Corridor Library

Dominik Petrich

December, 2020

Abstract

This documentation provides the technical background of the Corridor library. It consists of an introduction to cubic spline interpolation, which is used as reference line approximation for a corridor segment. The applied algorithms for creating a cubic spline from a set of support points are discussed, including the influence of different boundary constraints and how to parameterize the curve by its arc-length. Furthermore, the transformation of an arbitrary Cartesian state vector into Frenet coordinates of a reference line is described together with error propagation. Last but not least, this document provides a description of some probabilistic association functions which are applied to formulate semantic relationships between an object and a corridor.

Contents

1 Introduction

For deriving motion constraints imposed by roads it is common to approximate the course of a traffic lane by a reference curve. When designing such a curve, it is helpful to consider that roads are usually constructed to enable comfortable driving with low lateral jerk [Hasberg et al., 2012]. Thus, the variation of the road curvature can be considered as smooth and the reference line model should ideally reflect this characteristic. This can be obtained by representing the reference line as a two times differential function. Besides an accurate interpolation of road lane courses it enables also the usage of the Frenet-Serret formulas [D. Wang et al., 2015]. They allow us to describe the kinematics of a particle moving along a curve. A property, which is useful for both, modeling and interpretation of object kinematics along a corridor.

However, a corridor isn't sufficiently represented by a reference line. A minimal corridor model requires also a representation of the left and right boundary. These boundaries define the corridor width at arbitrary positions along the reference line and specify the usable area of the lane. Within the Corridor library each boundary is defined as an individual polyline in Frenet coordinates, e.g. it consists of a list signed distances d at specific arc-lengths l. As shown in Figure 1, all three lines combined declare the geometric representation of a corridor object. To assure maximal flexibility in the geometric model, both boundary lines can have their own individual sampling rate, which is independent of the reference line's sampling rate. A corridor can be used to model traffic lanes, including special lanes such as bus or bike lanes. Depending on the information

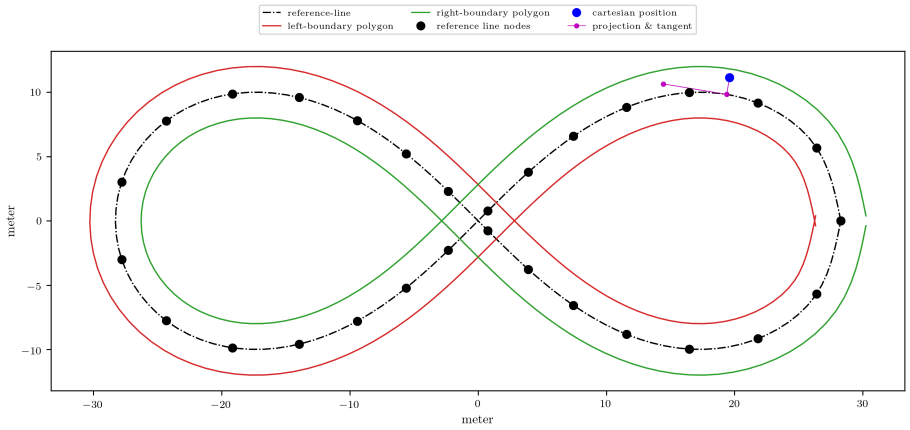

Figure 1: Example of an arbitrary corridor consisting of three lines which is constructed by a cubic spline interpolation of the reference line nodes. Both boundary polygons are constructed with a constant offset of 2 meters to the reference line. The blue node represents an arbitrary Cartesian position. The perpendicular projection of this point onto the reference line is visualized by the magenta line. It also shows the tangent vector at the projection point.

density of the underlying map data, it is also possible to model side- and crosswalks for VRUs.

This report provides the technical background about how a corridor can be generated from map data and how it is used for kinematic state transformation, as well as object to corridor assignments. After an introduction to the properties of cubic spline interpolation, this report gives an overview how cubic splines are implemented and used within the Corridor library. Furthermore, details are provided about how Frenet–Serret formulas can be applied to express the kinematic properties of a particle in curvilinear frame of the cubic spline. This transformation is used to convert a Cartesian state into a Frenet state with respect to a Corridor's reference line, including error propagation. This documentation closes with the introduction of probabilistic assignment features, which are applied to assign any observed objects to the corridor in its surrounding.

2 Cubic Spline Interpolation

Cubic spline interpolation approximates an arbitrary curve described by a sequence of support points (also known as control points, knots or nodes). The resulting function interpolates this sequence by piecewise cubic polynomials. Among all twice continuously differentiable functions, the **cubic spline** $\mathbf{s}(l)$

yields the minimal curvature energy:

$$\mathbf{E} = \int [\mathbf{s}''(l)]^2 dl \tag{1}$$

As a consequence the resulting interpolation provides minimal oscillations in between the supporting points. Therefore, it is often preferred over pure polynomial interpolation since the interpolation error can be kept small and constrained by the density of sample points. In contrast to curve-fitting algorithms, like (non-) linear regression or NURBS, all sample points are part of the resulting curve. This makes cubic spline interpolation appropriate to use for curve approximation, but not if the sample points are subject to noise and outliers. If smoothing is necessary, it should be applied previously to the cubic spline interpolation.

When deriving the interpolant, a general distinction can be drawn between local and global cubic splines. For local cubic splines, or Hermite splines, the polynomial is specified by the values and first derivatives at the end of the corresponding parameter interval domain. Making the Hermite splines continuous over all pieces and guarantees at least C^1 continuity.

For global cubic splines the first derivative at internal support points are not specified but instead computed from imposed continuity conditions over the whole spline. Thus, changing the location of one support point can potentially change the shape of the entire spline. Consequently, the global spline guarantees C^2 continuity which makes it preferable for the use case of the Corridor library.

2.1 Definition

Given a set of $n+1$ points, $\mathbf{p}_i = [p_{x,i}, p_{y,i}]^T$ for $i = 0, \ldots, n$, the cubic spline interpolation is the process of constructing the global curve $\mathbf{s} = \{\mathbf{s}_i | i \in [0, n]\}$ as piecewise composition of n planar cubic polynomials:

$$\mathbf{s}_i(l) = \begin{bmatrix} s_x(l) \\ s_y(l) \end{bmatrix} = \begin{bmatrix} a_{x,i} + b_{x,i}(l-l_i) + c_{x,i}(l-l_i)^2 + d_{x,i}(l-l_i)^3 \\ a_{y,i} + b_{y,i}(l-l_i) + c_{y,i}(l-l_i)^2 + d_{y,i}(l-l_i)^3 \end{bmatrix} \tag{2}$$

In here, l denotes the curve parameter, l_i corresponds to the value at point p_i. Segment i interpolates the interval $\mathbf{s}_i(l) \in [l_i, l_{i+1}]$ with end points $\mathbf{s}_i(l_i) = \mathbf{p}_i$ and $\mathbf{s}_i(l_{i+1}) = \mathbf{p}_{i+1}$. The set of unknown polynomial coefficients $\{a, b, c, d\}$ are determined by interpolating the support points and satisfying smoothness conditions at inner nodes. Within the Corridor library, following conditions are evaluated:

1. All support points must interpolate at both ends of a segment, i.e. for $i \in [0, n-1]$:
$$\mathbf{s}_i(l_i) = \mathbf{p}_i \qquad \mathbf{s}_i(l_{i+1}) = \mathbf{p}_{i+1} \tag{3}$$
2. First and second derivatives must be continuous at every internal support point, i.e. for $i \in [1, n-1]$:
$$\begin{aligned} \mathbf{s}'_{i-1}(l_i) &= \mathbf{s}'_i(l_i) \\ \mathbf{s}''_{i-1}(l_i) &= \mathbf{s}''_i(l_i) \end{aligned} \tag{4}$$

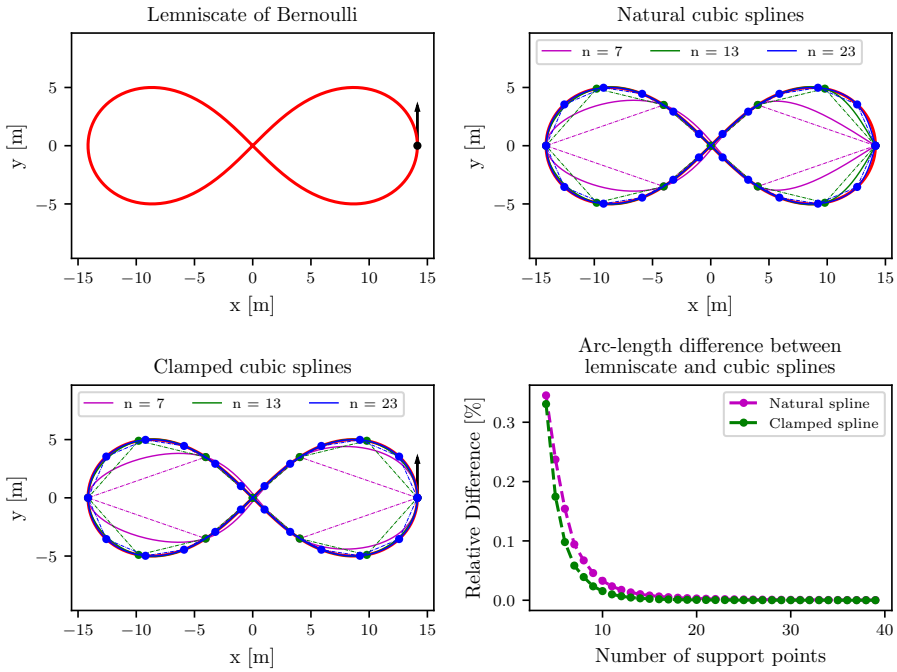

Figure 2: Approximation of lemniscate of Bernoulli ($\alpha = 10$) by different cubic splines and number of support points. The upper right picture shows a selection of interpolations with natural cubic splines, the lower left with clamped cubic splines. The lower right pictures gives an overview about the arc-length differences between the two interpolation methods based on the number of nodes. As expected a higher sampling rate yields to a better approximation of the curve and therefore a similar arc-length. In general, the clamped spline provides a better approximation result with viewer nodes, especially around start and end of the curve. The black arrow represents the scaled tangent vector at the first and last node of the lemniscate, which is used as additional constrain in the clamped cubic spline approximation.

3. Either one of the following conditions at the two end points of the spline at $\mathbf{s}_0(l_0) = \mathbf{p}_0$ and $\mathbf{s}_{n-1}(l_n) = \mathbf{p}_n$:

   (a) **Natural cubic spline.** Both end points are modeled as curvature free:

   $$\mathbf{s}_0''(l_0) = 0 \qquad \mathbf{s}_{n-1}''(l_n) = 0 \tag{5}$$

   (b) **Clamped cubic spline.** Both tangents $\mathbf{t}$ at the end points are known:

   $$\mathbf{s}_0'(l_0) = \mathbf{t}(l_0) \qquad \mathbf{s}_{n-1}(l_n) = \mathbf{t}(l_n) \tag{6}$$

As shown in figure 2 a clamped cubic spline gives a more accurate approximation to the initial curve, especially in the vicinity of start and end point. However, it requires additional knowledge about the tangent vectors which is not always given. Consequently, the Corridor library constructs a clamped cubic spline whenever the two tangent vectors are provided and a natural cubic spline if not.

Since the x- and y-component of one spline segment are separated from each other, each coordinate can be determined individually. For the remainder of this section we'll focus solely on the x-coordinate of the spline, the y-component are determined accordingly.

Due to the cubic definition is the second derivative of each polynomial a linear function

$$s''_{x,i}(l) = m_{x,i} + \frac{m_{x,i+1} - m_{x,i}}{l_{i+1} - l_i}(l - l_i) \tag{7}$$

where $m_{x,i} = m_x(l_i)$ and $m_{x,i+1} = m_x(l_{i+1})$ denote the moments at their respective support points. By definition corresponds the moment vector $\mathbf{m}_i = [m_{x,i}, m_{y,i}]^T$ to the curvature vector of these points. The unknown moments can be calculated by imposing the continuity conditions (1) and (2), leading to a linear equation system for the moments at the inner nodes of the spline for $i \in [1, n-2]$:

$$\begin{aligned} h_i m_{x,i} + 2(h_i + h_{i+1}) m_{x,i+1} + h_{i+1} m_{x,i+2} = \\ \frac{6}{h_i} p_{x,i} - (\frac{6}{h_i} + \frac{6}{h_{i+1}}) p_{x,i+1} + \frac{6}{h_{i+1}} p_{x,i+2} \end{aligned} \tag{8}$$

with $h_i = l_{i+1} - l_i$ denoting the length of segment i. The moments at $i = 0$ and $i = n$ are determined using condition (3) depending on the choice for the end condition of the spline. The polynomial coefficients are evaluated based on the support points and the yet unknown moments:

$$\begin{aligned} a_{x,i} &= p_{x,i} & b_{x,i} &= \frac{1}{h_i}(p_{x,i+1} - p_{x,i}) - \frac{h_i}{6}(2m_{x,i} + m_{x,i+1}) \\ c_{x,i} &= \frac{1}{2} m_{x,i} & d_{x,i} &= \frac{1}{6h_i}(m_{x,i+1} - m_{x,i}) \end{aligned} \tag{9}$$

Extracting the vector of moments $\mathbf{m}_x = [m_{x,0}, ..., m_{x,n}]$ and rearranging the linear system (8) in matrix notation leads to

$$\mathbf{M}\mathbf{m}_x = \mathbf{L}\mathbf{q}_x. \tag{10}$$

where $\mathbf{q}_x$ contains the external information about the support points, and, potentially, the tangents at start and end point. Besides the unknown moment vector the definition depends on the chosen end point condition.
Under a **natural cubic spline** interpolation, the matrices $\mathbf{M}^{\mathrm{NCS}}$, $\mathbf{L}^{\mathrm{NCS}}$ and

support vector $\mathbf{q}_x^{\mathrm{NCS}}$ are defined as follows:

$$\mathbf{M}^{\mathrm{NCS}} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \dots & 0 & 0 & 1 \end{bmatrix}_{(n+1)\times(n+1)} \tag{11}$$

$$\mathbf{L}^{\mathrm{NCS}} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ \frac{6}{h_0} & (\frac{-6}{h_0}+\frac{-6}{h_1}) & \frac{6}{h_1} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \frac{6}{h_{n-2}} & (\frac{-6}{h_{n-2}}+\frac{-6}{h_{n-1}}) & \frac{6}{h_{n-1}} \\ 0 & \dots & 0 & 0 & 0 \end{bmatrix}_{(n+1)\times(n+1)} \tag{12}$$

$$\mathbf{q}_x^{\mathrm{NCS}} = [\, p_{x,0}, \dots, p_{x,n} \,]^T_{(n+1)\times 1} \tag{13}$$

For a **clamped cubic spline** interpolation, the system is defined by:

$$\mathbf{M}^{\mathrm{CCS}} = \begin{bmatrix} 2h_0 & h_0 & 0 & \dots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \dots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix}_{(n+1)\times(n+1)} \tag{14}$$

$$\mathbf{L}^{\mathrm{CCS}} = \begin{bmatrix} -6 & \frac{-6}{h_0} & \frac{6}{h_0} & 0 & \dots & 0 \\ 0 & \frac{6}{h_0} & (\frac{-6}{h_0}+\frac{-6}{h_1}) & \frac{6}{h_1} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \frac{6}{h_{n-2}} & (\frac{-6}{h_{n-2}}+\frac{-6}{h_{n-1}}) & \frac{6}{h_{n-1}} & 0 \\ 0 & \dots & 0 & \frac{6}{h_{n-1}} & \frac{-6}{h_{n-1}} & 6 \end{bmatrix}_{(n+1)\times(n+3)} \tag{15}$$

$$\mathbf{q}_x^{\mathrm{CCS}} = [\, t_{x,0}, p_{x,0}, \dots, p_{x,n}, t_{x,n} \,]^T_{(n+3)\times 1} \tag{16}$$

Compared to the natural spline interpolation not only the definition of coefficients are different but also the size of matrix $\mathbf{L}^{\mathrm{CCS}}$ and vector $\mathbf{q}_x^{\mathrm{CCS}}$ is increased. This is necessary to hold the additional information about the tangent vectors at the end points.

Both, the natural cubic spline and the clamped cubic spline boundary conditions yield a system of $n+1$ linear equations for the $n+1$ unknown moments. Moreover, in both cases is matrix $\mathbf{M}$ a tridiagonal matrix. As a consequence the unknown moment vector $\mathbf{q}$ can be determined in $\mathcal{O}(n)$ by solving

$$\mathbf{m}_x = \mathbf{M}^{-1}\mathbf{L}\mathbf{q}. \tag{17}$$

Once the moments are found, the polynomial coefficients can be determined by equations (9).

It is worth mentioning, that both boundary conditions could be mixed. A spline could be modeled at the beginning using a tangent vector and strain free at the end.

2.2 Arc-length parametrization

To enable a simple application of the planar Frenet-Serret formulas, the resulting curve $\mathbf{s}(l)$ must be parameterized according to the arc-length. Thus, the arc-length between $\mathbf{s}(l_a)$ and $\mathbf{s}(l_b)$ must be equal to the difference of the curve parameter values:

$$L_{a,b} = \int_{l_a}^{l_b} \left\|\mathbf{s}'(\tau)\right\| d\tau \stackrel{!}{=} l_b - l_a \qquad \text{for} \quad l_a, l_b \in [l_0, l_n] \tag{18}$$

Since the shape of curve $\mathbf{s}(l)$ remains unknown at the beginning of the interpolation, another parametrization needs to be found first.

As presented by Hasberg et al., 2012, this can be accomplished by recursively calculating the chord-length of the polyline which interpolates the known support points:

$$u_{i+1} = u_i + \left\|\mathbf{p}_{i+1} - \mathbf{p}_i\right\|. \tag{19}$$

The initial chord-length of the first point is set to zero $u_0 = 0$. Applying the interpolation method described in the previous section results in the chord-length parameterized curve $\mathbf{s}(u)$. However, any other parametrization will work too as long its is differentiable and monotonic increasing.

Converting any arbitrary parameterized curve into arc-length parametrization can be achieved by a simple two-step process [H. Wang et al., 2002]:

1. Compute arc-length l as a function of parameter u: $l = A(u)$.
2. Compute inverse of the arc-length function $u = A^{-1}(l)$ and substituting in $\mathbf{s}(l) = \mathbf{s}(A^{-1}(l))$ leads to an arc-length parameterized curve.

For cubic splines is function $A(u)$ a geometric integration:

$$A(u) = \int_{u_0}^{u} \sqrt{(s'_x(u))^2 + (s'_y(u))^2}\, du \tag{20}$$

which cannot be computed analytically and must be evaluated numerically.

Within the Corridor library, the arc-length integral (20) is approximated by a Gauss–Legendre quadrature algorithm of order 3, as discussed by Floater et al., 2006. This adaptive Gaussian integration method divides every curve segment into two halves and integrates each sub-interval. The sum of these two intervals is compared with the integral computed over the entire segment. Is the difference larger than the desired accuracy, the same procedure is recursively applied on the sub-intervals; otherwise it returns the sum of the two halves. While such a bisection method achieves high accuracy, it converges rather slowly compared to other known algorithms as discussed in H. Wang et al., 2002. Besides formulating the arc-length computation as a Newton-Raphson root finding problem, Walter et al. (1996) proposes an approximated closed-form solution by a one or two-span Bézier curve.

However, the arc-length parametrization can be performed during the map initialization, prior to run-time queries. The only run-time costs occur when querying a curve position at a specific arc-length for determining the curve segment and evaluating the corresponding cubic polynomial.

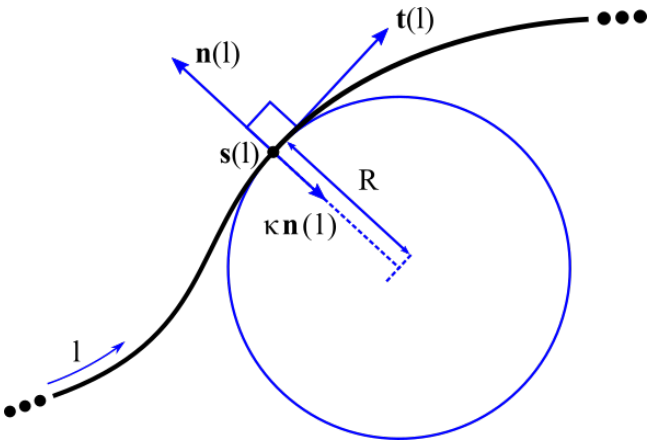

Figure 3: Curve point $\mathbf{s}(l)$ with tangent vector $\mathbf{t}(l)$, normal vector $\mathbf{n}(l)$, curvature $\kappa(l)$ and osculating circle with radius R.

3 Constructing a Frenet frame

Let $\mathbf{s}(l)$ be a regular, two times differentiable function, describing the shape of a planar curve parameterized by its arc-length, $l \in I = [l_{min}, l_{max}]$. The first derivative of $\mathbf{s}(l)$ within the parameter interval domain I is called **tangent vector**:

$$\frac{d}{dl}\mathbf{s}(l) = \mathbf{t}(l) \tag{21}$$

For regular curves, i.e. $\mathbf{t}(l) \neq 0$, the tangent vector points always in the direction of increasing arc-length. Due to the arc-length parameterization of curve $\mathbf{s}(l)$ we can assume that $\|\mathbf{t}(l)\| = 1$ and

$$\mathbf{t}(l)^T \mathbf{t}(l) = 1. \tag{22}$$

Differentiation of this expression results in

$$\frac{d}{dl}\mathbf{t}(l)^T\ \mathbf{t}(l) = 0. \tag{23}$$

Excluding the case $\frac{d}{dl}\mathbf{t}(l) = 0$, leads to the conclusion that the tangent's first derivative is orthogonal to the tangent itself. Without loss of generality it follows:

$$\frac{d}{dl}\mathbf{t}(l) = \kappa(l)\,\mathbf{n}(l) \tag{24}$$

where the scalar, $\kappa(l) = \left\|\frac{d}{dl}\mathbf{t}(l)\right\| \geq 0$, denotes the curvature. Vector $\mathbf{n}(l)$ is the primary normal vector of curve $\mathbf{s}(l)$. As shown in Figure 3, the product $\kappa(l)\,\mathbf{n}(l)$ points always towards the center of the osculating circle.

Both vectors, $\mathbf{t}(l)$ and $\mathbf{n}(l)$, are orthogonal to each other and compose a planar Cartesian frame[1]. This moving frame is called the **Frenet frame**, or moving frame of planar curve $\mathbf{s}(l)$. Within domain I of curve $\mathbf{s}$ it is consistently defined by the triple:

$$\frac{d}{dl}\mathbf{s}(l) = \mathbf{t}(l); \qquad \frac{d}{dl}\mathbf{t}(l) = \kappa(l)\,\mathbf{n}(l); \qquad \frac{d}{dl}\mathbf{n}(l) = -\kappa(l)\,\mathbf{t}(l) \tag{25}$$

The third term of these planar Frenet equations can be derived by the same argumentation which lead to expression (24). Differentiation of $\mathbf{n}(l)^T\mathbf{n}(l) = 1$ yields:

$$\frac{d}{dl}\mathbf{n}(l)^T\mathbf{n}(l) = 0. \tag{26}$$

Thus, the first derivative of the normal vector must be orthogonal to the normal vector. Since all vectors are defined in a 2-dimensional coordinate frame, the normal's derivative must be parallel to the tangent vector:

$$\frac{d}{dl}\mathbf{n}(l) = \alpha\,\mathbf{t}(l) \tag{27}$$

The scalar factor α can be obtained by evaluating the derivative of the tangent-normal product: $\mathbf{n}(l)^T\mathbf{t}(l) = 0$:

$$\frac{d}{dl}\mathbf{n}(l)^T\mathbf{t}(l) + \mathbf{n}(l)^T\frac{d}{dl}\mathbf{t}(l) = 0 \tag{28}$$

Utilizing the second term of the Frenet equations, c.f. equation (24), leads to:

$$\frac{d}{dl}\mathbf{n}(l)^T\mathbf{t}(l) = -\kappa(l) \tag{29}$$

Substituting (27) in (29) results in $\alpha = -\kappa(l)$.

At any arbitrary arc-length $l_p \in I$ the Frenet equations can be utilized to construct the translation vector and rotation matrices between Frenet frame and Cartesian frame. The translation is defined by the position $\mathbf{s}(l_p)$, while the rotation matrices are specified by the orthogonal vector base of the Frenet frame:

- from Cartesian to Frenet frame

$${}^F\mathbf{R}^C_p = \begin{bmatrix} \cos\psi_p & -\sin\psi_p \\ \sin\psi_p & \cos\psi_p \end{bmatrix} = \begin{bmatrix} \mathbf{t}_p^T \\ \mathbf{n}_p^T \end{bmatrix} \tag{30}$$

- from Frenet to Cartesian frame

$${}^C\mathbf{R}^F_p = \begin{bmatrix} \cos\psi_p & \sin\psi_p \\ -\sin\psi_p & \cos\psi_p \end{bmatrix} = \begin{bmatrix} \mathbf{t}_p & \mathbf{n}_p \end{bmatrix} \tag{31}$$

To simplify the expressions above, all Frenet values interpolated at arc-length l_p are denoted with suffix $\{\}_p$. The orientation $\psi_p \in [-\pi, \pi]$ denotes the angle between tangent vector and x-axis of the Cartesian frame.

[1] Including the normal vector $\mathbf{b} = \mathbf{t}(l) \times \mathbf{n}(l)$ of the planar surface defines a 3-dimensional Cartesian frame.

3.1 Time derivatives

Imagine a particle which is moving along the curve $\mathbf{s}(l)$. Given the start position of the particle $l_p(t_0)$ and a velocity function $v_p(t)$ all future positions, velocities and accelerations can be expressed in the stationary Cartesian frame of the $n+1$ support points. Those time derivatives are needed to describe the kinematic attributes of a moving particle on or around the curve.
The time derivative for an arbitrary position on the spline curve is given by:

$$\begin{aligned} \frac{d}{dt}\mathbf{s}(l(t)) &= \frac{d}{dl}\mathbf{s}(l(t))\frac{dl}{dt} = \mathbf{t}(l)\frac{dl}{dt} \qquad \text{with} \quad \frac{dl}{dt} = v(t) \\ \mathbf{v}(l(t)) &= v(t)\,\mathbf{t}(l) \end{aligned} \tag{32}$$

As expected, the time derivative of the position leads to the velocity $\mathbf{v}(l)$ of this point along the curve. It is built by the product between the absolute velocity of the point $v(t)$ and the tangent vector $\mathbf{t}(l)$.
The time derivative of the tangent and normal vector can be obtained in an analogous manner by utilizing the Frenet formulas (25).
Tangent vector:

$$\begin{aligned} \frac{d}{dt}\mathbf{t}(l) &= \frac{d}{dl}\mathbf{t}(l)\frac{dl}{dt} = \kappa\,\mathbf{n}(l)\,\frac{dl}{dt} = \underbrace{\kappa(l)\,v(t)}_{\dot{\psi}(l(t))}\,\mathbf{n}(l) \\ &= \dot{\psi}(l(t))\,\mathbf{n}(l) \end{aligned} \tag{33}$$

The product of curvature value and absolute velocity represents the angular velocity $\dot{\psi}(l,t)$.
Normal vector:

$$\begin{aligned} \frac{d}{dt}\mathbf{n}(l) &= \frac{d}{dl}\mathbf{n}(l)\frac{dl}{dt} = -\kappa(l)\,\mathbf{t}(l)\,v(t) \\ &= -\dot{\psi}(l(t))\,\mathbf{t}(l) \end{aligned} \tag{34}$$

These time derivatives are applied to derive the velocity derivative of the particle:

$$\begin{aligned} \frac{d}{dt}\mathbf{v}(l(t)) &= \frac{d}{dt}v(t)\,\mathbf{t}(l) = a(t)\,\mathbf{t}(l) + v(t)\,\frac{d}{dt}\mathbf{t}(l) \\ \mathbf{a}(l(t)) &= a(t)\,\mathbf{t}(l) + \kappa(l)\,v^2(t)\,\mathbf{n}(l) \end{aligned} \tag{35}$$

3.2 Frenet coordinates

As visualized in Figure 4, any point $\mathbf{r}$, sufficiently close to curve $\mathbf{s}(l)$, can be expressed by **Frenet coordinates**. In the planar case, the Frenet coordinates consist of two variables, spanning a 2-dimensional curvilinear coordinate system.

In Frenet coordinates point $\mathbf{r}$ is represented by

$$\mathbf{r}^F = \begin{bmatrix} l_r & d_{p,r} \end{bmatrix}^T, \tag{36}$$

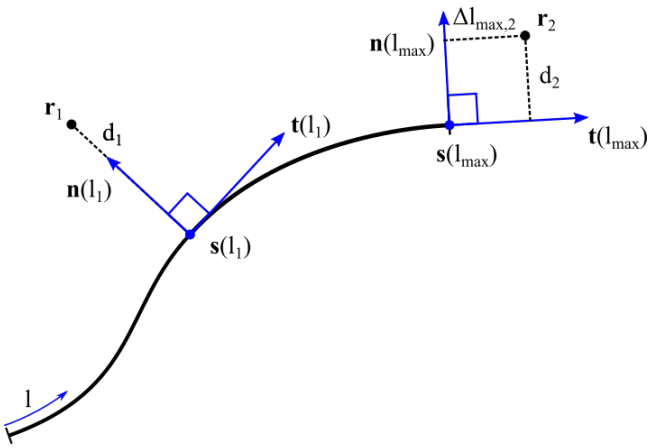

Figure 4: Projection of a point $\mathbf{r}$ on a curve which is defined within parameter interval domain $l \in I = [l_{min}, l_{max}]$. For point $\mathbf{r}_1$ a perpendicular projection can be found and Δl_p evaluates to zero (or below the search threshold). Point $\mathbf{r}_2$ is located behind parameter interval domain I and can't be projected onto the curve. Thus, $\Delta l_p = \Delta l_{max,2}$ is larger than zero.

where l_r indicates the arc-length and $d_{p,r}$ denotes the deviation from the curve. Both values are also referred to as longitudinal displacement (l) and lateral displacement (d).

If deviation d is within radius of the osculating circle, point $\mathbf{r}$ can be expressed by a unique pair of Frenet coordinates. When curve $\mathbf{s}$ includes several stretches with bends and arcs, a single Cartesian point could potentially be represented by multiple different Frenet coordinates. However, which of these Frenet coordinates are used depends on the specific application. In the Corridor library all possible perpendicular projections of point $\mathbf{r}$ on $\mathbf{s}$ are determined but only the one with the smallest deviation is used. In the same way, any other distance norm or selection algorithm can be applied.

Utilizing the rotation matrix ${}^{C}\mathbf{R}_{p}^{F}$, c.f. (31), the correlation between an arbitrary Frenet coordinate and its Cartesian expression is given by:

$$\mathbf{r}^{C} = \mathbf{s}(l_p) + {}^{C}\mathbf{R}_{p}^{F}\ \mathbf{d}_{p,r}^{F} \tag{37}$$

where $\mathbf{d}_{p,r}^{F}$ denotes the relative vector between base point on the curve and point $\mathbf{r}$ in Frenet coordinates. It consists of the arc-length difference between point $\mathbf{r}$ and base point $\mathbf{s}(l_p)$, as well as the deviation from the curve:

$$\mathbf{d}_{p,r}^{F} = [\Delta l_{p,r} \;\; d_{p,r}]^{T} \qquad \text{with} \quad \Delta l_{p,r} = l_r - l_p. \tag{38}$$

If arc-length l_r is within the parameter interval domain I, a perpendicular projection of point $\mathbf{r}_{\mathbf{x}}^{C}$ on curve exists. In this case, $\Delta l_{p,r}$ evaluates to zero. However, if l_r is outside interval I, the delta arc-length $\Delta l_{p,r}$ determines the offset

between the closest point on the curve and the projection on the tangent vector at this point. An example is given in Figure 4 (point $\mathbf{r}_2$).

3.2.1 Projection point

Given a Cartesian coordinate $\mathbf{r}^C$ the arc-length l_p, representing its projection on curve $\mathbf{s}$, can be determined by minimizing following distance function $D(\mathbf{s}(l), \mathbf{r})$:

$$D(\mathbf{s}(l), \mathbf{r}) = \|\mathbf{r} - \mathbf{s}(l)\|^2 = ||\mathbf{d}(l)||^2 = \mathbf{d}(l)^T \mathbf{d}(l) \tag{39}$$

Extreme points of the distance function can be found by evaluating its first derivative:

$$\begin{aligned} \frac{d}{dl} D(\mathbf{s}(l), \mathbf{r}) &= \frac{d}{dl} \mathbf{d}(l)^T \mathbf{d}(l) \\ &= \left(-\frac{d}{dl} \mathbf{s}(l) \right)^T (\mathbf{r} - \mathbf{s}(l)) + (\mathbf{r} - \mathbf{s}(l))^T \left(-\frac{d}{dl} \mathbf{s}(l) \right) \\ &= -2\, \mathbf{t}(l)^T \ (\mathbf{r} - \mathbf{s}(l)) \\ &= -2\, \mathbf{t}(l)^T \ \mathbf{d}(l) \end{aligned} \tag{40}$$

which is proportional to the perpendicular projection of the distance vector $\mathbf{d}(l)$ onto the tangent $\mathbf{t}(l)$. Without loss of generality we can drop factor -2, which yields to the tangential projection function:

$$\delta(l, \mathbf{r}) = \mathbf{t}(l)^T \mathbf{d}(l) \tag{41}$$

Finding arc-length l_p of the projection of an arbitrary position $\mathbf{r}$ around the curve requires to find the zero points (roots) of the tangential projection

$$\delta(l_p, \mathbf{r}) \stackrel{!}{=} 0 \tag{42}$$

which correlates to extreme points of distance function (39). Due to the cubic characteristic of distance vector $\mathbf{d}(l)$ and the quadratic function for tangent $\mathbf{t}(l)$ it produces a quintic equation of the form:

$$\alpha_5 l_p^5 + \alpha_4 l_p^4 + \alpha_3 l_p^3 + \alpha_2 l_p^2 + \alpha_1 l_p + \alpha_0 \stackrel{!}{=} 0 \tag{43}$$

Unfortunately, in case of arbitrary coefficients (like the one at hand), the *Abel-Ruffini theorem* states that no general solution for such polynomial equations of degree five or higher can be found [Ramond, 2020]. Consequently, we need to utilize a numerical solution for finding the projection point.

3.2.2 Numerical solution for projection point search

Assuming curve $\mathbf{s}(l)$ is a cubic interpolation of $n+1$ support points $\mathbf{p}_i$ for $i = 0, \ldots, n$.

First step in finding arc-length l_p is to determine all spline segments which are suspected to hold a perpendicular projection of point $\mathbf{r}$. This is achieved

by evaluating equation (41) at all support points. The resulting values of the tangential projection function is then evaluated using the *Intermediate value theorem*[2]. Whenever the signs of value $\delta(l_i)$ and $\delta(l_{i+1})$ are different, segment i contains a solution for equation (42). Thus, this segment must contain a perpendicular projection point within its arc-length interval $l_p \in [l_i, l_{i+1}]$.

For each of the found spline segments $\mathbf{q}_i = [\mathbf{p}_i, \mathbf{p}_{i+1}]$, the relative arc-length $\Delta l_p = l_p - l_i$ must be determined.

A first possibility to solve the root-finding problem of equation (42) is the **Newton-Raphson method**. Starting from an initial guess $\Delta l_{p,0}$, it iteratively approximates the root by evaluating the recursive function:

$$\Delta l_{p,k+1} = \Delta l_{p,k} - \frac{\delta(\Delta l_{p,k})}{\delta'(\Delta l_{p,k})} \tag{44}$$

Since the spline segment is known, either the start or end arc-length could be used as initial guess: $l_{p,0} \in \{l_i,\, l_{i+1}\}$. The derivative of the tangential projection function can be calculated quite efficiently, following:

$$\begin{aligned} \delta'(l, \mathbf{r}) &= \frac{d}{dl}\left(\mathbf{t}(l)^T \mathbf{d}(l)\right) \\ &= \kappa(l)\, \mathbf{n}(l)^T \mathbf{d}(l) + \mathbf{t}(l)^T(-\mathbf{t}(l)) \\ &= \kappa(l)\, \mathbf{n}(l)^T \mathbf{d}(l) - 1 \end{aligned} \tag{45}$$

As visualized in Figure 5, arc-length Δl_p is found when the difference between two iterations is either zero, below a predefined threshold or converged to one of either interval bounds of the segment. In the best case, this method has a quadratic convergence rate, but it can collapse to a merely linear rate. Moreover, there is a chance that this method doesn't converge at all. Especially if the root is close to areas with high curvature and/or high curvature change rate.

An alternative solution is the **bisection method** (or binary search algorithm). This method repeatedly bisects the defined interval and selects the subinterval in which function (41) changes sign. This subsection must contain the root and is therefore used as new interval. The bisection method is very simple to implement and robust in finding the root, but the convergence rate is rather slow.

To achieve faster convergence rates a combination of bisection method, secant method and inverse quadratic interpolation can be used as well. This is known as **Brent's method**, which combines the reliability of bisection but can converge as quickly as the less reliable methods [Brent, 2013]. Nowadays it is

[2]If a continuous function has values of opposite signs inside an interval, then it has a root in that interval.

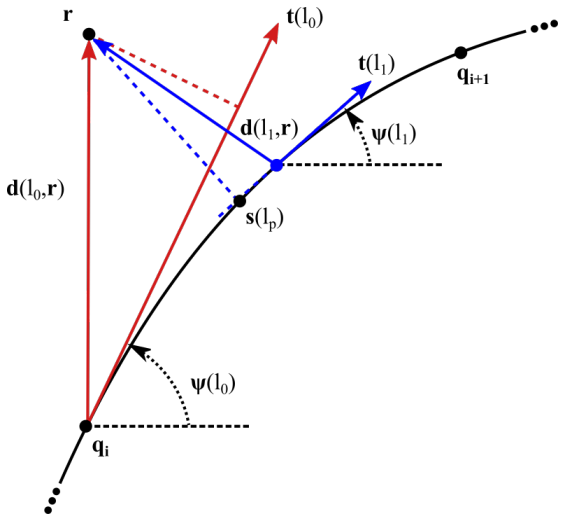

Figure 5: Visualization of the Newton-Raphson method for approximation of the arc-length of the perpendicular projection of point $\mathbf{r}$ on curve $\mathbf{s}(l)$.

used as the de-facto standard for root-finding of single-variable functions, applied in Python's Numpy library and MATLAB.

All three methods are implemented within in the Corridor library and can be used to find the arc-length of a projected point. Default method is **Brent's Method**.

4 Kinematic state transformation

Given an arbitrary state vector, defined in Cartesian coordinates, this section introduces the necessary kinematic transformation functions to express the state in Frenet coordinates of curve $\mathbf{s}$. In the Corridor library a state vector consist of position and velocity information, but could be extended by acceleration data if accessible.

Assuming Cartesian representation of position and velocity vector, the kinematic transformation fulfills a mapping from the Cartesian frame to Frenet frame:

$$\mathbf{x}^F = {}^F\mathbf{f}^C(\mathbf{x}^C) \qquad \text{with} \quad {}^F\mathbf{f}^C : \mathbb{R}^4 \to \mathbb{R}^4 \tag{46}$$

Where the components of each vector are defined as follows:

$$\mathbf{x}^C = \begin{bmatrix} \mathbf{r}_\mathbf{x}^C \\ \mathbf{v}_\mathbf{x}^C \end{bmatrix} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \qquad \mathbf{x}^F = \begin{bmatrix} \mathbf{r}_\mathbf{x}^F \\ \mathbf{v}_\mathbf{x}^F \end{bmatrix} = \begin{bmatrix} l_r \\ d_{p,r} \\ v_l \\ v_d \end{bmatrix} \tag{47}$$

4.1 Position transformation

The transformation of Cartesian position $\mathbf{r}_{\mathbf{x}}^{C}$ into Frenet coordinates can be determined by inverting equation (37) and solving for relative vector $\mathbf{d}_{p,r}^{F}$:

$$\mathbf{d}_{p,r}^{F} = {}^{C}\mathbf{R}_{p}^{F}\left(\mathbf{r}_{\mathbf{x}}^{C} - \mathbf{s}(l_p)\right) = {}^{C}\mathbf{R}_{p}^{F}\,\mathbf{d}_{p,r}^{C} \tag{48}$$

By utilizing the identity from equation (30), the rotation of relative vector from Cartesian to Frenet frame can be expressed by projecting it to orthogonal vectors $\mathbf{t}(l_p)$ and $\mathbf{n}(l_p)$.

$$\mathbf{d}_{p,r}^{F} = {}^{C}\mathbf{R}_{p}^{F}\,\mathbf{d}_{p,r}^{C} = \begin{bmatrix} \mathbf{t}_{p}^{T} \\ \mathbf{n}_{p}^{T} \end{bmatrix} \mathbf{d}_{p,r}^{C} \tag{49}$$

Decomposing the right-hand side and separating the Frenet position leads to wanted transformation function:

$$\mathbf{r}_{\mathbf{x}}^{F} = \begin{bmatrix} \mathbf{t}_{p}^{T} \\ \mathbf{n}_{p}^{T} \end{bmatrix} \left(\mathbf{r}_{\mathbf{x}}^{C} - \mathbf{s}(l_p)\right) + \begin{bmatrix} l_p \\ 0 \end{bmatrix} \tag{50}$$

4.2 Velocity transformation

The velocity in Frenet coordinates $\mathbf{v}_{\mathbf{x}}^{F}$ can be obtained from the time derivative of the position (50):

$$\mathbf{v}_{\mathbf{x}}^{F} = \frac{d}{dt}\mathbf{d}_{p,r}^{F} + \begin{bmatrix} v_p \\ 0 \end{bmatrix} \tag{51}$$

It consists of the time derivative of the relative position $\mathbf{v}_{p,r}^{F} = \frac{d}{dt}\mathbf{d}_{p,r}^{F}$ and the absolute velocity of the projection point along the curve $v_p = \frac{d}{dt}l_p$.
Incorporating the Frenet formulas (25) and their time derivatives (32)-(35) leads to following expression of the Frenet velocity:

$$\mathbf{v}_{\mathbf{x}}^{F} = \underbrace{\begin{bmatrix} \mathbf{t}_{p}^{T} \\ \mathbf{n}_{p}^{T} \end{bmatrix} \mathbf{v}_{\mathbf{x}}^{C}}_{\text{abs. vel in FF}} + \underbrace{\dot{\psi}_p \begin{bmatrix} \mathbf{n}_{p}^{T} \\ -\mathbf{t}_{p}^{T} \end{bmatrix} \left(\mathbf{r}_{\mathbf{x}}^{C} - \mathbf{s}(l_p)\right)}_{\text{rot. vel of FF}} \tag{52}$$

4.3 Velocity of projection point

Besides the Cartesian state vector $\mathbf{x}^{C}$ and the cubic spline $\mathbf{s}(l)$, the introduced transformation functions depend also on the velocity $v_p(t)$ of the projection point at l_p.

While the position $s(l_p)$ is derived from the projection of the Cartesion position $\mathbf{r}_{\mathbf{x}}^{C}$ onto the curve, we can formulate different assumption about the velocity of the projection point.

In next section the following two assumptions are discussed:

A-1: Projection point is frozen at current time, leading to the assumption that the projected velocity is zero: $v_p = 0$. As a result, the transformation matrices, ${}^{C}\mathbf{R}_{p}^{F}$ and ${}^{F}\mathbf{R}_{p}^{C}$, are constant in time as well.

A-2: Projection point's motion correlates with projected velocity onto the tangent vector $\mathbf{t}_p$. This leads to absolute velocity greater than zero if the velocity direction is not perpendicular to the curve.

Both assumptions are valid and useful for certain use cases. The effect of both to the transformation functions is discussed in the next two paragraphs.

4.3.1 [A-1] Projection point is frozen at current time

Imagine that the projection point of $\mathbf{r}_\mathbf{x}^C$ onto the curve $\mathbf{s}(l)$ is frozen at projection time t_p, then the time derivative of arc-length l_p vanishes:

$$v_p(t_p) \stackrel{!}{=} 0 \tag{53}$$

Therefore, also the angular velocity, introduced in (34), evaluates to zero:

$$\dot{\psi}_p = \kappa_p \, v_p(t_p) = 0 \tag{54}$$

As a result, the relative velocity $\mathbf{v}_{p,r}^F$ is given with respect to a non-moving reference frame. Therefore, the absolute velocity in Frenet coordinates is independent of the kinematics of the Frenet Frame. Similar assumptions are also known as *"given over ground"*.

This simplifies the equation of the velocity vector in Frenet coordinates, (52), to a projection of the Cartesian velocity vector to the axes of the (non-moving) Frenet frame:

$$\mathbf{v}_\mathbf{x}^F = \begin{bmatrix} \mathbf{t}_p^T \\ \mathbf{n}_p^T \end{bmatrix} \mathbf{v}_\mathbf{x}^C \tag{55}$$

This assumption works best for developing simple motion models which utilizes information from the Frenet coordinates. Since the resulting velocity and acceleration is given relatively to a fixed Frenet frame it is also useful for comparing velocities of different objects respectively to one common curve. It is also useful for a fast approximation of time measures, like time-to-arrive and time-to-collision.

4.3.2 [A-2] Projection point motion correlates with projected velocity

The second assumption states, that the projection point's velocity v_p correlates with the projection of the velocity vector $\mathbf{v}_\mathbf{x}^C$ onto the tangent vector $\mathbf{t}_p$:

$$v_p(t_p) \stackrel{!}{=} \mathbf{t}_p^T \mathbf{v}_\mathbf{x}^C \tag{56}$$

In areas with non-zero curvature, this leads to following expression for the angular velocity $\dot{\psi}_p$:

$$\dot{\psi}_p(t_p) = \kappa_p \mathbf{t}_p^T \mathbf{v}_\mathbf{x}^C \tag{57}$$

Such non-zero velocities render a moving Frenet frame along the curve. Substituting these terms in equation (52), the velocity in Frenet coordinates is defined as:

$$\mathbf{v}_{\mathbf{x}}^{F} = \begin{bmatrix} \mathbf{t}_p^T \\ \mathbf{n}_p^T \end{bmatrix} \mathbf{v}_{\mathbf{x}}^{C} + \kappa_p \mathbf{t}_p^T \mathbf{v}_{\mathbf{x}}^{C} \begin{bmatrix} \mathbf{n}_p^T \\ -\mathbf{t}_p^T \end{bmatrix} \left(\mathbf{r}_{\mathbf{x}}^{C} - \mathbf{s}(l_p)\right). \tag{58}$$

This assumption is particular useful when applying motion models in Frenet coordinates, or whenever the propagation of the full kinematic Frenet state is needed.

5 Propagation of uncertainty

When dealing with noisy data one of the biggest challenges is to determine the resulting distribution after applying a transformation. For multi-variate normal distributions a consistent estimate of the first two moments of the transformed state distribution is sufficient: the mean μ and covariance matrix Σ.

Let $\boldsymbol{\mathcal{X}}$ be an n-dimensional random variable with mean $\boldsymbol{\mu}_{\mathbf{x}}$ and covariance $\boldsymbol{\Sigma}_{\mathbf{x}}$. A second random variable, $\boldsymbol{\mathcal{Y}}$ is related to $\boldsymbol{\mathcal{X}}$ through transformation $\mathbf{f}$:

$$\boldsymbol{\mathcal{Y}} = \mathbf{f}\left(\boldsymbol{\mathcal{X}}\right) \tag{59}$$

Assuming $\mathbf{f}(\cdot)$ is sufficiently differentiable it is possible to expand $\mathbf{f}$ about $\boldsymbol{\mu}_{\mathbf{x}}$ using multidimensional Taylor Expansion [Hendeby et al., 2007]:

$$\mathbf{f}\left(\boldsymbol{\mathcal{X}}\right) = \mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}}) + \nabla\mathbf{f}\boldsymbol{\delta}_{\boldsymbol{x}} + \frac{1}{2}\nabla^2\mathbf{f}{\boldsymbol{\delta}_{\boldsymbol{x}}}^2 + \frac{1}{3!}\nabla^3\mathbf{f}{\boldsymbol{\delta}_{\boldsymbol{x}}}^3 + \cdots \tag{60}$$

where, $\boldsymbol{\delta}_{\boldsymbol{x}} = (\boldsymbol{\mathcal{X}} - \boldsymbol{\mu}_{\mathbf{x}})$ and $\nabla^i\mathbf{f}{\boldsymbol{\delta}_{\boldsymbol{x}}}^i$ represents the ith order term of the Taylor Series. According to Julier (2002) it can be shown, that the first two moments of $\boldsymbol{\mathcal{Y}}$ are determined by:

$$\begin{aligned}
\boldsymbol{\mu}_{\mathbf{y}} &= \mathrm{E}[\boldsymbol{\mathcal{Y}}] \\
&= \mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}}) + \frac{1}{2}\nabla^2\mathbf{f}\underbrace{\mathrm{E}[(\boldsymbol{\mathcal{X}} - \boldsymbol{\mu}_{\mathbf{x}})^2]}_{\boldsymbol{\Sigma}_{\mathbf{x}}} + \frac{1}{3!}\nabla^3\mathbf{f}\mathrm{E}[(\boldsymbol{\mathcal{X}} - \boldsymbol{\mu}_{\mathbf{x}})^3] + \cdots \\
\boldsymbol{\Sigma}_{\mathbf{y}} &= \mathrm{E}[(\boldsymbol{\mathcal{Y}} - \boldsymbol{\mu}_{\mathbf{y}})(\boldsymbol{\mathcal{Y}} - \boldsymbol{\mu}_{\mathbf{y}})^T] \\
&= \nabla\mathbf{f}\boldsymbol{\Sigma}_{\mathbf{x}}(\nabla\mathbf{f})^T + \frac{1}{4}\nabla^2\mathbf{f}\mathrm{E}[{\boldsymbol{\delta}_{\boldsymbol{x}}}^3](\nabla\mathbf{f})^T + \frac{1}{2}\nabla\mathbf{f}\mathrm{E}[{\boldsymbol{\delta}_{\boldsymbol{x}}}^3](\nabla^2\mathbf{f})^T \\
&\quad + \frac{1}{2}\nabla^2\mathbf{f}\left(\mathrm{E}[{\boldsymbol{\delta}_{\boldsymbol{x}}}^4] - \mathrm{E}[{\boldsymbol{\delta}_{\boldsymbol{x}}}^2\boldsymbol{\Sigma}_{\mathbf{x}}] - \mathrm{E}[\boldsymbol{\Sigma}_{\mathbf{x}}{\boldsymbol{\delta}_{\boldsymbol{x}}}^2] + \boldsymbol{\Sigma}_{\mathbf{x}}^2\right)(\nabla^2\mathbf{f})^T \\
&\quad + \frac{1}{3!}\nabla^3\mathbf{f}\mathrm{E}[{\boldsymbol{\delta}_{\boldsymbol{x}}}^4](\nabla\mathbf{f})^T + \cdots
\end{aligned} \tag{61}$$

Under the assumption that $\boldsymbol{\mathcal{Y}}$ is modeled as a normal distribution (or a mixture of these), the Taylor Expansion is usually aborted after the first order. Sometimes the second order terms are used as well. However, approximation of the second moment requires to know the expectation of $E[\boldsymbol{\mathcal{X}}^4]$, which is rarely the case.

5.1 Linear transformation

In the most basic situation is $\mathbf{f} = \{f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\}$ a set of m functions which are linear combinations of n variables $\{x_1, x_2, \ldots, x_n\}$:

$$\mathbf{f} = \mathbf{A}\mathbf{x} \tag{62}$$

The Taylor Expansion (60) simplifies to the first two terms since all partial derivations above degree one vanish. Resulting in a transformation function which boils down to the first terms for the two moments of $\boldsymbol{\mathcal{Y}}$:

$$\begin{aligned} \boldsymbol{\mu}_\mathbf{y} &= \mathbf{A}\boldsymbol{\mu}_\mathbf{x} \\ \boldsymbol{\Sigma}_\mathbf{y} &= \mathbf{A}\boldsymbol{\Sigma}_\mathbf{x}\mathbf{A}^T \end{aligned} \tag{63}$$

5.2 Non-linear transformation

When $\mathbf{f}$ is a set of non-linear combinations of variable $\mathbf{x}$ and $\boldsymbol{\mathcal{Y}}$ is considered as normally distributed, the Taylor Expansion can be reduced to its first order.

$$\begin{aligned} \boldsymbol{\mathcal{Y}} &= \mathbf{f}(\boldsymbol{\mathcal{X}}) \\ &\approx \mathbf{f}(\boldsymbol{\mu}_\mathbf{x}) + \mathbf{J}_{\mathbf{f}_\mathbf{x}}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^T \mathbf{H}_{\mathbf{f}_\mathbf{x}}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}) \end{aligned} \tag{64}$$

In this expression, $\mathbf{J}_{\mathbf{f}_\mathbf{x}}$ represents the Jacobian matrix and $\mathbf{H}_{\mathbf{f}_\mathbf{x}}$ the Hessian matrix, both evaluated at point $\boldsymbol{\mu}_\mathbf{x}$. This approximation works well if the remaining terms are considered to be small, relating to both: the state covariance $\boldsymbol{\Sigma}_\mathbf{x}$ and the degree of non-linearity of $\mathbf{f}$. Hendeby et al. (2007) provide following rule of thumb when this approximation can be applied:

- transformation function $\mathbf{f}$ is almost linear around $\boldsymbol{\mu}_\mathbf{x}$
- signal-to-noise ratio is high, in which case the covariance can be considered sufficiently small.

If the approximation is applicable, the first two moments of $\boldsymbol{\mathcal{Y}}$ are defined as:

$$\begin{aligned} \boldsymbol{\mu}_\mathbf{y} &\approx \mathbf{f}(\boldsymbol{\mu}_\mathbf{x}) \\ \boldsymbol{\Sigma}_\mathbf{y} &\approx \mathbf{J}_{\mathbf{f}_\mathbf{x}} \boldsymbol{\Sigma}_\mathbf{x} \mathbf{J}_{\mathbf{f}_\mathbf{x}}^T \end{aligned} \tag{65}$$

5.3 Unscented Transformation

The Unscented Transformation (UT) operates on the premise that it is easier to approximate a normal distribution than it is to approximate an arbitrary non-linear function. Instead of linearizing using Jacobian and Hessian matrices, the UT is using a deterministic sampling approach to capture the mean and covariance matrix of $\boldsymbol{\mathcal{X}}$ by selecting a set of $2n + 1$ sigma points [Julier, 2002]. Consequently, the UT is used within the Corridor library to perform all non-linear transformation functions. The implementation is kept generic, so normal

distributions of any dimension and arbitrary transformation functions can be applied. This involves transformations which contain conversions from and to angular expressions as well. The implementation constraints generated sigma point automatically to the interval $[-\pi, \pi]$.

Each sigma point $\boldsymbol{\mathcal{S}}_{\mathbf{x},i} = \{\bar{\mathbf{x}}_i, \omega_i\}$ consists of a sigma vector $\bar{\mathbf{x}}_i$ and its corresponding weight ω_i. The sigma points can be chosen in a number of ways, but must satisfy:

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{x}} &= \sum_{i=0}^{2n} \omega_i \, \bar{\mathbf{x}}_i \\ \boldsymbol{\Sigma}_{\mathbf{x}} &= \sum_{i=0}^{2n} \omega_i \, (\bar{\mathbf{x}}_i - \boldsymbol{\mu}_{\mathbf{x}}) \, (\bar{\mathbf{x}}_i - \boldsymbol{\mu}_{\mathbf{x}})^T \end{aligned} \tag{66}$$

The sigma points are passed through the non-linear transformation, yielding a transformed set of points. The new mean and covariance estimate is then computed based on the transformed sigma vectors $\bar{\mathbf{y}}_i$.

$$\begin{aligned} \bar{\mathbf{y}}_i &= \mathbf{f}(\bar{\mathbf{x}}_i) \\ \boldsymbol{\mu}_{\mathbf{y}} &= \sum_{i=0}^{2n} \omega_i \, \bar{\mathbf{y}}_i \\ \boldsymbol{\Sigma}_{\mathbf{y}} &= \sum_{i=0}^{2n} \omega_i \, (\bar{\mathbf{y}}_i - \boldsymbol{\mu}_{\mathbf{y}}) \, (\bar{\mathbf{y}}_i - \boldsymbol{\mu}_{\mathbf{y}})^T \end{aligned} \tag{67}$$

5.3.1 Choosing sigma points

There are numerous possibilities mentioned in the literature how to chose sigma points [Menegaz et al., 2015].

In the Corridor library the selection algorithm presented in Wan et al. (2000) is used. It performs well with a variety of problems and optimizes the trade-off between performance and accuracy. The algorithm of Wan and Merwe is hereby a slight adaption of the *Scaled Unscented Transform* by Julier (2002). It is based on three parameters to control how the sigma points are distributed and weighted: α, β and κ. The parameters are chosen depending on the problem at hand. α determines the spread of the sigma points around $\boldsymbol{\mu}_{\mathbf{x}}$ and is usually set to a small positive value (e.g. $1e^{-3}$ as default). κ is a secondary scaling parameter which is usually set to 0, whereas β is used to incorporate prior knowledge of the initial distribution $\boldsymbol{\mathcal{X}}$. For Gaussian distributions, $\beta = 2$ is considered to be optimal [Wan et al., 2000].

The $2n + 1$ sigma vectors and their corresponding weights can be calculated

using the scaling parameter $\lambda = \alpha^2(n + \kappa) - n$:

$$\begin{aligned}
\bar{\mathbf{x}}_0 &= \boldsymbol{\mu}_{\mathbf{x}} \\
\bar{\mathbf{x}}_i &= \begin{cases} \boldsymbol{\mu}_{\mathbf{x}} + \left[\sqrt{(n+\lambda)\boldsymbol{\Sigma}_{\mathbf{x}}}\right]_i & i = 1, \ldots, n \\ \boldsymbol{\mu}_{\mathbf{x}} - \left[\sqrt{(n+\lambda)\boldsymbol{\Sigma}_{\mathbf{x}}}\right]_{i-n} & i = n+1, \ldots, 2n \end{cases} \\
\omega_0^{(m)} &= \frac{\lambda}{n+\lambda} \\
\omega_0^{(c)} &= \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\
\omega_i^{(m)} &= \omega_i^{(c)} = \frac{1}{2(n+\lambda)} \qquad i = 1, \ldots, 2n
\end{aligned} \tag{68}$$

The subscription at $\left[\sqrt{(n+\lambda)\boldsymbol{\Sigma}_{\mathbf{x}}}\right]_i$ denotes the ith row vector of the matrix square root. Noteworthy is that the weight for mean $\omega_0^{(m)}$ and covariance $\omega_0^{(c)}$ is computed differently in this algorithm. The remaining weights are identical. Idea behind is to compensate a potential bias in the transformed covariance matrix estimation in case of strong non-linearities in transformation function $\mathbf{f}(\bar{\mathbf{x}}_i)$.

6 State transformation under uncertainty

The Cartesian state is almost always estimated based on noisy sensor information, which leads to the inevitable challenge of propagating measurement error and system uncertainty. To represent the uncertainty about the state vector, it is usually defined as random variable with normal distribution, or more general, as Gaussian mixture model. Within the Corridor library the Cartesian state vector is represented by a 4-dimensional normal distribution model:

$$\boldsymbol{\mathcal{X}}^C \sim \mathcal{N}_4(\boldsymbol{\mu}_{\mathbf{x}}^C, \boldsymbol{\Sigma}_{\mathbf{x}}^C) \tag{69}$$

The expectation vector $\boldsymbol{\mu}_{\mathbf{x}}^C$ contains the position and velocity with respect to the Cartesian frame:

$$\boldsymbol{\mu}_{\mathbf{x}}^C = \begin{bmatrix} \mu_x & \mu_y & \mu_{v_x} & \mu_{v_y} \end{bmatrix}^T \qquad \boldsymbol{\mu}_{\mathbf{x}}^F = \begin{bmatrix} \mu_l & \mu_d & \mu_{v_l} & \mu_{v_d} \end{bmatrix}^T \tag{70}$$

For this specific state vector the most general definition of the uncertainty information is a (4×4) covariance matrix where position and velocity are correlated to each other. For simplicity reasons we only denote the upper triangle of the symmetric covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}}^C$:

$$\boldsymbol{\Sigma}_{\mathbf{x}}^C = \begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 & \sigma_{xv_x}^2 & \sigma_{xv_y}^2 \\ & \sigma_{yy}^2 & \sigma_{yv_x}^2 & \sigma_{yv_y}^2 \\ & & \sigma_{v_xv_x}^2 & \sigma_{v_xv_y}^2 \\ & \textit{symm.} & & \sigma_{v_yv_y}^2 \end{bmatrix} \tag{71}$$

The Frenet state can be obtained by converting the random Cartesian state vector into the Frenet frame of a corridor:

$$\begin{aligned} \boldsymbol{\mathcal{X}}^F &= {}^F\mathbf{f}^C(\boldsymbol{\mathcal{X}}^C) \\ &\sim \mathcal{N}_4(\boldsymbol{\mu}_{\mathbf{x}}^F, \boldsymbol{\Sigma}_{\mathbf{x}}^F) \end{aligned} \tag{72}$$

6.1 Uncertainty propagation for assumption A-1

Assumption A-1 leads to a linear transformation for converting the Cartesian state vector into the Frenet frame, c.f. (55). Substituting the linear transformation functions into equation (46) results in:

$$\begin{aligned} \mathbf{x}^F &= {}^F\mathbf{f}_{A1}^C(\mathbf{x}^C, \mathbf{v}_{\mathbf{x}}^C) \\ &= \underbrace{\begin{bmatrix} {}^F\mathbf{R}_p^C & \mathbf{0} \\ \mathbf{0} & {}^F\mathbf{R}_p^C \end{bmatrix}}_{{}^F\bar{\mathbf{R}}_p^C} \begin{bmatrix} \mathbf{r}_{\mathbf{x}}^C - \mathbf{s}_p^C \\ \mathbf{v}_{\mathbf{x}}^C \end{bmatrix} + \begin{bmatrix} \mathbf{r}_p^F \\ 0 \end{bmatrix} \end{aligned} \tag{73}$$

where $\mathbf{s}_p^C$ denotes the base point of the Frenet frame and $\mathbf{r}_p^F = [l_p\ 0]^T$ the base point's arc-length along the curve.

Reflecting the uncertainty of the Cartesian state vector results in following expressions for the mean and covariance matrix of the Frenet state:

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{x}}^F &= {}^F\mathbf{f}_{A1}^C(\boldsymbol{\mu}_{\mathbf{x}}^C) \\ \boldsymbol{\Sigma}_{\mathbf{x}}^F &= {}^F\bar{\mathbf{R}}_p^C \boldsymbol{\Sigma}_{\mathbf{x}}^C ({}^F\bar{\mathbf{R}}_p^C)^T \end{aligned} \tag{74}$$

Due to the non-moving Frenet frame the transformation boils down to a translation induced by the arc-length of the projection point and a rotation, defined by the tangential and normal vector at this point.

6.2 Uncertainty propagation for assumption A-2

Assuming a moving Frenet frame with velocity $v_p = \mathbf{t}_p^T \mathbf{v}_{\mathbf{x}}^C$, the conversion from a Cartesian state to the Frenet frame yields:

$$\mathbf{x}^F = {}^F\mathbf{f}_{A2}^C(\mathbf{x}^C, \mathbf{v}_{\mathbf{x}}^C) = \begin{bmatrix} \mathbf{t}_p^T \left[\mathbf{r}_{\mathbf{x}}^C - \mathbf{s}_p^C\right] + l_p \\ \mathbf{n}_p^T \left[\mathbf{r}_{\mathbf{x}}^C - \mathbf{s}_p^C\right] \\ \mathbf{t}_p^T \mathbf{v}_{\mathbf{x}}^C + \kappa_p \left(\mathbf{t}_p^T \mathbf{v}_{\mathbf{x}}^C\right) \ \mathbf{n}_p^T \left[\mathbf{r}_{\mathbf{x}}^C - \mathbf{s}_p^C\right] \\ \mathbf{n}_p^T \mathbf{v}_{\mathbf{x}}^C - \kappa_p \left(\mathbf{t}_p^T \mathbf{v}_{\mathbf{x}}^C\right) \ \mathbf{t}_p^T \left[\mathbf{r}_{\mathbf{x}}^C - \mathbf{s}_p^C\right] \end{bmatrix} \tag{75}$$

The position transformation remains linear, while the frenet velocity is a non-linear function of Cartesian position and velocity.

For the error propagation either the linearized transformation or the Unscented Transformation can be used. The linearized transformation, defined in section 5.2, utilized the Jacobean of function ${}^F\mathbf{f}_{A2}^C$ to obtain the covariance matrix of the converted state:

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{x}}^F &= {}^F\mathbf{f}_{A2}^C(\boldsymbol{\mu}_{\mathbf{x}}^C) \\ \boldsymbol{\Sigma}_{\mathbf{x}}^F &= \mathbf{J}_{\mathbf{f}_{A2}} \boldsymbol{\Sigma}_{\mathbf{x}}^C (\mathbf{J}_{\mathbf{f}_{A2}})^T \end{aligned} \tag{76}$$

The Unscented Transformation, defined in section 5.3, converts the selected sigma points $\bar{\mathbf{x}}^C$ into the Frenet frame. The resulting transformed sigma points $\bar{\mathbf{x}}^F$ are then used to determine the mean and covariance of the Frenet state:

$$\begin{aligned}\bar{\mathbf{x}}^F &= {}^F\mathbf{f}_{A2}^C(\bar{\mathbf{x}}^C)\\ \boldsymbol{\mu}_{\mathbf{x}}^F &= \sum_{i=0}^{2n} \omega_i\, \bar{\mathbf{x}}^F\\ \boldsymbol{\Sigma}_{\mathbf{x}}^F &= \sum_{i=0}^{2n} \omega_i \left(\bar{\mathbf{x}}^F - \boldsymbol{\mu}_{\mathbf{x}}^F\right)\left(\bar{\mathbf{x}}^F - \boldsymbol{\mu}_{\mathbf{x}}^F\right)^T\end{aligned} \tag{77}$$

6.3 Evaluation

In this section the evaluation of both state transformation assumptions are discussed. Each state conversion is approximated by both, linearized and unscented transformation, which leads to a total of four test cases:

1. Linearized transformation, assuming a stationary Frenet frame (A-1)
2. Linearized transformation, assuming a moving Frenet frame (A-2)
3. Unscented transformation, assuming a stationary Frenet frame (A-1)
4. Unscented transformation, assuming a moving Frenet frame (A-2)

The evaluation is conducted by a Python script accessing the Corridor library. Test environment is modeled by a single corridor with a constant width of 4 meters. Its reference line is defined by three points with a distance of seven meters from each other along the x-axis. The initial y-coordinate for all nodes are set to zero. To determine the influence of the reference line's curvature, the middle point is moved incrementally by $\Delta y \in [0, 7]$ meters. The resulting curvature at the middle point ranges from $\kappa_{start} = 0.0\frac{1}{\mathrm{m}}$ to $\kappa_{end} = 0.2\frac{1}{\mathrm{m}}$, which reflects occurring curvatures in most urban areas. For all test cases, the Cartesian state is positioned one meter above the middle point and modeled as normal distributed with following first two moments:

$$\boldsymbol{\mu}_{\mathbf{x}}^C = \begin{bmatrix} 7 & \Delta y & 5 & -2 \end{bmatrix}^T \qquad \boldsymbol{\Sigma}_{\mathbf{x}}^F = \begin{bmatrix} 0.7 & 0.3 & 0 & 0 \\ 0.3 & 0.5 & 0 & 0 \\ 0 & 0 & 0.7 & 0.2 \\ 0 & 0 & 0.2 & 0.8 \end{bmatrix} \tag{78}$$

The evaluation is performed by comparing the converted state distributions with ground truth data obtained from a Monte Carlo transformation with 5000 sample points. The approximation quality of each transformation is determined by following two metrics:

- Euclidean distance between mean state vectors:

$$d_{eucl} = \sqrt{(\boldsymbol{\mu}_{\mathbf{x}}^F - \boldsymbol{\mu}_{GT}^F)^T(\boldsymbol{\mu}_{\mathbf{x}}^F - \boldsymbol{\mu}_{GT}^F)}$$

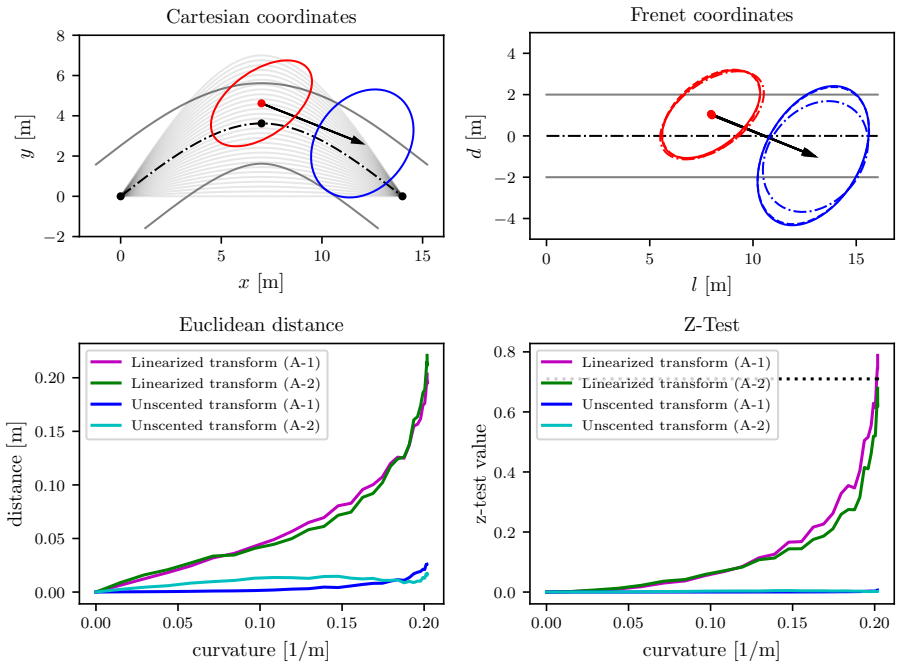

Figure 6: Evaluation of the approximation error of the different state transformations, determined at different corridor curvatures in the range of $\kappa \in [0, 0.2]$ m^{-1}. The upper left picture visualizes the corridor and state in the Cartesian frame. The red ellipse represents the position uncertainty of three standard deviations (99.7 %), the black arrow represents the velocity mean vector and the blue ellipse the velocity uncertainty of three standard deviations. The gray area visualizes all reference lines of the generated corridors. The upper right picture displays the transformed state in the Frenet frame of the corridor. Similar to the Cartesian coordinates, the red and blue ellipses represent the uncertainty of the position and velocity. The uncertainty ellipses of the ground truth is shown as solid lines, the linearized transformation as dashed-dot line. The unscented transformation is represented by a dashed line. The evaluation of both state transformations compared to the ground truth is shown in the lower two diagrams. A state transformation approximation is rejected when the z-test value is above 0.71 (dashed line in the lower right diagram).

- Z-test, which compares the transformed distribution against ground truth by testing two hypotheses:

$$H_0 : \boldsymbol{\mu}_{\mathbf{x}}^F = \boldsymbol{\mu}_{GT}^F \qquad \text{versus} \qquad H_1 : \boldsymbol{\mu}_{\mathbf{x}}^F \neq \boldsymbol{\mu}_{GT}^F$$

using the z-test statistic:

$$Z = \left(\boldsymbol{\mu}_{\mathbf{x}}^{F} - \boldsymbol{\mu}_{GT}^{F}\right)^{T} \left(\frac{n_{GT}\, \boldsymbol{\Sigma}_{\mathbf{x}}^{F} + n_x \boldsymbol{\Sigma}_{GT}^{F}}{n_x\, n_{GT}} \right)^{-1} \left(\boldsymbol{\mu}_{\mathbf{x}}^{F} - \boldsymbol{\mu}_{GT}^{F}\right)$$

Here, n_x and n_{GT} denote the sample size. In the special case $\boldsymbol{\Sigma}_{\mathbf{x}}^{F} = \boldsymbol{\Sigma}_{GT}^{F}$ the z-test value corresponds to a scaled *Mahalanobis-distance*.
For the test case at hand, the sample size of the ground truth is set to $n_{GT} = 5000$, while the sample size of the unscented transportation is $n_{x,ut} = 2n+1 = 9$. The same size is used for the linearized transformation, which allows a direct comparison of both Z-test values. Hypothesis H_0 is rejected with significance $\alpha = 95\%$ if $Z \geq \chi^2_{\alpha}(4) = 0.71$.

As shown in Figure 6 both transformations yield good results when the corridor is almost straight. However, with increasing curvature the approximation error of the linearized transformation becomes more and more significant until hypothesis H_0 needs to be rejected. The unscented transformation, on the other hand, displays a good approximation over the complete curvature range. This is mainly caused by a better involvement of curve $\mathbf{s}$ and the projected arc-length. The linearized transformation converts the Cartesian state by using one Frenet frame at the projection of the mean state vector. The unscented transformation, on the other hand, determines one Frenet frame for each sigma point; similar to the Monte-Carlo transformation. This improved performance comes, of course, at a higher computational cost.

It is worth mentioning, that the performance of the unscented transformation is only achieved by altering the spread of the sigma points. In literature, like Wan et al., 2000, it is recommended to set $\alpha = 1e^{-3}$. This value, however, lead to an unstable sum of the transformed mean and covariance matrix. By increasing α to $\alpha = 1.0$ the approximation error decreases and came to an optimum as shown in the evaluation.

The choice which state transformation should be applied, boils down to a trade-off between accuracy and performance. In areas with lower expected curvatures, like highway scenarios, the approximation accuracy of the linearized state transformation might be sufficient. Additionally, the distances to other objects are normally considerably larger with a potentially higher relative velocities than in other traffic scenarios. Therefore, the additional approximation error might not as significant as in urban areas. For traffic scenarios with small distances and higher curvature the unscented transformation will lead to significantly better results.

7 Semantic relationships

Semantic relationships are associations which exists between two or more entities. They provide a qualitative (semantic) description about the relation of

these entities, which enables a more abstract description of the available information. Such abstractions can be used to formulate generic interpretation and inference algorithms to describe arbitrary traffic situations.

Within the Corridor library, semantic relationships are build between an observed object, like a vehicle or pedestrian, and a corridor. Each relationship consists of one or more semantic labels which are evaluated to determine its existence confidence. Two types of semantic relationships are implemented which will provide answers to the following questions:

1. Located-on relation: Is an object located on a specific corridor? How much of the object's area is located on the corridor?

2. Moves-along relation: Is the object in motion or stationary? Is it moving along or perpendicular to the corridor?

Information about an object is usually derived from noisy sensor data, therefore two kinds of uncertainty need to be considered when formulating semantic relations: attribute and structure uncertainty. The first type is due to sensor noise, the second reflects the uncertainty of associations. While attribute uncertainty is generally modeled by (multivariate) random variable, structural uncertainty can be expressed by the confidence value, $\mathcal{C}(e_i, e_j, r_k) \in [0, 1]$, that relation r_k between two entities e_i and e_j exists.

Deriving the confidence value of a specific semantic relation from a continuous state space can be a challenging task. The solution applied in the Corridor library follows the proposal of [Petrich et al., 2014]. Here, the quantity of an existence confidence is determined as function of one or more semantic label confidences:

$$\mathcal{C}(e_i, e_j, r_k) = \mathbf{f}(\mathcal{C}_A(f_{ij}, g_A), \ldots, \mathcal{C}_Z(f_{ij}, g_Z)) \tag{79}$$

Let feature f_{ij} represent the quantitative relation between entity e_i and e_j, which can be expresses by a normally distributed mapping of entity features:

$$\begin{aligned} f_{ij} &= \mathbf{f}(e_i, e_j) : \mathbb{R}^n \to \mathbb{R} \\ &\sim \mathcal{N}(\mu_f, \sigma_f^2), \end{aligned} \tag{80}$$

The general form its *probability density function* (PDF) can be modeled as standard normal distribution $\phi(x)$ stretched by the standard deviation σ_f and translated by mean μ_f:

$$p_f(x) = \frac{1}{\sigma_f}\, \phi\left(\frac{x - \mu_f}{\sigma_f}\right) \tag{81}$$

The association function g_L represents the semantic label L in the definition area of feature f_{ij}. Then, the confidence value of semantic label $\mathcal{C}_L$ can be obtained by integrating the product of association function g_L and PDF $p_f(x)$:

$$\mathcal{C}_L(f_{ij}, g_L) = \int_{-\infty}^{\infty} g_L(\xi)\, p_f(\xi)\, d\xi \tag{82}$$

By formulating certain conditions towards the association functions, this integral can be simplified and solved rather efficiently.

Let $g_\alpha(x)$ be a linear association function defined within interval I_α

$$g_\alpha(x) = m\,x + b \qquad x \in I_\alpha = [u, v], \tag{83}$$

then, integral (82) can be determined by:

$$\mathcal{C}_\alpha(f, g_\alpha) = \int_u^v (m\,\xi + b)\ \frac{1}{\sigma_f}\,\phi\left(\frac{\xi - \mu_f}{\sigma_f}\right) d\xi \tag{84}$$

Substituting $\tau = \frac{\xi - \mu}{\sigma}$ and utilizing the *cumulative distribution function* (CDF)

$$\Phi(x) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{85}$$

leads to following expression for the confidence value:

$$\begin{aligned}\mathcal{C}_\alpha(f, g_\alpha) = (m\,\mu_f + b)\left[\Phi\left(\frac{v - \mu_f}{\sigma_f}\right) - \Phi\left(\frac{u - \mu_f}{\sigma_f}\right)\right] \\ - m\,\sigma_f\left[\phi\left(\frac{v - \mu_f}{\sigma_f}\right) - \phi\left(\frac{u - \mu_f}{\sigma_f}\right)\right]\end{aligned} \tag{86}$$

Any arbitrary shaped association function can be expressed or approximated by a piece-wise defined linear function. The final confidence value is then determined by the sum over all definition intervals of complete association function. The majority of association functions are either step-, rectangular- or trapezoidal-functions.

The closer the confidence value reaches 1, the more likely or plausible is the semantic label and therefor the existence of relationship r_k. For example, a confidence value of $\mathcal{C}_A = 1$ characterizes the semantic relationship, which is based on label A, as *true* or *reliable*. On the other hand, nonexistent and unknown semantic relations are identified by a confidence of $\mathcal{C} = 0$ (open-world assumption).

Figure 7 provides a generic example of a semantic relation with two associations function and the PDF of relative feature f_{ij}.

7.1 Located-on relation

The confidence value about the *located-on* relation is defined by the product of two semantic labels:

$$\mathcal{C}_{locOn}(o_i, c_j, r_{locOn}) = \mathcal{C}_{lat}(f_{lat}, g_{lat})\,\mathcal{C}_{long}(f_{long}, g_{long}) \tag{87}$$

The first factor denotes the assignment confidence for the *laterally-matched* label, the second represents the assignment confidence of the *longitudinally-matched* label. Both labels are designed to be greater than zero when the

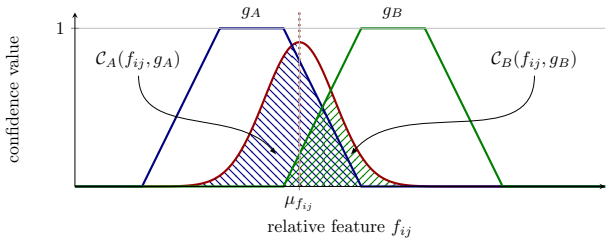

Figure 7: Deriving the confidence value $\mathcal{C}$ by integrating the product of feature f_{ij} with two association functions representing semantic labels A and B. The relational feature f_{ij} between entity e_i and e_j is expressed by a general normal distribution with mean μ_f and standard deviation σ_f.

object is at least protruding the corridor boundaries in lateral or longitudinal direction. The relational features between object o_i and corridor c_j is the Frenet position of the center point of the object's bounding box, which corresponds to $\mathbf{r}^F = [l_r \; d_{p,r}]^T$.

7.1.1 Lateral assignment confindence

Usually the reference line is not aligned with the center of the corridor. To enable a general definition of the lateral assignment function g_{lat}, the lateral displacement $d_{p,r}$ is modified by the reference line's offset to the center d_{ref}. The new lateral distance is denoted by $f_{lat} = d_\eta$. It is always specified relatively to the center of the corridor at the projection point's arc-length l_p. Since the modification by a constant is a linear operation the mean value is shifted by the offset, while its variance remains unchanged: $d_\eta \sim \mathcal{N}(\mu_{d_{p,r}} - d_{ref}, \sigma^2_{d_{p,r}})$.

The assignment function g_{lat} is defined by the corridor width w_{corr} and the projected object width $\hat{w}_{obj}$ at arc-length l_p:

$$\hat{w}_{obj} = \cos(\phi^F_{obj})\, l_{obj} + \sin(\phi^F_{obj})\, w_{obj} \tag{88}$$

where, ϕ^F_{obj} refers to object's orientation in Frenet coordinates, l_{obj} and w_{obj} to

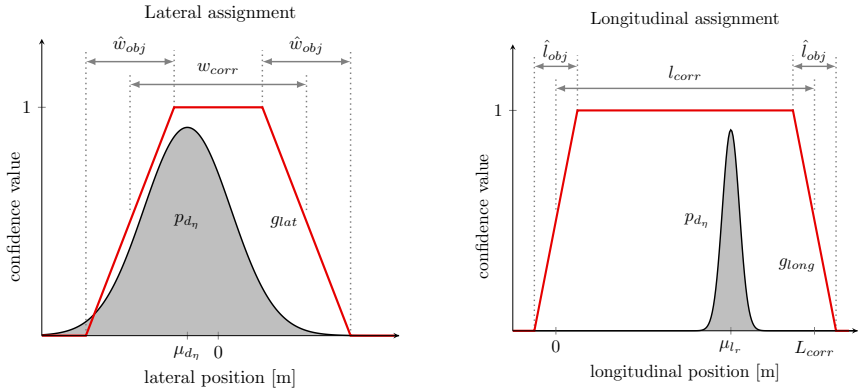

Figure 8: Generic definition of the lateral and longitudinal assignment functions, g_{lat} and g_{long}, in comparison to an exemplary probability density function (PDF) of relative feature f_{ij} per semantic label. The convolution of assignment function and PDF leads to the confidence value of the respective label, either *laterally-matched* or *longitudinally-matched*. The product of both provides the confidence of the *located-on* relation.

the object's length and width[3].

$$g_{lat}(w_{corr}, \hat{w}_{obj}, x) = \begin{cases} \frac{x}{\hat{w}_{obj}} + \frac{1}{2}\left(1 + \frac{w_{corr}}{\hat{w}_{obj}}\right) & \text{if } x \in [-\frac{w_{corr}+\hat{w}_{obj}}{2}, -\frac{w_{corr}-\hat{w}_{obj}}{2}] \\ 1 & \text{if } x \in [-\frac{w_{corr}-\hat{w}_{obj}}{2}, \frac{w_{corr}-\hat{w}_{obj}}{2}] \\ \frac{-x}{\hat{w}_{obj}} + \frac{1}{2}\left(1 + \frac{w_{corr}}{\hat{w}_{obj}}\right) & \text{if } x \in [\frac{w_{corr}-\hat{w}_{obj}}{2}, \frac{w_{corr}+\hat{w}_{obj}}{2}] \\ 0 & \text{else} \end{cases} \tag{89}$$

In case of $\hat{w}_{obj} = 0$ the first and last term vanishes and the assignment function becomes a rectangular function. The middle term vanishes if $w_{corr} \leq \hat{w}_{obj}$ and the assignment function degenerates to a triangle function. An example of a lateral assignment function g_{lat} with $0 < \hat{w}_{obj} < w_{corr}$ is visualized on the left side of figure 8.

This leads to a lateral assignment confidence which depends on mean and variance of the lateral feature's PDF, the corridor width w_{corr} and projected object's $\hat{w}_{obj}$ width:

$$\mathcal{C}_{lat}(d_\eta, g_{lat}) = \int_{-\infty}^{\infty} g_{lat}(w_{corr}, \hat{w}_{obj}, \xi)\, p_{d_\eta}(\xi)\, d\xi \tag{90}$$

[3]The dimensions of the object and its orientation in the Frenet frame are subject to sensor noise and thus only estimations. If the expected uncertainty of the length, width and orientation can assumed to be much smaller than the position uncertainty it is sufficient to use the mean values for the located-on relation. Otherwise, a lower and upper bound of the assignment confidence can be calculated using an arbitrary sigma interval of the resulting PDF of the object's projected width.

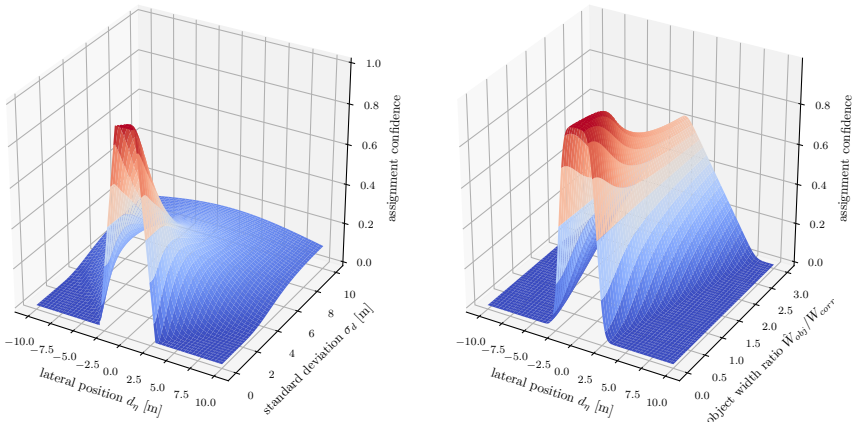

Figure 9: The variation of the lateral assignment confidence as a function of input parameters in equation (90). The left picture visualized the assignment confidence depending on the expected lateral displacement μ_{d_η} and its standard deviation σ_{d_η}. The corridor width is set to $w_{corr} = 4$ m, the projected object width to $\hat{w}_{obj} = 2$ m. In the right picture, the standard deviation is set constantly to $\sigma_{d_\eta} = 0.5$ m, while the ratio between projected object width and corridor width is varied in the range of $\hat{w}_{obj}/w_{corr} \in [0, 3]$.

The variation of the resulting confidence value, depending on all four input parameters, is presented in both pictures of Figure 9. It demonstrates, that the label confidence is much more sensitive to the lateral uncertainty than it is to the projected object width. An increasing relative object width asymptotically leads to a label confidence of 0.5, while growing position uncertainty generates confidence values towards 0.2, until the signal-noise ratio is too small to carry information. A significant association is only possible if the standard deviation is smaller than the corridor width.

7.1.2 Longitudinal assignment confindence

The longitudinal assignment confidence is calculated analogously. Here, the relative feature is defined by the position of the object's center along the reference line l_r. Due to the position uncertainty it is modeled as the PDF of the arc-length: $l_r \sim \mathcal{N}(\mu_{l_r}, \sigma^2_{l_r})$. Since all reference lines start at arc-length zero, no additional modification is needed to calculate a generic assignment function. Instead of the corridor and object's width, it depends on corridor's length l_{corr} and projected object length:

$$\hat{l}_{obj} = \sin(\psi^F_{obj})\, l_{obj} + \cos(\psi^F_{obj})\, w_{obj} \tag{91}$$

The picture on the right side of Figure 8 displays an example of a longitudinal assignment function g_{long}, defined along the reference line of a corridor. The

definition of g_{long} is similar to the lateral assignment function and shares the same corner cases as described above:

$$g_{long}(l_{corr}, \hat{l}_{obj}, x) = \begin{cases} \frac{x}{\hat{l}_{obj}} + \frac{1}{2} & \text{if } x \in [-\frac{\hat{l}_{obj}}{2}, \frac{\hat{l}_{obj}}{2}] \\ 1 & \text{if } x \in [\frac{\hat{l}_{obj}}{2}, l_{corr} - \frac{\hat{l}_{obj}}{2}] \\ \frac{-x}{\hat{l}_{obj}} + \frac{1}{2}\left(1 + \frac{2l_{corr}}{\hat{l}_{obj}}\right) & \text{if } x \in [l_{corr} - \frac{\hat{l}_{obj}}{2}, l_{corr} + \frac{\hat{l}_{obj}}{2}] \\ 0 & \text{else} \end{cases} \tag{92}$$

The resulting longitudinal assignment confidence depends therefore on mean μ_{l_r} and variance σ_{l_r} of object's center arc-length coordinate, the length of the corridor l_{corr} and projected object length $\hat{l}_{obj}$:

$$\mathcal{C}_{long}(l_r, g_{long}) = \int_{-\infty}^{\infty} g_{long}(l_{corr}, \hat{l}_{obj}, \xi)\, p_{l_r}(\xi)\, d\xi \tag{93}$$

In Figure 9 the effects of varying input parameters on the laterally-matched label are discussed. Since both, the lateral and longitudinal assignment function are equivalent, the effects of different parameter constellations are similar as well.

7.2 Moves-along relation

As the name already suggests is the *moves-along* relation defined between a corridor and a moving object. It is composed by the product of the *is-moving* confidence $\mathcal{C}_{mov}$ and the confidence of the semantic label which classifies the relative orientation between object and corridor $\mathcal{C}_{\psi_{rel}}$:

$$\mathcal{C}_{movAlng}(o_i, c_j, r_{movAlng}) = \mathcal{C}_{mov}(\dots)\, \mathcal{C}_{\psi_{rel}}(\dots) \tag{94}$$

7.2.1 Is-moving confidence

The classification of an object's moving status depends solely on its absolute velocity and is therefore detached from any corridor definition. Hence, the confidence of the *is-moving* label can be determined in the originating frame of the velocity estimation. In the Corridor library this corresponds to the Cartesian frame, but is applicable to any other frame as well.

Let $p_{\|\mathbf{v}\|}$ the PDF of absolute velocity $\|\mathbf{v}\|$, then the objective of the is-moving label is to determine if the velocity is greater than zero and the signal-noise-ratio (SNR) is significant. The SNR is defined by the ratio of expected value and standard deviation of the absolute velocity: $\text{SNR} = \mu_{\|\mathbf{v}\|}/\sigma_{\|\mathbf{v}\|}$. A SNR higher than 1 indicates more signal than noise.

In the Corridor library, the *is-moving* confidence is required to be in the range of $\mathcal{C}_{mov} \in [0, 1]$. Thus, the assignment function g_{mov} follows a step function, where the start of the positive interval is defined by the product of SNR

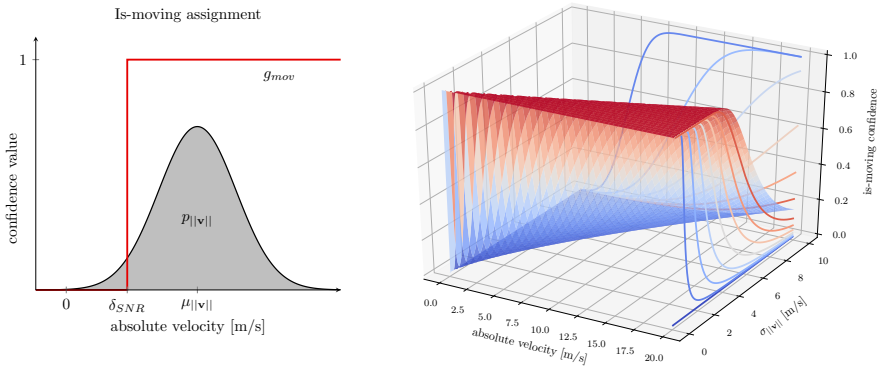

Figure 10: Left: generic definition of the is-moving assignment function in comparison to an exemplary probability density function (PDF) of absolute velocity. The integral of the product between assignment function and PDF leads to the confidence value of the label.
Right: Moving confidence as function of absolute velocity $\|\mathbf{v}\|$ and its standard deviation $\|\mathbf{v}\|$ in Cartesian coordinates. For non-zero velocities the moving confidence is approximately one, if the majority of the PDF is above the SNR limit δ_{SNR}.

and standard deviation of the absolute velocity, $\delta_{SNR} = \mathrm{SNR}\,\sigma_{\|\mathbf{v}\|}$:

$$g_{mov}(\delta_{SNR}, x) = \begin{cases} 1 & \text{if } x \geq \delta_{SNR} \\ 0 & \text{else} \end{cases} \tag{95}$$

Default value for the SNR is set to 3. Consequentially, the is-moving confidence is defined by the area of the absolute velocity's PDF which is above the SNR limit δ_{SNR}:

$$\mathcal{C}_{mov}(\|\mathbf{v}\|, g_{mov}) = \int_{-\infty}^{\infty} g_{mov}(\delta_{SNR}, \xi)\, p_{\|\mathbf{v}\|}(\xi)\, d\xi \tag{96}$$

The left picture in Figure 10 shows an example of *is-moving* assignment function and corresponding PDF of the absolute velocity. The right side visualizes the resulting confidence values for varying means and standard deviations of the absolute velocity.

7.2.2 Semantic labels of relative orientation

Usually, the orientation of an object is determined by its moving direction. In standstill, it is a much harder problem to solve which requires specifically trained detectors.

However, once the orientation information is available the classification of relative orientation between object o_i and corridor c_j is straight forward. In the

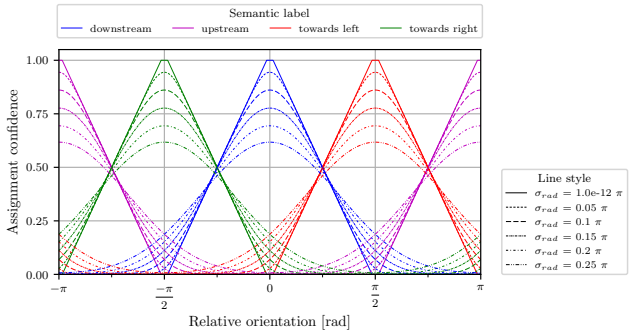

Figure 11: Semantic label, based on relative orientation between object and reference line (Frenet frame), $\phi_{rel} \in [-\pi, \pi]$. The different line styles represent some exemplary standard deviations of the object's orientation probability density function (PDF).

special case of planar coordinate frames is the transformation of a Cartesian angle in Frenet coordinates a linear operation:

$$\begin{aligned} \psi_{obj}^{F} &= \psi_{obj}^{C} - \psi_{p} \\ &\sim \mathcal{N}\left(\mu_{\psi_{obj}^{C}} - \psi_{p}, \sigma_{\psi_{obj}^{C}}^{2}\right) \end{aligned} \tag{97}$$

Please recall, that ψ_p denotes the orientation of the reference line at arc-length l_p. If the object orientation is not explicitly estimated by the environment perception, the Corridor library provides a UT conversion of velocity vectors into polar coordinates.

Unlike the above discussed semantic labels consists the *relative orientation* label of multiple association functions, which separates the Frenet orientation into four sub-labels. Each defined by its own assignment function, modeled as a symmetric trapezoidal function around a definition angle of the direction ψ_{dir}:

- **downstream**, g_{down}
  Directed along increasing arc-length definition of the corridor's reference line: $\psi_{down} = 0$ [rad].

- **upstream**, g_{up}
  Opposing direction of the corridor's reference line: $\psi_{up} = \pm\pi$ [rad].

- **towards-left**, g_{left}
  Directed towards the left-hand side of the corridor: $\psi_{left} = \frac{\pi}{2}$ [rad].

- **towards-right**, g_{right}
  Directed towards the right-hand side of the corridor: $\psi_{left} = -\frac{\pi}{2}$ [rad].

For an arbitrary definition angle of the direction ψ_{dir} the assignment function is defined by:

$$g_{dir}(\delta_\psi, x) = \begin{cases} \frac{2}{\pi - 4\delta_\psi}(x - \psi_{dir}) + \frac{\pi - 2\delta_\psi}{\pi - 4\delta_\psi} & \text{if } x \in [\psi_{dir} - (\frac{\pi}{2} + \delta_\psi), \psi_{dir} - \delta_\psi] \\ 1 & \text{if } x \in [\psi_{dir} - \delta_\psi, \psi_{dir} + \delta_\psi] \\ \frac{-2}{\pi - 4\delta_\psi}(x - \psi_{dir}) + \frac{\pi - 2\delta_\psi}{\pi - 4\delta_\psi} & \text{if } x \in [\psi_{dir} + \delta_\psi, \psi_{dir} + \frac{\pi}{2} + \delta_\psi] \\ 0 & \text{else} \end{cases} \tag{98}$$

The direction of the corridor is derived from the tangent vector of the reference line. Since this line is only an approximation of the corridor course, a small plateau is defined around the defining angle of the corridor:

$$\delta_\psi = \max\left(\frac{\pi}{64}, \mathrm{SNR}\,\sigma_{\psi_{obj}}\right) \tag{99}$$

Default value is set to $\mathrm{SNR} = 3$.

Subsumed by the discrete probability distribution over all sub-labels, the resulting confidence value of the *relative-orientation* label does always add up to exactly one:

$$\begin{aligned} \mathcal{C}_{\psi_{rel}}(\psi^F_{obj}, g_{dir}) &= \sum_{dir} \int_{-\infty}^{\infty} g_{dir}(\psi_{dir}, \xi)\, p_{\psi^F_{obj}}(\xi)\, d\xi \\ &\stackrel{!}{=} 1 \end{aligned} \tag{100}$$

Figure 11 displays the resulting confidence values of all labels at an arbitrary relative orientation ψ_{rel}, evaluated for some exemplary standard deviations of the object's orientation PDF.

Within the Corridor library, such sub-label confidence distribution is realized by the *SemanticLabelSet* class. It provides simple functions to initialize and access the sub-label distribution and values for further evaluations.

References

Brent, Richard P. (2013). *Algorithms for Minimization without Derivatives*. Courier Corporation.

Floater, Michael S. et al. (Nov. 2006). "Point-Based Methods for Estimating the Length of a Parametric Curve". In: *Journal of Computational and Applied Mathematics* 196.2, pp. 512–522. ISSN: 03770427. DOI: 10.1016/j.cam.2005.10.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377042705006023 (visited on 12/18/2020).

Hasberg, C. et al. (June 2012). "Simultaneous Localization and Mapping for Path-Constrained Motion". In: *IEEE Transactions on Intelligent Transportation Systems* 13.2, pp. 541–552. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2011.2177522. URL: http://ieeexplore.ieee.org/document/6112801/ (visited on 12/14/2020).

Hendeby, Gustaf et al. (2007). "On Nonlinear Transformations of Gaussian Distributions". In: Technical Report from Automatic Control, p. 3.

Julier, S.J. (2002). "The Scaled Unscented Transformation". In: *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*. Proceedings of 2002 American Control Conference. Anchorage, AK, USA: IEEE, 4555–4559 vol.6. ISBN: 978-0-7803-7298-6. DOI: 10.1109/ACC.2002.1025369. URL: http://ieeexplore.ieee.org/document/1025369/ (visited on 12/29/2020).

Menegaz, Henrique M. T. et al. (Oct. 2015). "A Systematization of the Unscented Kalman Filter Theory". In: *IEEE Transactions on Automatic Control* 60.10, pp. 2583–2598. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2015.2404511. URL: http://ieeexplore.ieee.org/document/7042740/ (visited on 12/15/2020).

Petrich, Dominik et al. (Oct. 2014). "Assessing Map-Based Maneuver Hypotheses Using Probabilistic Methods and Evidence Theory". In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC). Qingdao, China: IEEE, pp. 995–1002. ISBN: 978-1-4799-6078-1. DOI: 10.1109/ITSC.2014.6957818. URL: http://ieeexplore.ieee.org/document/6957818/ (visited on 01/05/2021).

Ramond, Paul (Nov. 5, 2020). *Abel-Ruffini's Theorem: Complex but Not Complicated!* arXiv: 2011.05162 [math]. URL: http://arxiv.org/abs/2011.05162 (visited on 12/20/2020).

Walter, Marcelo et al. (1996). "Approximate Arc Length Parametrization". In: p. 8.

Wan, E.A. et al. (2000). "The Unscented Kalman Filter for Nonlinear Estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. Symposium on Adaptive Systems for Signal Processing Communications and Control. Lake Louise, Alta., Canada: IEEE, pp. 153–158. ISBN: 978-0-7803-5800-3. DOI: 10.1109/ASSPCC.2000.882463. URL: http://ieeexplore.ieee.org/document/882463/ (visited on 12/30/2020).

Wang, Delun et al. (June 22, 2015). *Kinematic Differential Geometry and Saddle Synthesis of Linkages*. 1st ed. Wiley. ISBN: 978-1-118-25504-9 978-1-118-25505-6. DOI: 10.1002/9781118255056. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118255056 (visited on 12/13/2020).

Wang, Hongling et al. (2002). "Arc-Length Parameterized Spline Curves for Real-Time Simulation". In: *International Conference on Curves and Surfaces*. International Conference on Curves and Surfaces. Vol. 5, pp. 387–396.