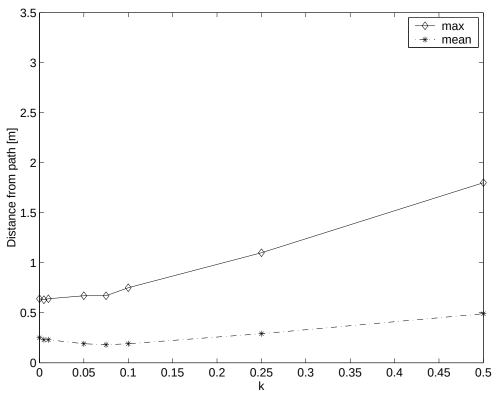


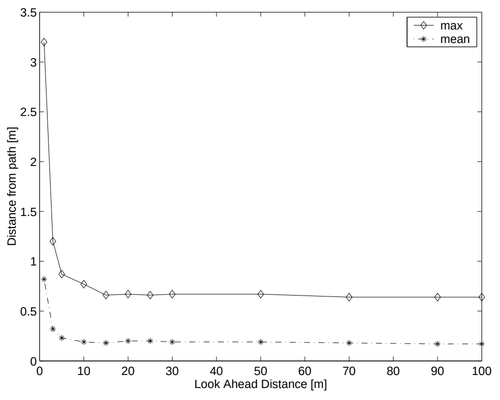
Figure 20: Method two does not have the same difficulties handling noisy position signals as Method one. The standard deviation is the same as in the left figure, 35m.

noise levels when applying high k-values. This means that there is a problem with tuning Method one for optimal performance, with regard to both noise sensitivity and ability to get back to the path quickly after avoiding an obstacle. For Method two, a corresponding problem exists. If we increase the Look Ahead Distance ℓ, the algorithm is less sensitive to noise. But as we can see in Figure 15 the vehicle turns very slowly back towards the path for example when avoiding obstacles. However, as opposed to Method one, Method two does not become unstable when applying a high (or low) Look Ahead Distance.

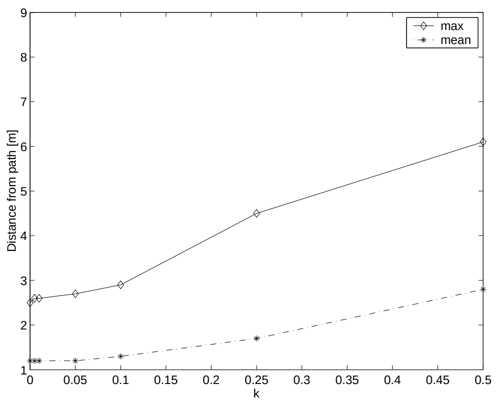
Is the performance for the two algorithms similar for different paths? To answer this question, the algorithms are applied on three different paths (shown in Figure 23). The result, shown in Figure 22, indicates that we should be able to find values for k and ℓ that work well on a variety of paths. Table 2 shows the settings for which the two methods performed best with the different paths. Even if the optimal settings may differ between paths, the general conclusion is that with larger noise, a smaller k-value and a longer Look Ahead Distance gives the best performance. Note that the ability to quickly return to the path is not regarded in this test.



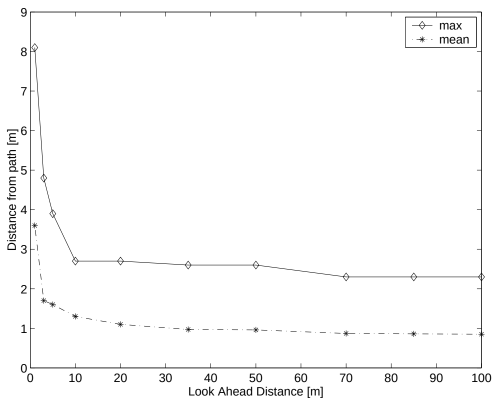
(a) Method one with $\sigma = 0.3m$.



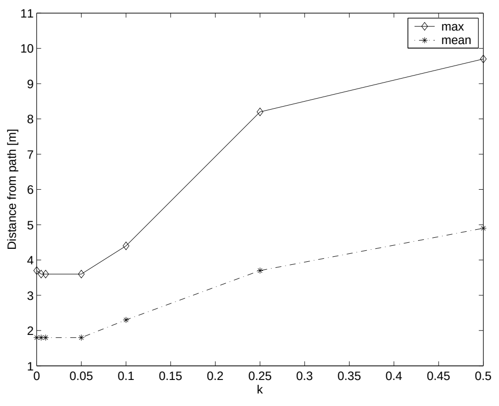
(b) Method two with $\sigma = 0.3m$.



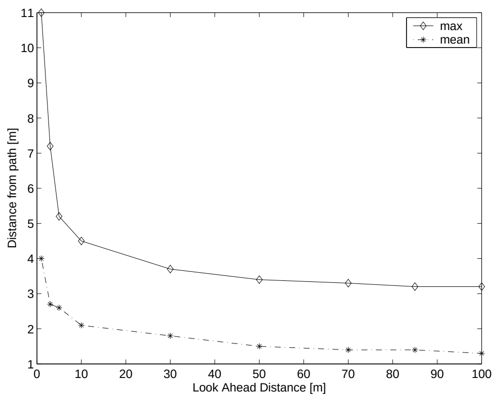
(c) Method one with $\sigma = 3m$.



(d) Method two with $\sigma = 3m$.

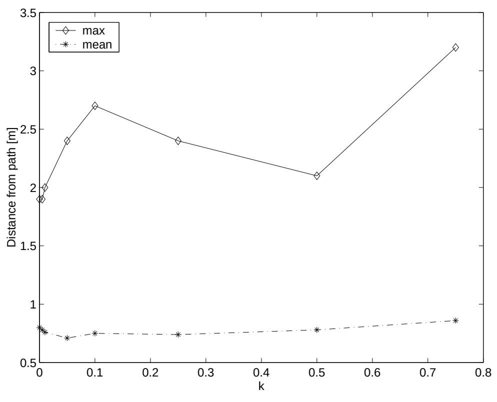


(e) Method one with $\sigma = 5m$.

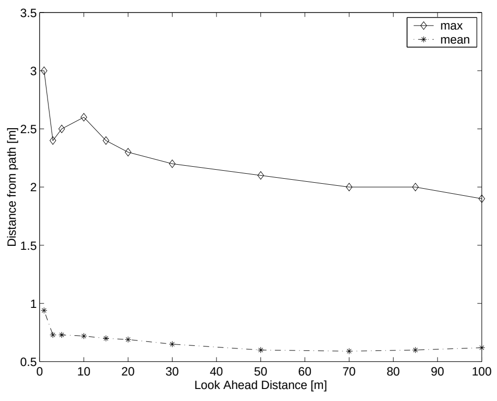


(f) Method two with $\sigma = 5m$.

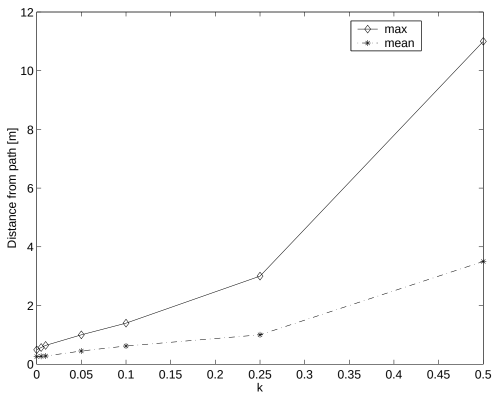
Figure 21: Max and mean distance from the path with different settings for the k-value and Look Ahead Distance, ℓ, and with different noise settings. The path shown in Figure 23(a) was used in this test.



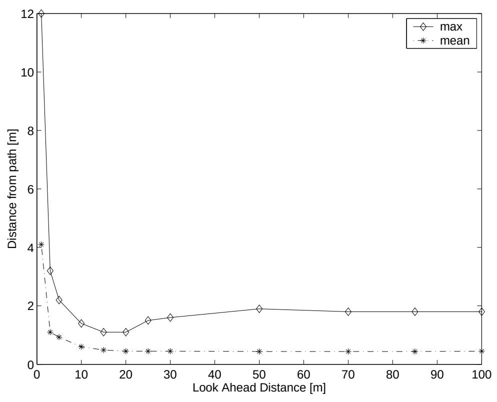
(a) Method one for path one



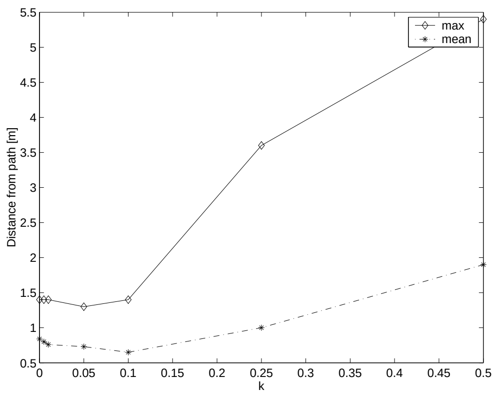
(b) Method two for path one



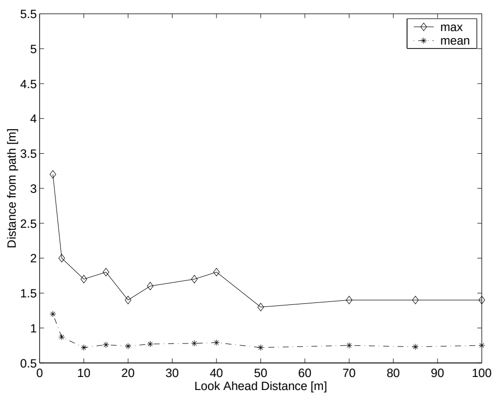
(c) Method one for path two



(d) Method two for path two

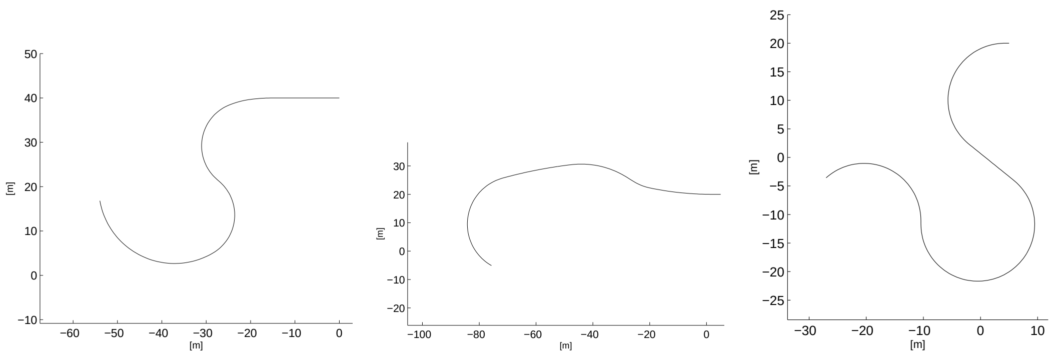


(e) Method one for path three



(f) Method two for path three

Figure 22: Max and mean distance from the path with different settings for the k-value and Look Ahead Distance ℓ for different paths and with $\sigma = 1m$. All the paths shown in Figure 23 were used in this test.



(a) Path 1 (b) Path 2 (c) Path 3

Figure 23: The three different paths used in Figure 22 and Table 2.

	Method one		**Method two**	
$\sigma[m]$	max[m] (k)	mean[m] (k)	max (ℓ)[m]	mean (ℓ)[m]
0.3	0.63 (0.005)	0.18 (0.075)	0.64 (70)	0.17 (90)
1	1.90 (0)	0.71 (0.05)	1.90 (100)	0.59 (70)
3	2.50 (0)	1.20 (0)	2.30 (70)	0.85 (100)
5	3.60 (0.005)	1.80 (0)	3.20 (85)	1.30 (100)

Table 1: The lowest values of the max and mean distance to path at different noise standard deviation σ. The values within parenthesis are the corresponding optimal k-value and Look Ahead Distance (ℓ) respectively. (Refer to Figure 21).

	Method one		**Method two**	
Path	max[m] (k)	mean[m] (k)	max (ℓ)[m]	mean (ℓ)[m]
1	1.90 (0)	0.71 (0.05)	1.90 (100)	0.59 (70)
2	0.49 (0)	0.26 (0)	1.10 (15)	0.44 (50)
3	1.30 (0.05)	0.65 (0.10)	1.30 (50)	0.72 (10)

Table 2: The lowest values for the max and mean distance to path for different paths. Noise standard deviation $\sigma = 1$. The values within parenthesis are the corresponding optimal k-value and Look Ahead Distance (ℓ) respectively. (Refer to Figure 22)

4.2 The effect of the three different behaviours

In this section we look at how the three different behaviours affect the path follower in different situations. We use Look Ahead Distance 12m and k-value 0.07. Note that the graphs showing the distance to the path in the figures are the real distance, and not based on the noisy position signal.

Figure 24 and Figure 25 show the situation when the vehicle is located on the path already from the beginning. In this case both ϕ_α and ϕ_β are essentially zero and the resulting steering angle equals the driver's steering angle ϕ_γ.

Figure 26 and Figure 27 show the same situation but with a simulated sinusoidal position noise with $\sigma = 0.3m$ (see section 4.1.1). The vehicle is located on the path already from the beginning. Apart from a spike on ϕ_α for Method two, both methods behave in the same manner. The reason for of this spike is discussed further in Section 4.2.2. The noise gives gives raise to disturbances in both ϕ_β and ϕ_α. This disturbance is propagated to the resulting steering angle. The resulting path tracking also shows a minor oscillation about the path as a result of the noise.

In Figure 28 and Figure 29, the vehicle starts 15 meters away from the path, approaches it and tracks it nicely to the end of the path. Figure 28 shows the behavior for Method One and Figure 29 for Method Two. No significant difference between the two methods can be seen in this example. The Resulting steering angle is composed of all three behaviors as long as the vehicle is far away from the path. About 65 seconds form start, both ϕ_α and ϕ_β become dominated by the noise, and the total behavior almost entirely consists of mimicking the steering angle (ϕ_γ).

For reference, Figure 30 and 31 shows how Follow the Carrot and Pure Pursuit behaves with noise.

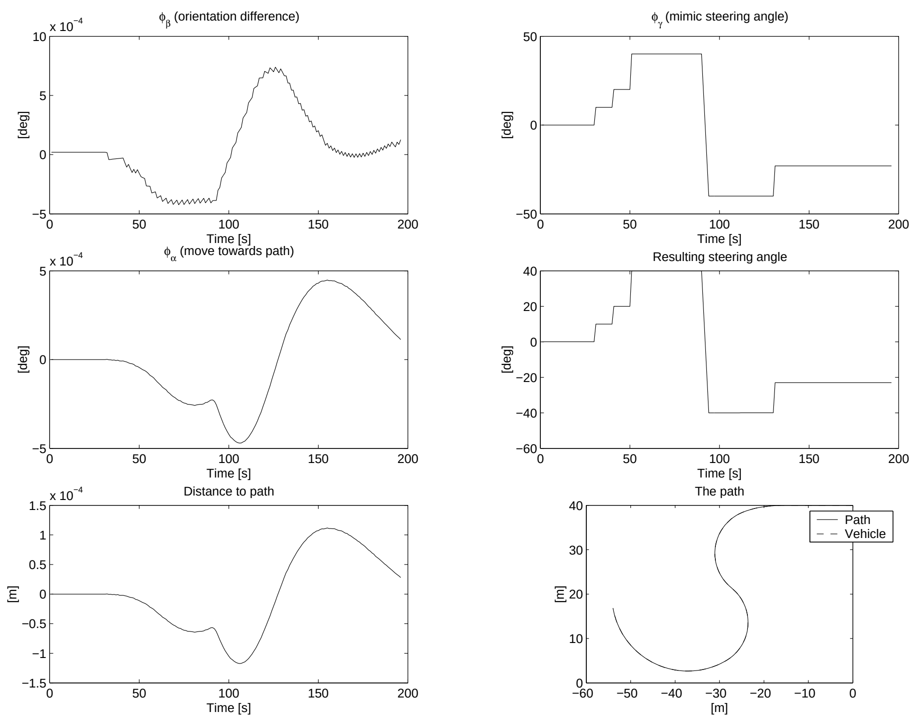


Figure 24: The three different behaviours with Method one and no noise.

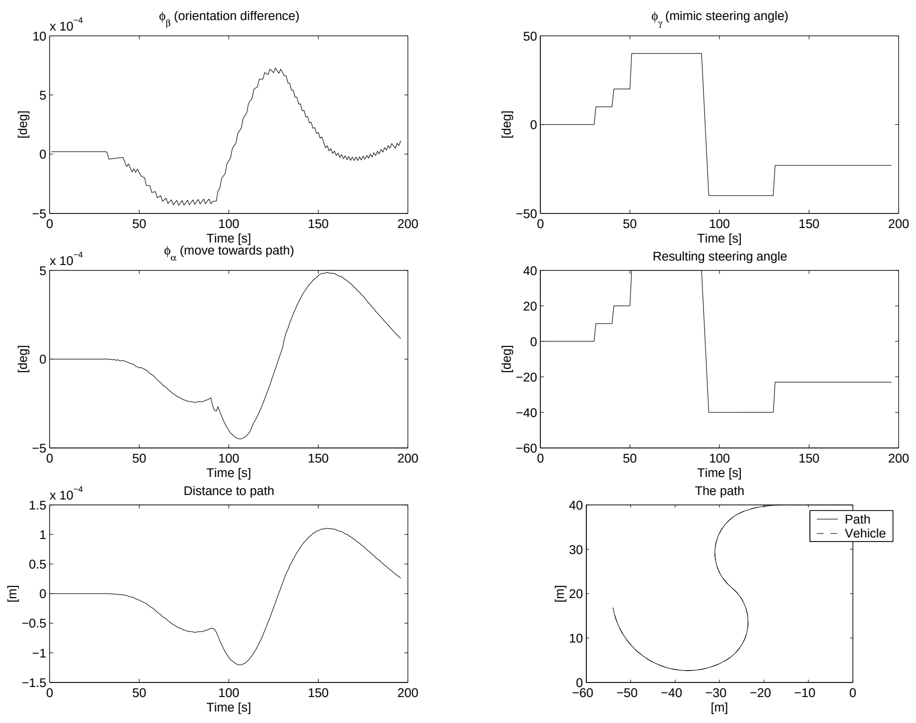


Figure 25: The three different behaviours with Method two and no noise.

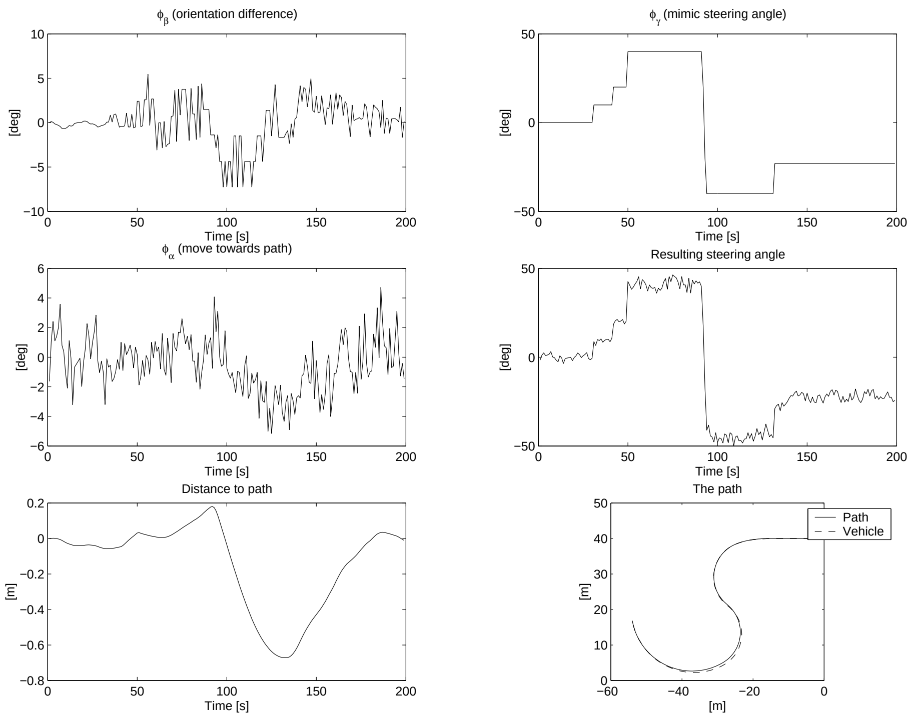


Figure 26: The three different behaviours with Method one and noise with 0.3 meter standard deviation.

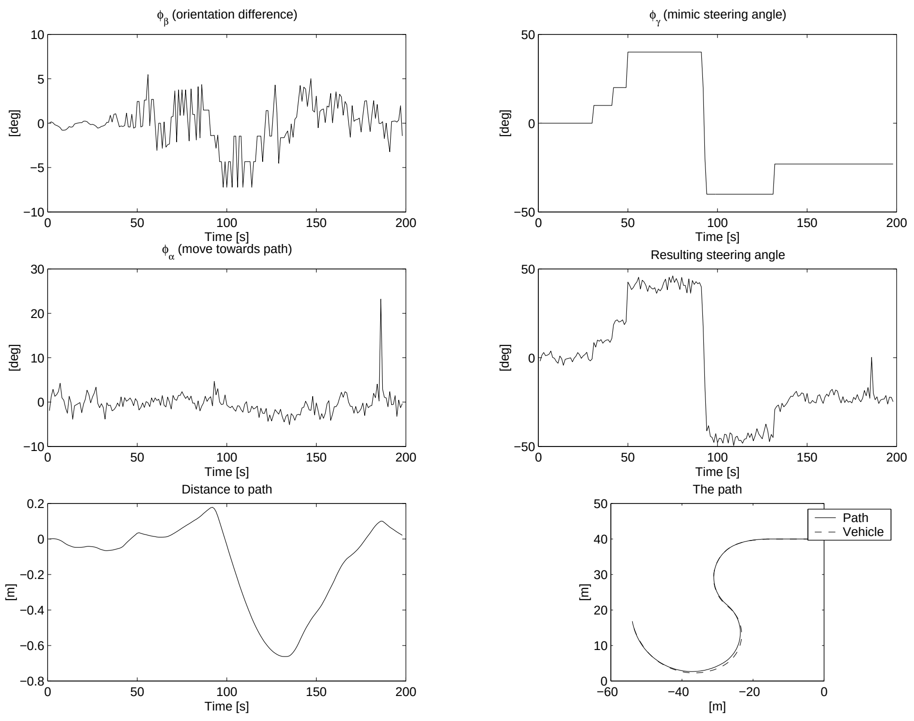


Figure 27: The three different behaviours with Method two and noise with 0.3 meter standard deviation.

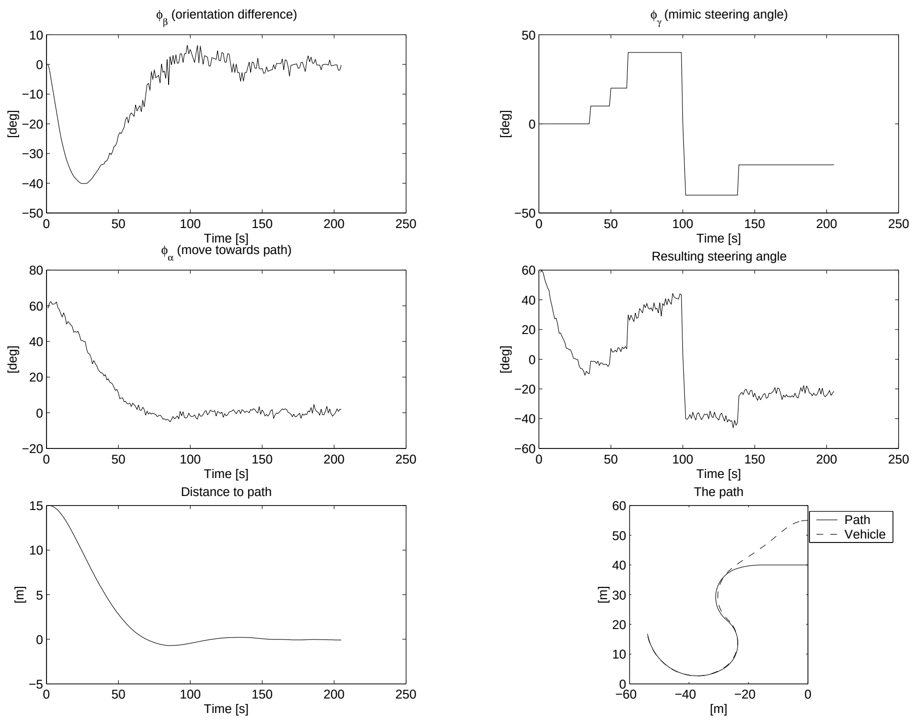


Figure 28: The three different behaviours with Method one, when starting 15 meters from the path and with noise with 0.3 meter standard deviation.

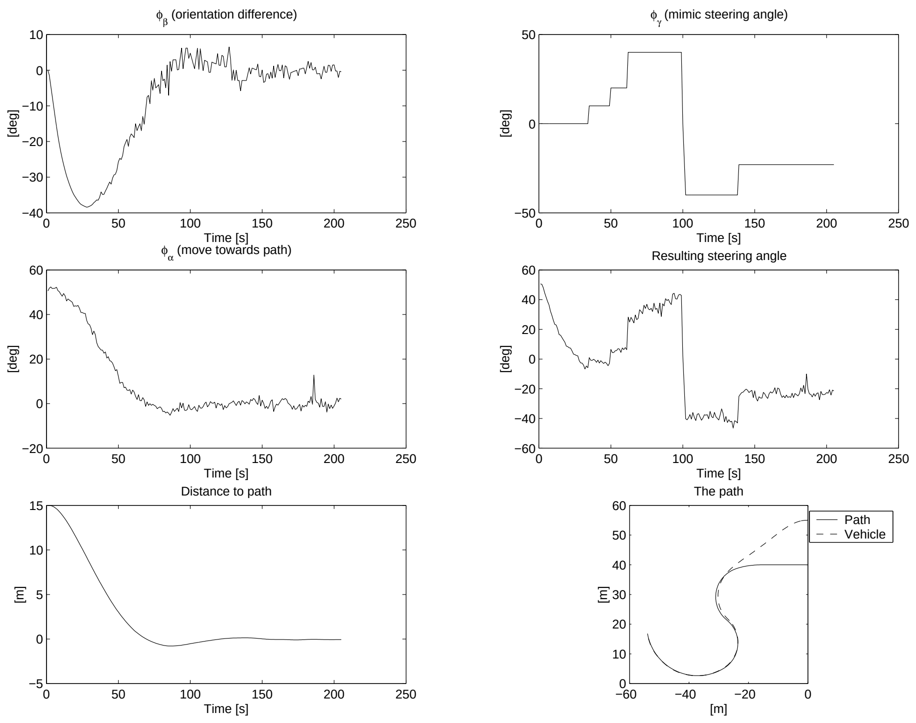


Figure 29: The three different behaviours with Method two, when starting 15 meters from the path and noise with 0.3 meter standard deviation .

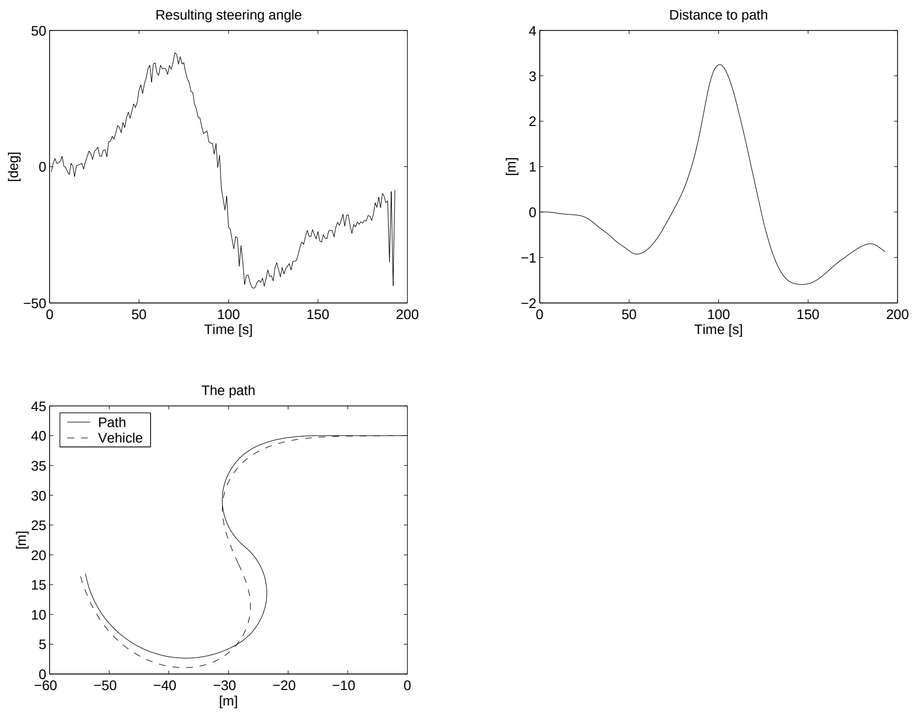


Figure 30: Follow the Carrot with 0.3 meter standard deviation noise.

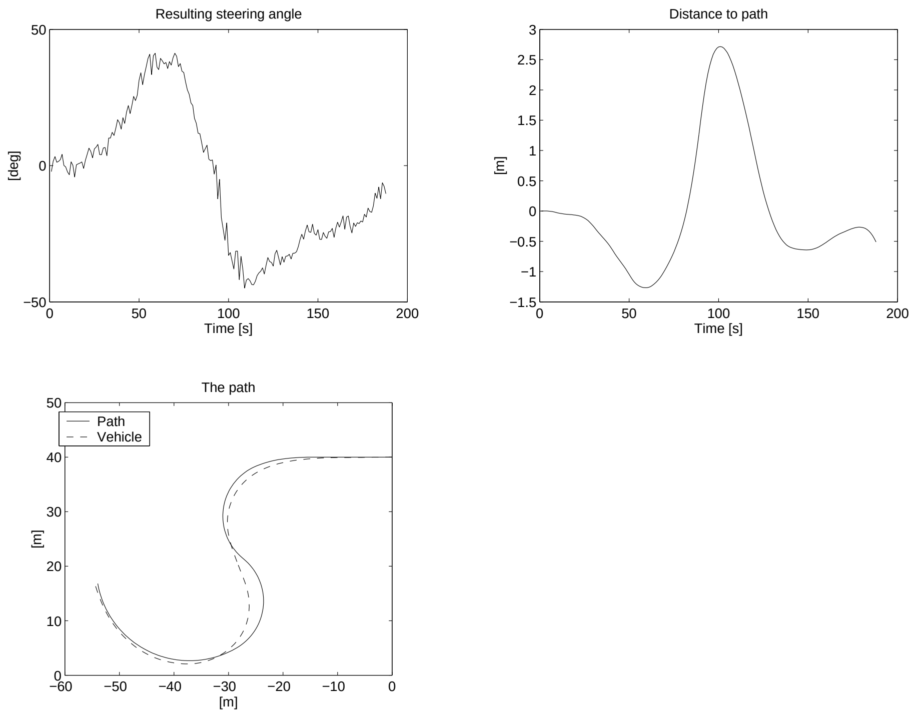


Figure 31: Pure pursuit with 0.3 meter standard deviation noise.

4.2.1 Limitations

As can be seen in Figure 19, Method one does not handle large noise very well. In this section, we will look at what this is caused by, and the difference between the two methods when applying a large noise. In Figure 32, the position given by the position sensors varies very much between each sample and when a noise with standard deviation $\sigma = 35m$ is applied, the measured position can be over 100 meters from where the vehicle actually is. This of course makes it very difficult to know exactly where you are, and even which side of the path you are on and therefore a bad performance can be expected. As mentioned earlier, this is not a realistic level of noise when controlling a forest machine (in the sense that it can not be tolerated), but in order to analyze why Method one does not work, it is interesting to look at what occurs at this noise level.

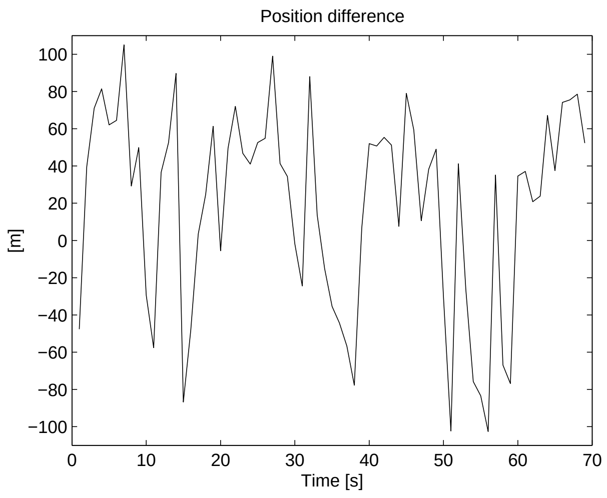
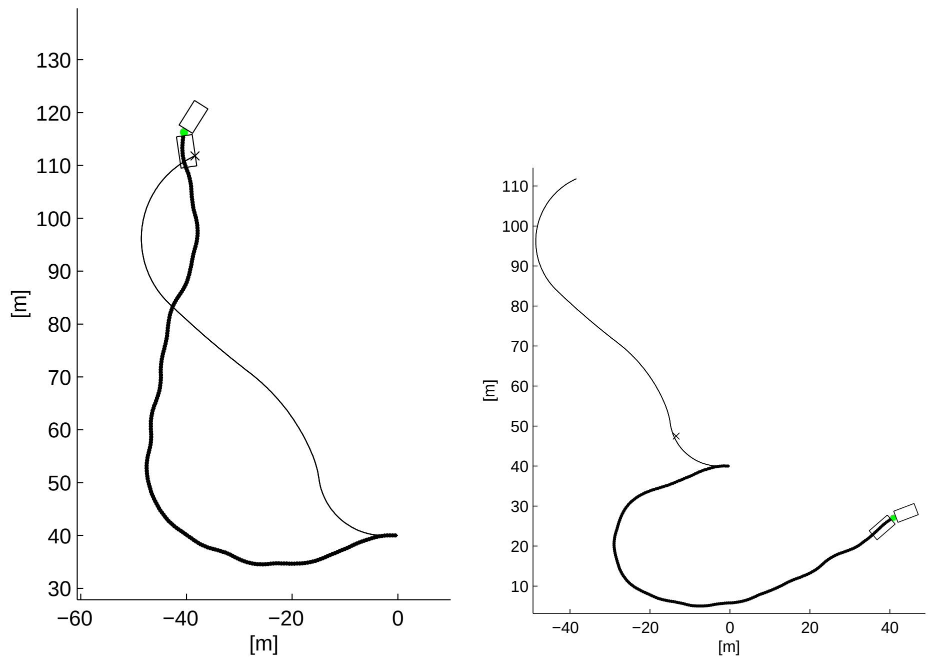


Figure 32: The difference between the real and the believed (noisy) position can be very large. Noise with standard deviation $\sigma = 35m$ is applied in this example

The only difference between the two methods is the way ϕ_α is calculated (i.e how to move back to the path), so the behaviour of Method one must be due to the way ϕ_α is computed. If we do not limit ϕ_α in method one to be within $\pm 90°$ (see Equation 2.3), the error arises at much lower noise levels than with the limitation. If the k-value is large, the error appears at lower noise levels than with a lower k-value, see Figure 33. This would suggest that ϕ_α in Method one becomes too large in certain situations. Consider Figure 34. In this example we provoke the error by letting the vehicle start outside of and approximately perpendicular to the path, but with no noise applied. The sum of the three behaviours is less than $-180°$, which means that the resulting steering angle ϕ_t changes sign and becomes positive, which in turn means that the vehicle turns in the wrong direction. If we would choose a smaller k-value in this example, ϕ_α would become smaller, ϕ_t would be larger than $-180°$ and the vehicle would turn in the correct direction, refer to Figure 35.

According to Equation 2.8, the steering angle for Method two is calculated as the Look Ahead Angle minus the orientation $(\psi - \theta)$, which means that the steering angle automatically has the right sign and magnitude, hence we do not get the problem described above in Method two.



(a) Method 1 with $k = 0.05$ and standard deviation $\sigma = 55$

(b) Method 1 with $k = 0.05$ and standard deviation $\sigma = 60$

Figure 33: With a lower k-value, Method one becomes less sensitive to noise (compare to Figure 19).

4.2.2 Endpoint correction

When the vehicle is avoiding an obstacle near the end of the path (or if it is beside the path for other reasons, such as noise in the position sensors), a problem can occur. Sometimes the vehicle does not return to the path quickly enough, with the result that it arrives beside the endpoint as in Figure 36(a). This problem is solved by setting the Look Ahead Point to the endpoint on the path if the following two conditions are satisfied:

- The distance between the vehicle and the path is large (larger than one meter

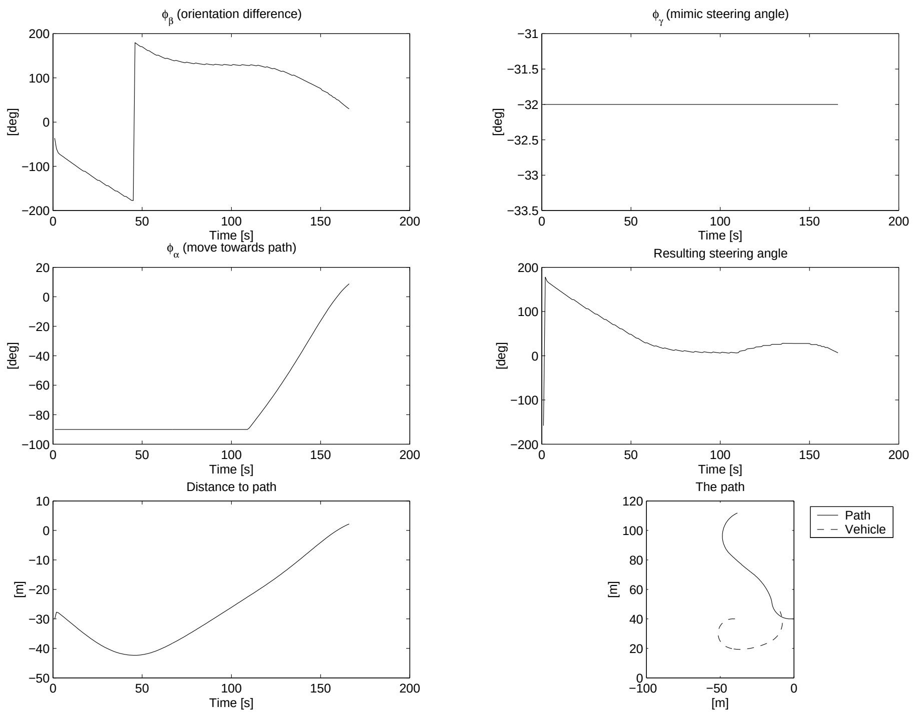


Figure 34: The behaviour of Method one when an error is provoked.

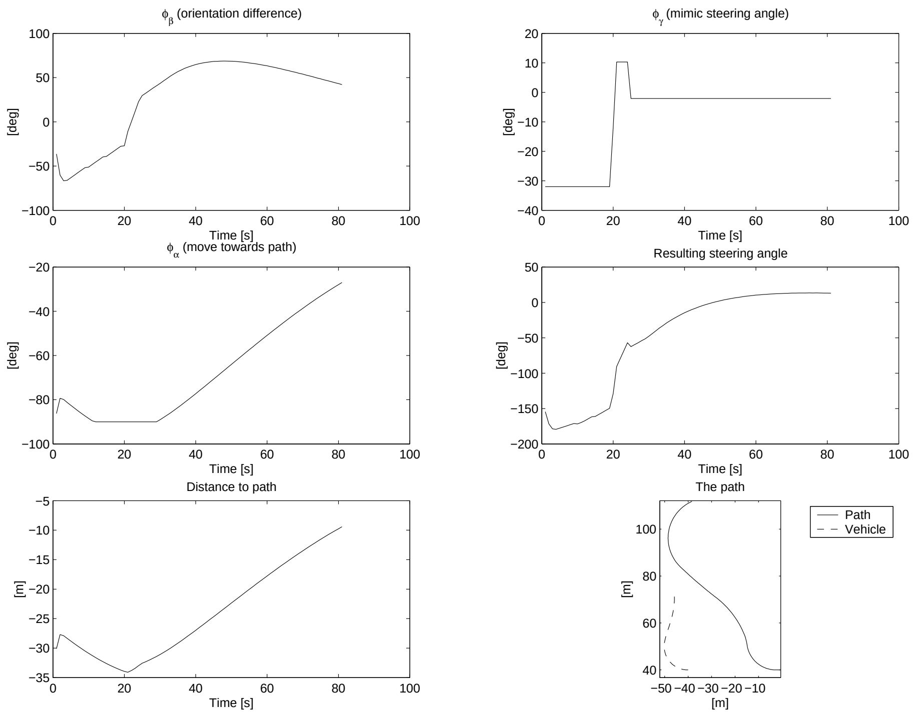
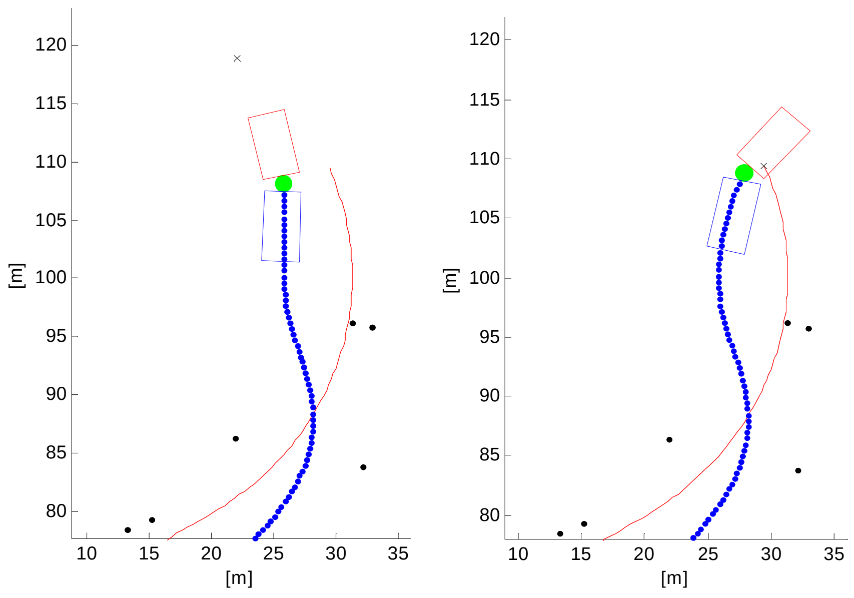


Figure 35: The same situation as in Figure 34 but with a smaller k-value which make the vehicle turn correctly.

in our current implementation).

- The Look Ahead Point is at or beyond the endpoint of the path.

Figure 36(b) shows the result when using the correction above. As we can see, the vehicle is able to return much faster to the path and thereby ends up closer to the endpoint. This correction can sometimes be seen in ϕ_α as a spike in the steering command when the vehicle is forced to steer harder towards the path (see Figure 27).



(a) Correction turned off (b) Correction turned on

Figure 36: The effect of applying a correction when the vehicle is not on the path and near the end.

4.3 Obstacle avoidance

In order to function in the forest machine application described in Section 1, the path tracker must be able to work together with routines for obstacle avoidance. Figure 37 shows the vehicle avoiding obstacles with the VFH+ method [UB98] while tracking the path with the Follow the Past method. The test shows that the Follow the Past method works well in combination with obstacle avoidance. Note that the deviation from the path is caused by the obstacle avoiding system when obstacles are detected near the vehicle.

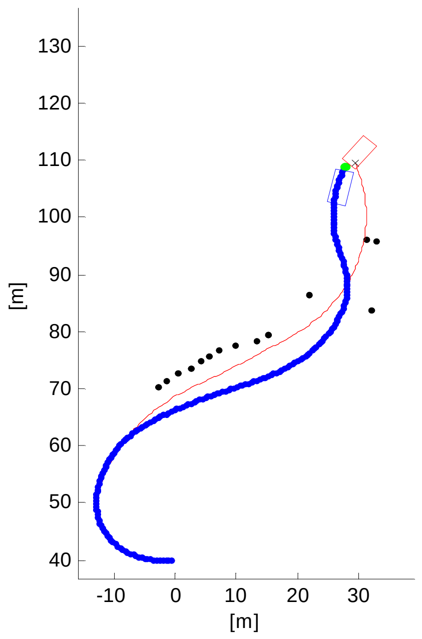


Figure 37: Avoiding obstacles with the VFH+ method in combination with Follow The Past works well. The deviation from the path is due to avoiding the obstacles, no noise is applied here.

5 Conclusions

We have implemented and evaluated a new path tracking algorithm that uses the operator's steering commands as input, in addition to position and heading information. The algorithm has superior performance when compared to standard methods like the Pure Pursuit and Follow the Carrot algorithms. The two different methods for computing ϕ_α, the behaviour responsible for moving towards the path, has a comparable performance at small noise levels in the position data. When the noise reaches higher levels Method 2 has better performance than Method 1, which tends to become unstable at high noise levels. Method 2 has not shown any signs of instability even at very high noise levels. This is required from a good path tracking algorithm.

The optimal settings for the Look Ahead Distance ℓ and the k-value respectively is a trade-off between the ability to quickly return to the path and the sensitivity to high noise in the position values. The exact nature of these relations will be investigated further in our future research.

6 Acknowledgements

This work was Funded by *The Kempe Foundation* and *VINNOVA* under Award No. 2003-00777.

References

[Bar01] Matthew J. Barton. *Controller Development and Implementation for Path Planning and Following in an Autonomous Urban Vehicle.* Undergraduate thesis, University of Sydney, November 2001.

[Cou92] R. Craig Coulter. *Implementation of the Pure Pursuit Path Tracking Algorithm.* Technical Report CMU-RI-TR-92-01, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1992.

[Hel02] Thomas Hellström. *Autonomous Navigation for Forest Machines.* Pre-study, University of Umeå, aug 2002.

[Rin03] Ola Ringdahl. *Path Tracking and Obstacle Avoidance for Forest Machines.* Master's thesis, University of Umeå, apr 2003.

[sof03] softSurfer. *About Lines and Distance to a Line*, 20 August 2003, `http://geometryalgorithms.com/Archive/algorithm_0102/`.

[UB98] I. Ulrich and J. Borenstein. *VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots. IEEE Int. Conf. on Robotics and Automation*, pages 1572–1577, May 1998.

A Variables used

ϕ	The vehicle's steering angle
ϕ'	The recorded steering angle
θ	Orientation of the vehicle
θ'	The recorded orientation
η	Heading of the vehicle
k	A constant $[deg/m]$ used in Method one
d	The closest distance to the path
ψ	Look Ahead Angle
l	Look Ahead Distance
σ	Standard deviation of the noise

Steering behaviours:

ϕ_α	Turn towards the recorded orientation
ϕ_γ	Mimic the recorded steering angle.
ϕ_α	Move towards the path.
ϕ_t	Fused turning behaviour (the sum of the angles above)