Cargo-ANTs Working Document

Trajectory Representation and Specification for Planning and Control of AGVs and ATs

Roland Philippsen, Halmstad University

March 24, 2016

1 Problem Statement

We need to send the planned paths / trajectories from the motion generation pipeline (which is being developed by HH as a collection of ROS nodes) to the two kinds of vehicle platforms used in the project. The AGVs are under the responsibility of ICT, who have expressed a preference for receiving trajectories as Clothoids and a separate velocity profile. Volvo is responsible for the ATs and prefers to receive parametric polynomials curves, such as cubic or quintic B-splines (the question of how to encode time or speed information appears to not yet have been settled). A number of trajectory planning and optimization approaches that HH expects to consider during the work on Cargo-ANTs rely on representing trajectories as a list of points regularly sampled according to some constant time-step. The interests and needs of these three stakeholders need to be brought together.

In addition to these representational questions, we also need to more clearly define the responsibilities in terms of implementation, as several interacting software and hardware components need to work together at the interface between the HH motion stack and the ICT and Volvo vehicle platforms.

2 Paths as Parametric Curves

For the purpose of planning and controlling the motion of vehicles in the plane, we define a **path** as a sufficiently continuous curve from a given start point to a given goal point. We consider the path to be **parametrized** over a closed interval of real numbers. Formally $p : \Lambda \to \mathbb{R}^2$ and

$$p(\lambda) = \begin{pmatrix} x \\ y \end{pmatrix} \tag{1}$$

$$\lambda \in \Lambda = [\lambda_{\text{start}}, \lambda_{\text{end}}]. \tag{2}$$

Notice that we could alternatively define the path to exist in configuration space $\mathcal{C}$, i.e. include the heading θ. This would alleviate problems to determine path heading at cusp points, but introduce constraints between the components of the path points. In practice, cusp points should be avoided anyway, and maneuvers that require switching between forward and reverse need to go through a zero-velocity phase anyway and can thus be handled using two or more distinct cusp-free paths.

Thus, we can assume a non-zero tangent, and we rely on geometrical analysis to determine θ as the angle between the X-axis and the tangent. Writing the configuration as $q \in \mathcal{C}$,

$$q(\lambda) = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{3}$$

$$\theta(\lambda) = \arctan \frac{y'}{x'} \tag{4}$$

where x' and y' denote the derivatives of the path components with respect to λ. In practice, arctan is computed using the `atan2` standard library function in order to avoid singularities and robustly handle quadrants.

An important property of curves is their (extrinsic) **curvature** κ, which is defined as the inverse of the radius of the osculating circle at a given point. Expressing and controlling κ is crucial for respecting steering constraints. It is also a central ingredient for understanding paths that are represented via Clothoids, which are generated via a linearly relationship between path length and curvature. Curvature can be derived as [**?**] (here we use κ to denote signed curvature, whereas sometimes κ refers to the magnitude of the curvature)

$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}}. \tag{5}$$

Notice that curvature is not defined when the tangent vector becomes zero. Care must thus be taken to avoid such situations. In practice, this boils down to avoiding cusps (as the parametrization family is typically chosen to avoid other degeneracies).

3 Trajectories as Time-Parametrized Curves

We define a **trajectory** to be a curve parametrized by time. Under the right assumptions (which we can make in our practical use cases), this is a straightforward reparametrization [**?**]. Simply let λ be a sufficiently continuous function of time and use function composition, thus

$$\lambda : T \to \Lambda \qquad\qquad T = [t_{\text{start}}, t_{\text{end}}] \tag{6}$$

$$p \circ \lambda : T \to \mathbb{R} \qquad\qquad \text{i.e.} \quad p = p(\lambda(t)) \tag{7}$$

Notice that in practice, λ is usuallly strictly monotonic over an open time interval, with zero derivative at the start and end of the interval. Under this assumption λ is invertible, i.e. we can compute $t = t(\lambda)$. Using the chain rule and assuming that $\lambda(t)$ is given, it is now straightforward to compute properties that are relevant for control, such as the translational and rotational velocities v and ω.

$$v(t) = \left\| \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \right\| = \left\| \begin{pmatrix} x' \\ y' \end{pmatrix} \right\| \dot{\lambda} = \dot{\lambda}\sqrt{x'^2 + y'^2}, \tag{8}$$

where dotted symbols denote time derivatives, e.g. $\dot{\lambda} = d\lambda/dt$. The rotational velocity is a little more intricate,

$$\omega(t) = \dot{\theta} = \frac{d}{dt} \arctan \frac{\dot{y}}{\dot{x}} = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}. \tag{9}$$

When computing time-derivatives, be careful not to forget the chain rule to take into account the influence of the varying rate of change of the path parameter. Thus, by repeated application of the chain rule,

$$\dot{x} = x'\dot{\lambda} \tag{10}$$

$$\ddot{x} = x''\dot{\lambda}^2 + x'\ddot{\lambda} \tag{11}$$

$$\dddot{x} = x'''\dot{\lambda}^3 + 3x''\dot{\lambda}\ddot{\lambda} + x'\dddot{\lambda} \tag{12}$$

and similarly for $\dot{y}$ etc.

It is worth noting that a trajectory need not be *directly represented* as $\lambda(t)$. Often, the path is first reparametrized by arc length s, as this is a more natural space to reason about constant bounds on time derivatives of the induced motion. In this case, we end up with the composition $p(\lambda(s(t)))$. This is illustrated with a worked example in section **??**.

The approach for representing paths and time parametrizations depends on application specific design- and implementation trade-offs. For example, state-time planning will be simplified if time is explicitly represented (i.e. there hardly is a notion of a time-independent path which gets subsequently parametrized by time). On the other hand, tying a reactive trajectory follower into an overall system is simplified when the velocity profile is kept separate from the path, such that it can easily be updated separately.

4 Commonly Used Path and Trajectory Representations

One of the easiest ways to generate smooth curves is to use **polynomials** of λ. Typical examples include B-splines [?] and (composite) Bézier curves. Their advantage is the simplicity of their expressions, the ease with which they can be concatenated to any desired degree of continuity, the trivial computation of their derivatives, and the natural link between their control points and the locations they traverse. Their drawbacks include the fact that several properties that are important for planning and control vary in non-trivial ways as λ changes. This includes the magnitude of their tangent vector (which is only a minor nuisance in practice) as well as their curvature (which is an important point for vehicles with strict steering constraints). Another drawback is that they are not naturally free of cusp points, so in practice we need to check for such points and handle them (for instance by cutting the curve in two segments at each cusp point).

At the other extreme, the possibly "most natural" way to define a smooth parametric curve is to concatenate pieces of **Clothoids** [?]. The idea here is to let the curvature κ be a piece-wise linear function of arc length. Thus, λ is directly the natural parametrization of the curve (making it trivial to e.g. attach appropriate velocity profiles), and it is trivial to control the curvature (i.e. keep it from violating steering constraints). Furthermore, Clothoid segments are free of cusps, and it is trivial to ensure that the knots where segments connect do not form cusps either. The problem with Clothoids is that it is difficult to construct a connection between start and end configurations, particularly when obstacles need to be taken into account. A solution to this is proposed in [?].

Another popular option is to avoid the problem of choosing a family of parametric curves, instead opting for some form of regularly sampled **list of points**. This is equivalent to a fine-grained approximation with line segments (one could also use higher-order schemes to estimate derivatives along the curve though). This is particularly beneficial for approaches that optimize or otherwise handle trajectories holistically, as opposed to separately considering the path and its time-parametrization. For example, CHOMP [?] treats a trajectory as a point in a high-dimensional space, and subjects it to gradient descent. As another example, two-point boundary value solvers underlying state-lattice approaches [?] tend to solve in the space of control signals, typically resulting in trajectories that do not belong to an easily discernible family of parametric curves.

5 Illustration: Three-Segment Uniform Cubic B-Spline

A uniform cubic B-spline requires four control points per segment and uses cubic expressions in λ to compute a moving weighted average between these points. Using three segments makes it straightforward to connect the resulting curve to a given start and end configuration (see Figure **??**).

$$p(\lambda) = \begin{cases} p_0(\lambda) & \text{for } \lambda \in [0,1] \\ p_1(\lambda - 1) & \text{for } \lambda \in [1,2] \\ p_2(\lambda - 2) & \text{for } \lambda \in [2,3]. \end{cases} \tag{13}$$

This requires six control points P_i. The start pose determines P_0, P_1, P_2: P_1 is placed onto the center of the vehicle at the start, P_0 and P_2 at equal distance behind and in front of the vehicle along its heading. The three end control points P_3, P_4, P_5 are determined analogously.

Each segment is defined as follows

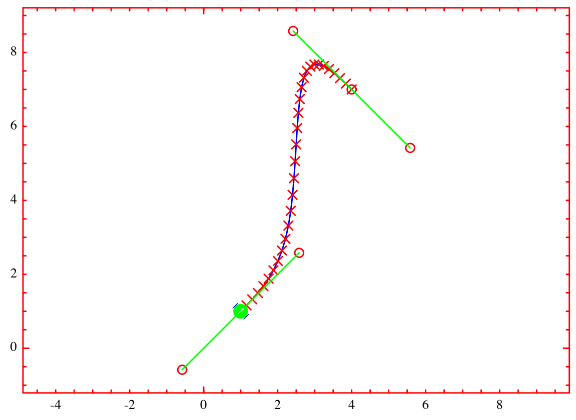


Figure 1: *Uniform cubic B-spline with three segments. The starting pose of the vehicle is shown as a filled green circle. The six control points are shown as hollow red circles. The red crosses indicate points along the curve resulting from regular sampling with constant* $\Delta\lambda$*.*

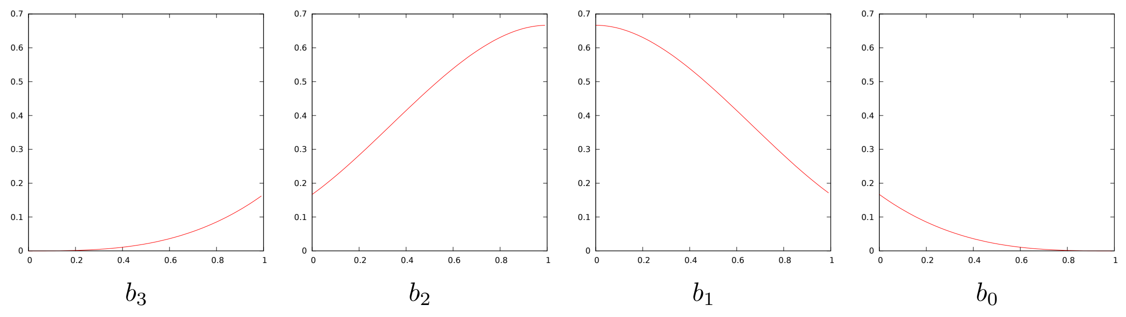


Figure 2: *The four basis functions of a uniform cubic B-spline, arranged in reverse order. This indicates the strength of a given control point. For example,* P_3 *will go from a weight of 0 to 1/6 during the first segment* $p_0(\lambda)$*, then up to 2/3 during* $p_1(\lambda)$*, and then its influence will wane again as it progresses through* p_2 *and* p_3*.*

$$p_i(\mu) = \sum_{k=0}^{3} b_k(\mu) P_{i+k} \quad \text{where} \begin{cases} b_0 & = (-\mu^3 + 3\mu^2 - 3\mu + 1)/6 \\ b_1 & = (3\mu^3 - 6\mu^2 + 4)/6 \\ b_2 & = (-3\mu^3 + 3\mu^2 + 3\mu + 1)/6 \\ b_3 & = \mu^3/6 \\ \mu & \in [0, 1]. \end{cases} \tag{14}$$

Figure **??** illustrates how the four bases b_k produce a weighted average of control points that shifts the point $p(\lambda)$ from the start to the end point as λ goes from 0 to 3. Notice that μ is introduced for notational convenience with respect to the three shifts in λ for the three segments of $p(\lambda)$. In order to compute θ and κ, we need the first and second derivatives of p_i with respect to λ, which are straightforward to derive and omitted here.

In order to parametrize a path by time, one possibility is to proceed as follows.

1. Determine the natural parametrization of the curve, i.e. by arc length s. This gives us the total curve length L and also allows to determine $\lambda = \lambda(s)$.

   We proceed by computing $s(\lambda)$ by integration. According to earlier remarks, this will be invertible. Both the integration and the inversion will be performed numerically, as they have no closed form solution in this example.

$$s(\lambda) = \int_{\lambda_{\text{start}}}^{\lambda} \|p'(\mu)\| d\mu \tag{15}$$

$$L = s(\lambda_{\text{end}}) \tag{16}$$

2. Determine a time profile $s(t)$ that respects desired bounds. For example, we computing a trapezoidal velocity profile over L with given bounds on acceleration $\hat{a}$ and velocity $\hat{v}$.

$$\dot{s}(t) = \begin{cases} \hat{a}t & \text{for } 0 < t \leq t_1 \\ \hat{v} & \text{for } t_1 < t \leq t_2 \\ \hat{v} - \hat{a}(t - t_2) & \text{for } t_2 < t \leq t_3 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$t_1 = \hat{v}/\hat{a} \tag{18}$$

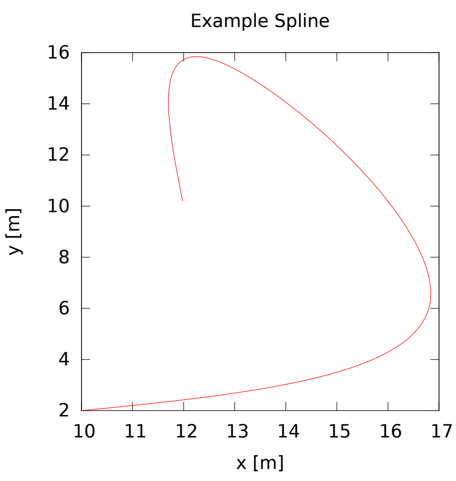
$$t_2 = L/\hat{v} \tag{19}$$

$$t_3 = t_1 + t_2 \tag{20}$$

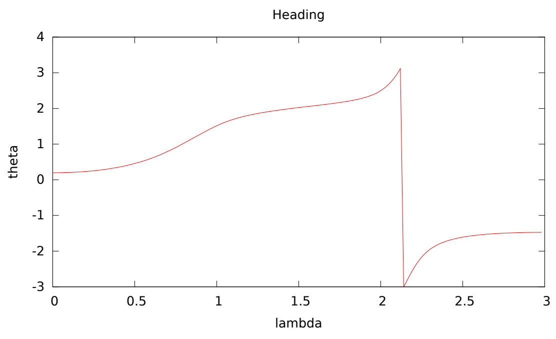
   Notice that the above assumes there always is a constant-velocity phase. This is not guaranteed in practice, it can happen that the trapezoidal profile degenerates to a triangular one. Also, in practice it may make sense to start the profile at a time other than $t = 0$. These considerations lead to slightly more involved expressions, which are omitted for the sake of brevity.

   Then, $s(t)$ is found by integration. For trapezoidal (and triangular) velocity profiles this leads to a piece-wise quadratic expression.

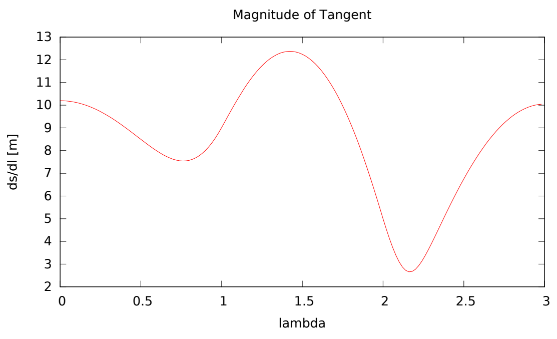
3. Determine $\lambda(t) = \lambda(s(t))$ and from there $p(\lambda(s(t)))$. Notice that $\lambda(s)$ is obtained via a numerical scheme in this example. Figure **??** shows the resulting time profiles. It can be seen that the desired trapezoidal velocity profile is respected very well in regions of low curvature. However, higher magnitudes of κ lead to over- or under-shooting the targeted velocity.



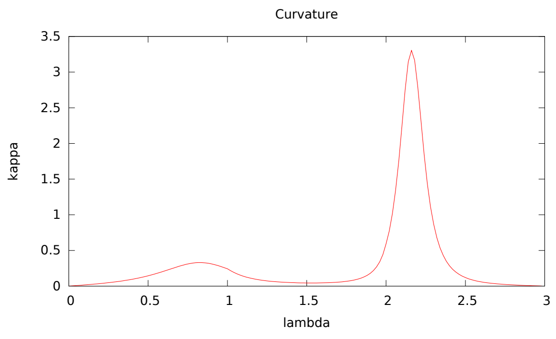
(a) planar curve (starts in lower left)



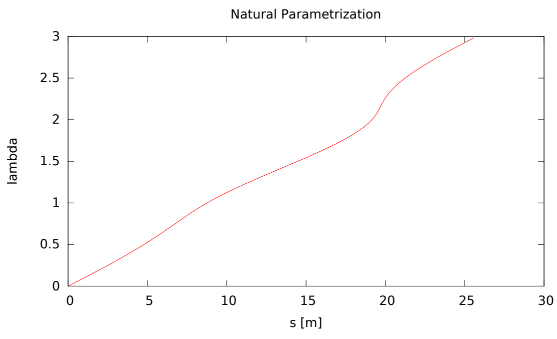
(b) heading



(c) $\|p'(\lambda)\|$



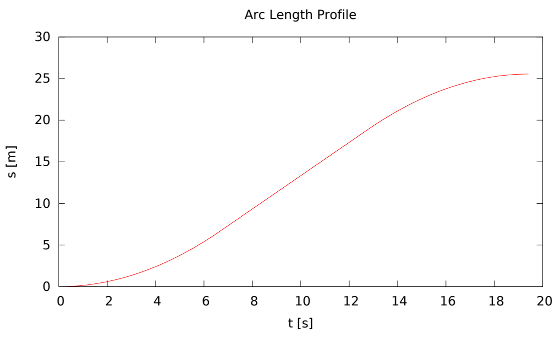
(d) curvature



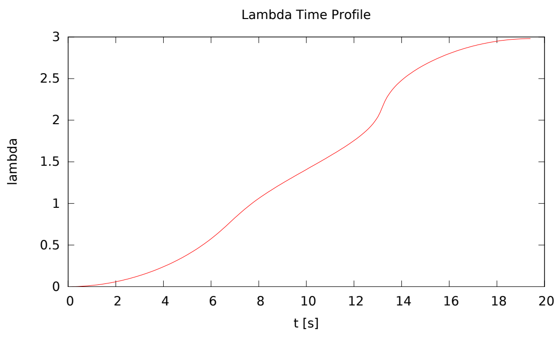
(e) $\lambda(s)$

Figure 3: *Illustrating example of a three-segment uniform cubic B-spline and some of its properties. Notice that all properties relate to λ here, reparametrizations will "squish and stretch" the graphs that show the properties, but the set of points in the curve is invariant.*

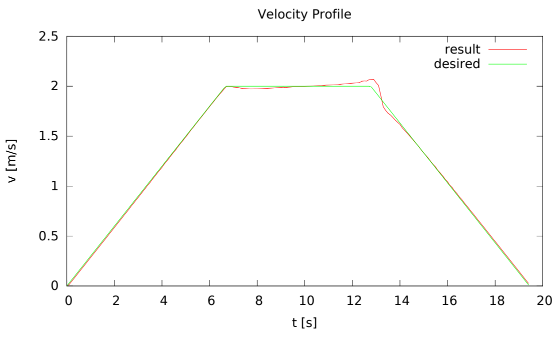
.



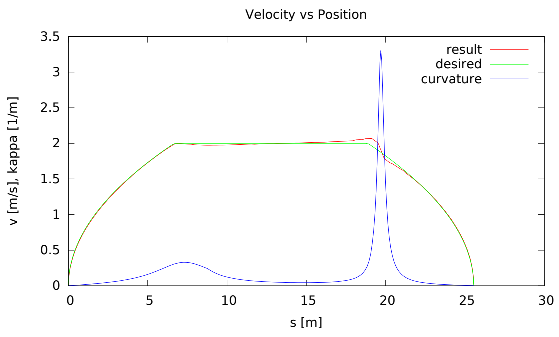
(a) desired $s(t)$



(b) resulting $\lambda(t) = \lambda(s(t))$



(c) desired and resulting $\dot{p}(t)$



(d) $\dot{p}(s)$ with overlaid $\kappa(s)$

Figure 4: *Time parametrization for the example spline shown in Figure* **??**.