AImotion Bavaria

Exploration of Efficient Computation for Trajectory Planning via Fixed-Point Arithmetic

PARS-Workshop 2024

Wenguang Xu, Richard Membarth 26.09.2024

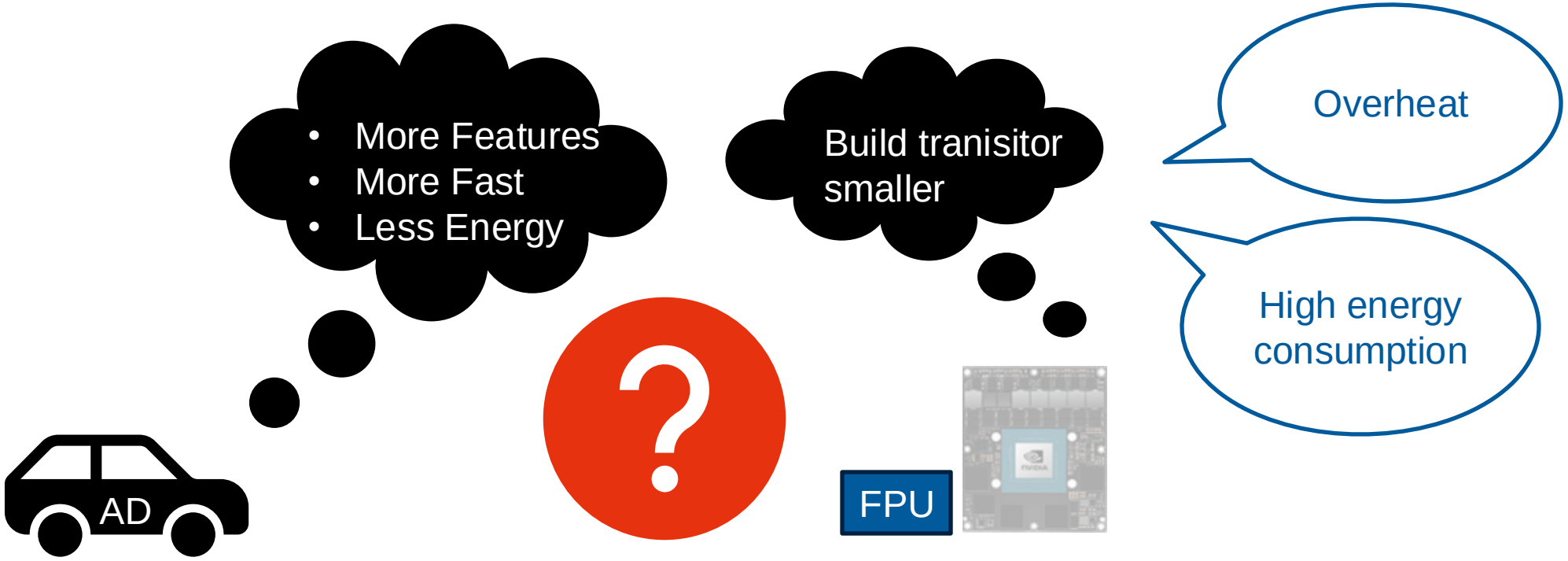
Work supported by BMBF as part of the MANNHEIM-AutoDevSafeOps project

dfki ai

Technische Hochschule Ingolstadt

Research Motivation

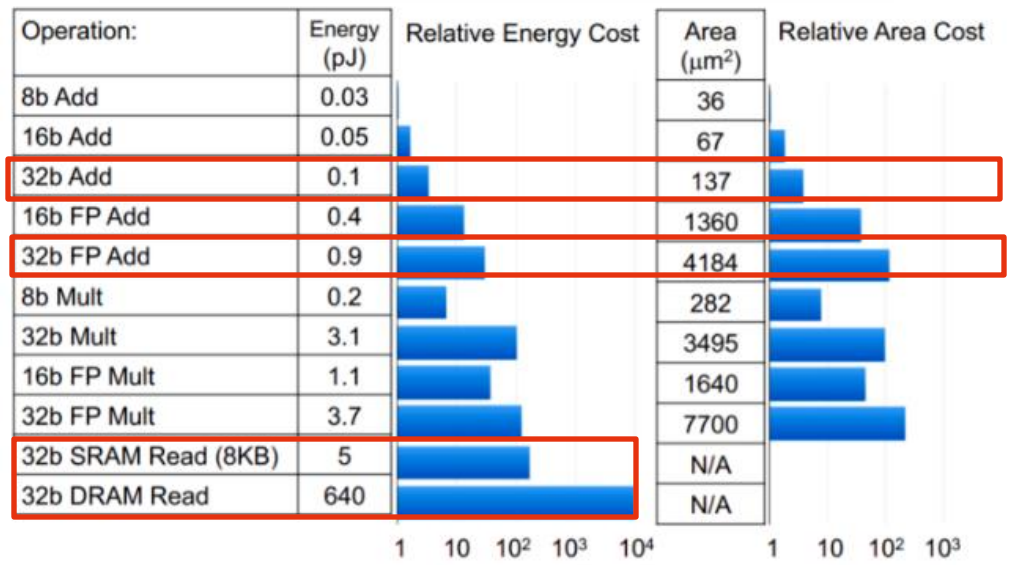
Introduction to the Problem



Research Motivation

Significance of Fixed-Point Arithmetic

Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A



Cited from Horowitz, Mark "1.1 Computing's Energy Problem (and what we can do about it)". ISSCC 2014

Precision Loss

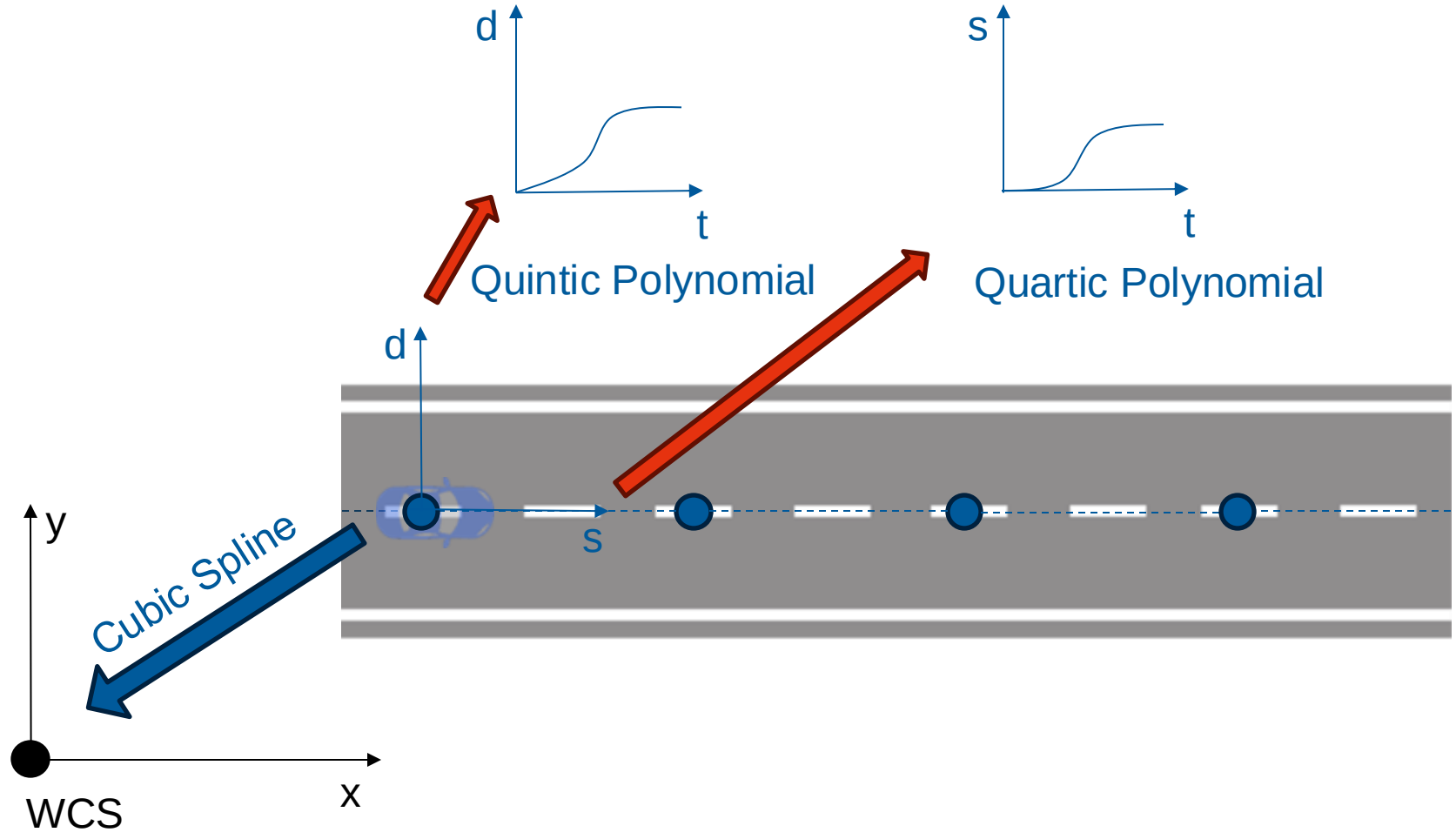
Research Motivation

Constraint and Goal

- Goal: explore the required **bit width** of fixed-point number to maintain the precision in trajectory planning
- Constraints
    - Precision loss to ±2 cm (identical to the precision loss of LiDAR)
    - Plan trajectory in 50 meters within 3 seconds

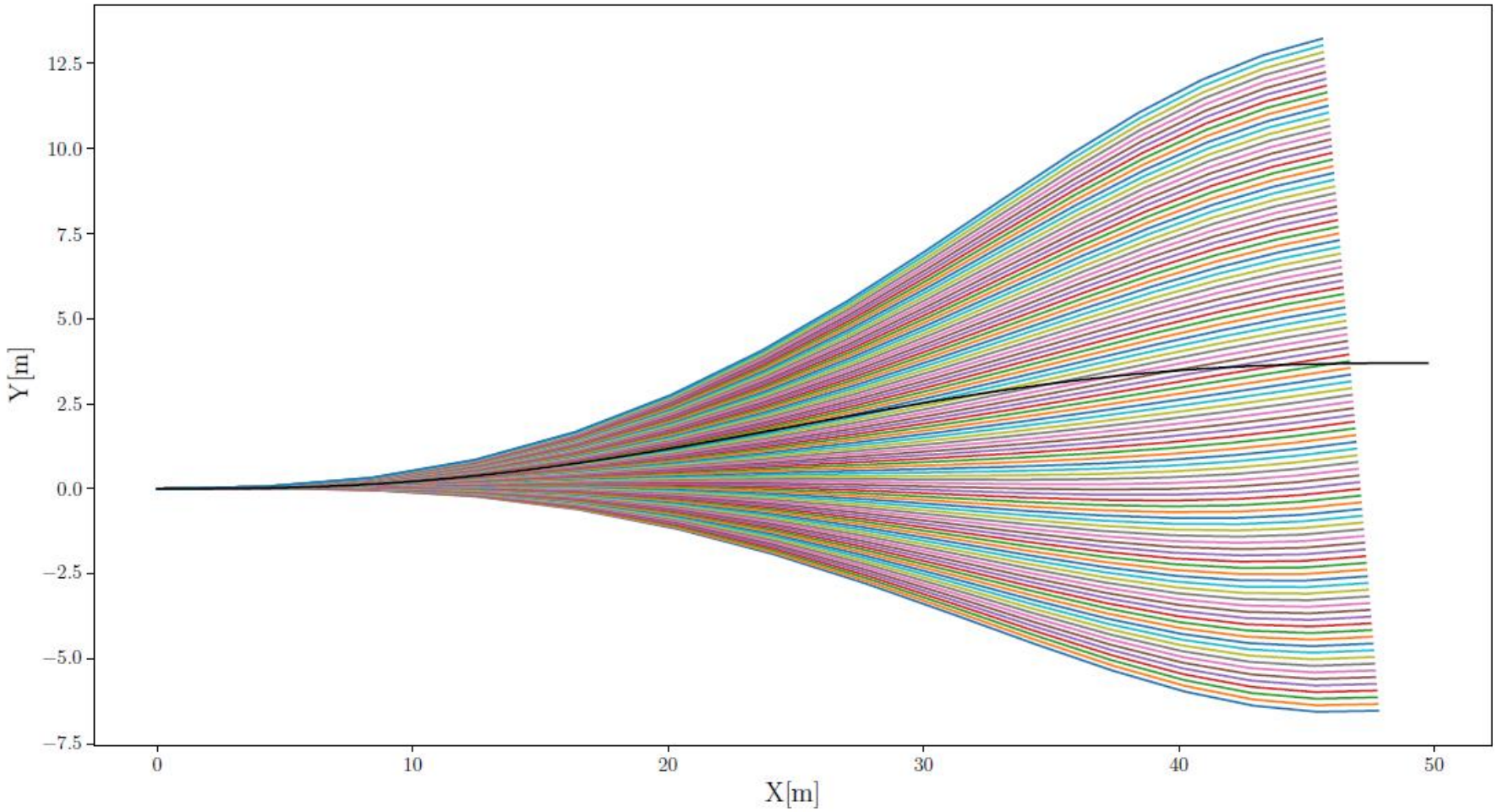
Approach

Frenet Optimal Planner



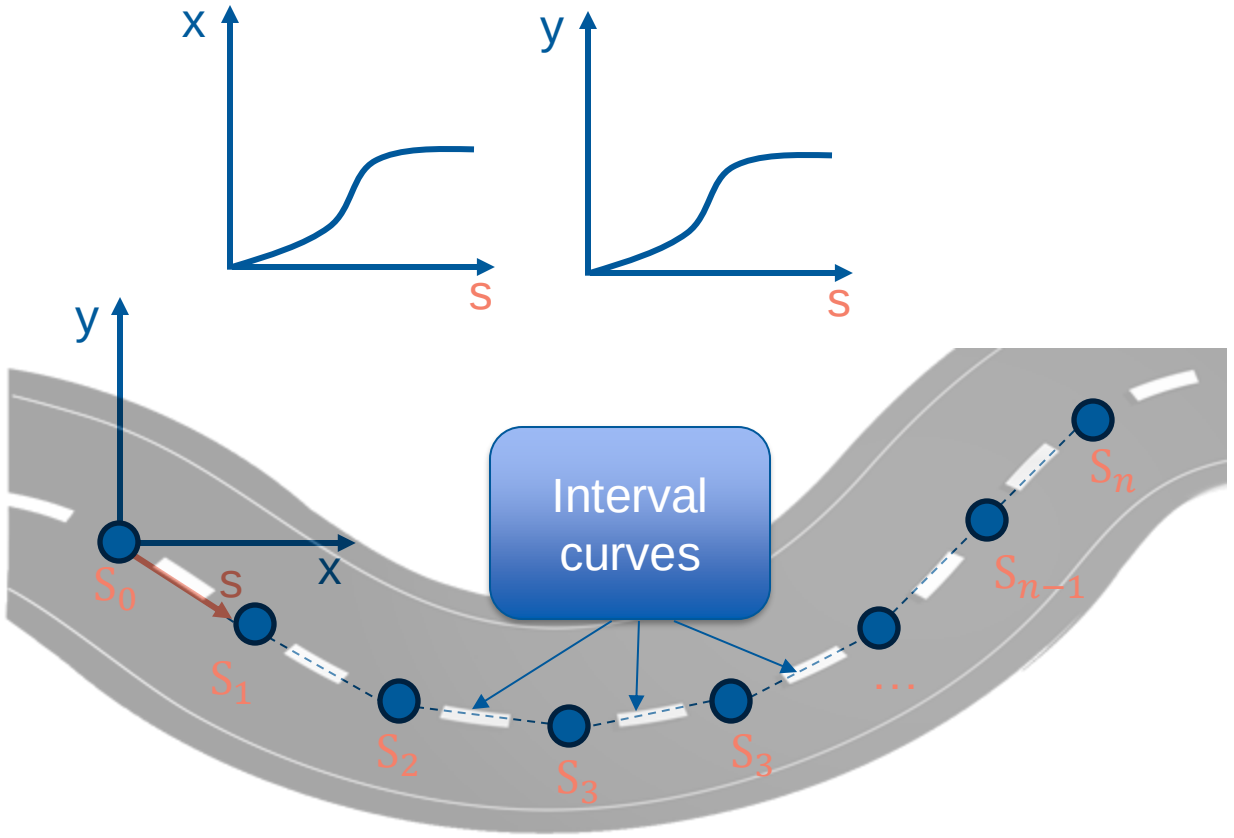
Approach

Frenet Optimal Planner



Approach

Cubic Spline



$$F_i(s) = a_i + b_i(s - s_i) + c_i(s - si)^2 + d_i(s - s_i)^3$$

With boundary conditions:

- $x_i = F_i(s_i)$ and $x_{i+1} = F_i(s_{i+1})$
- $F_i(s_{i+1}) = F_{i+1}(s_i)$, $F_i'(s_{i+1}) = F_{i+1}'(s_i)$ and $F_i''(s_{i+1}) = F_{i+1}''(s_i)$
- $F_0''(s_0) = 0$ and $F_n''(s_{n+1}) = 0$

Approach

Cubic Spline

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & 0 & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ \vdots \\ m_{n-1} \\ m_n \end{bmatrix} = 6 \begin{bmatrix} 0 \\ \frac{y_2-y_1}{h_2} - \frac{y_1-y_0}{h_1} \\ \frac{y_3-y_2}{h_3} - \frac{y_2-y_1}{h_2} \\ \vdots \\ \frac{y_n-y_{n-1}}{h_{n-1}} - \frac{y_{n-1}-y_{n-2}}{h_{n-2}} \\ 0 \end{bmatrix}$$

How to avoid using matrix libraries ?

Approach

Thomas Algorithm

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}$$

O(n³) ->O(n)

Forward sweep:

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & i = 1, \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & i = 2,3,\dots,n-1 \end{cases}$$

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & i = 1, \\ \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & i = 2,3,\dots,n-1 \end{cases}$$

Backward sweep:

$$x_n = d'_n$$

$$x_i = d'_i - c'_i x_{i+1}, \quad i = n-1,\ n-2,\dots,1$$

Approach

Quintic and Quartic Polynomial

Parameter for boundary conditions: x_s, x_e, v_s, v_e, a_s, a_e

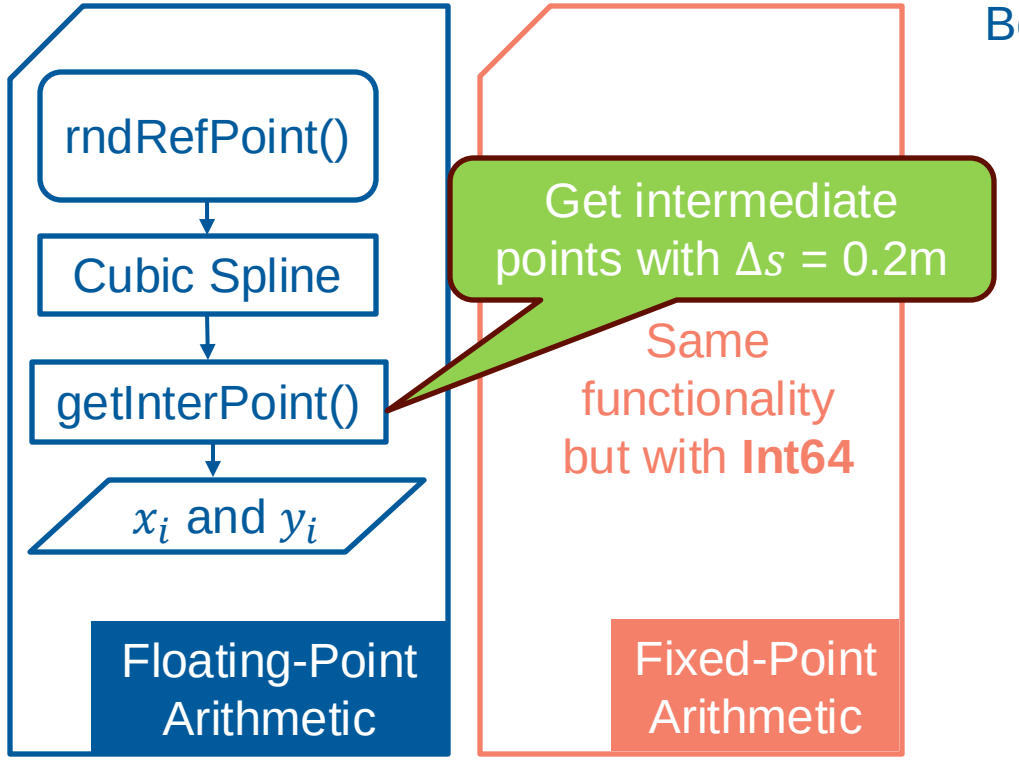
$$\mathrm{d}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

Parameters for boundary conditions x_s, v_s, v_e, a_s, a_e

$$\mathrm{d}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4$$

Experiment

Cubic Spline



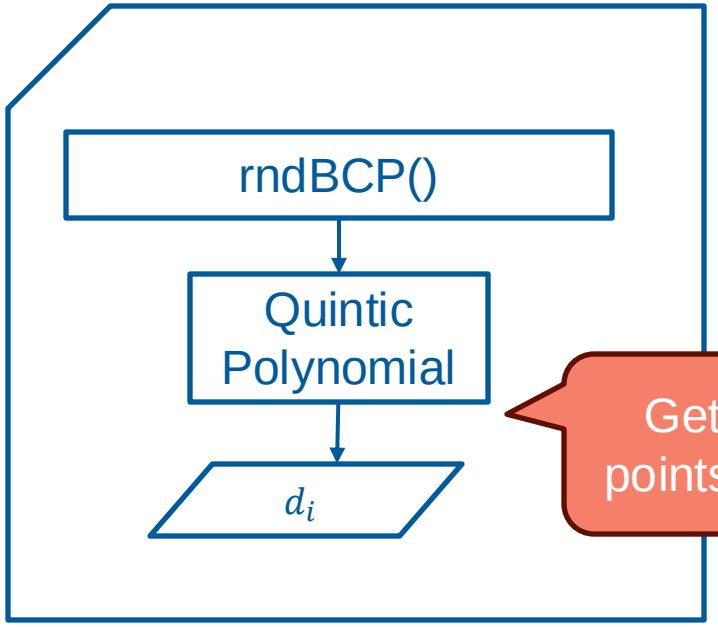
Boundary conditions:

- $x_s, x_e \in [-50,50]$ and $y_s, y_e \in [-50,50]$,
- $v_s, v_e, a_s, a_e = 0$

Repeat for 100 times with random value of boundary condition parameters

Experiment

Quintic Polynomial



Get intermediate points with Δt = 0.1s

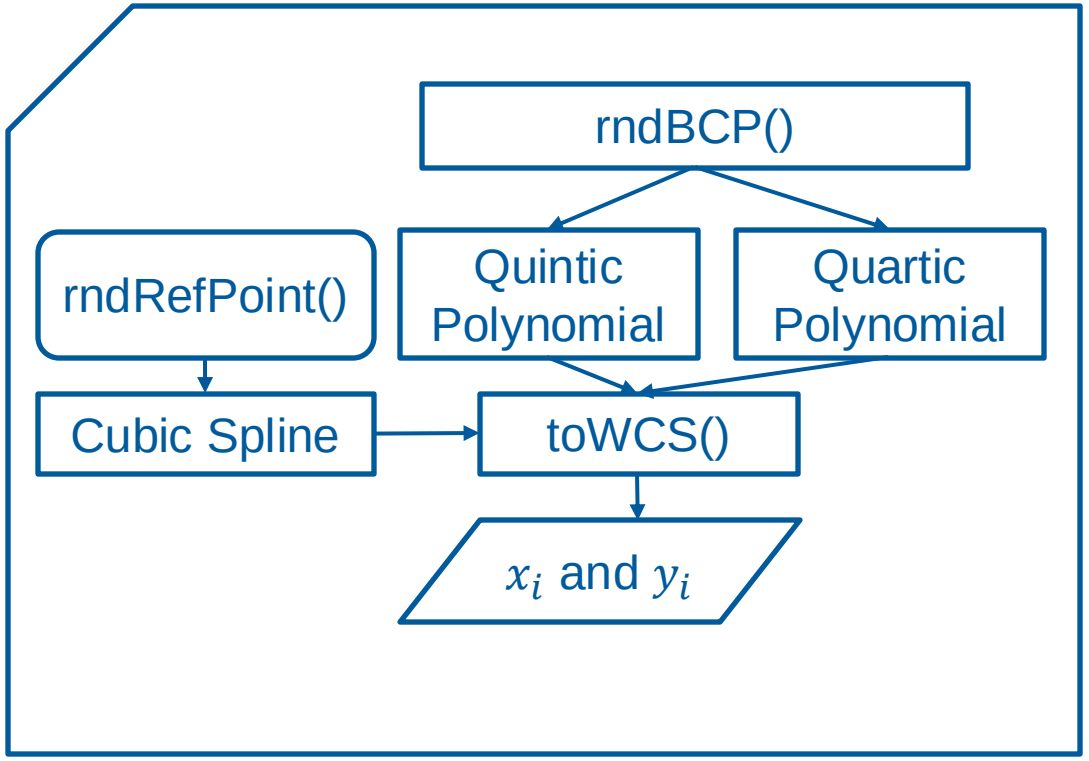
Boundary conditions:

- $d_s, d_e \in [-10,10]$ m
- $v_s, v_e \in [-40,40]$ m/s
- $a_s, a_e \in [-20,20]$ m/s²

Physical limit of sports cars

Experiment

Frenet Optimal Planner

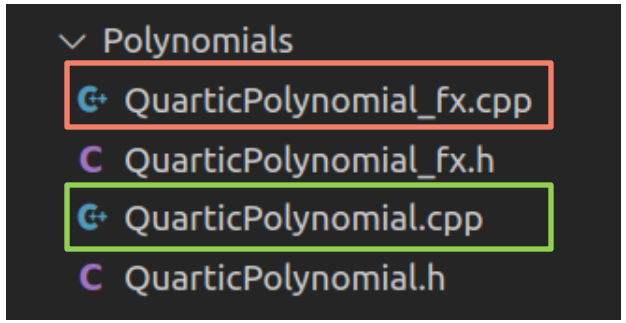


Boundary conditions:

- *Trajectories within 50 meters and 3 seconds*

Experiment

Implementation




```cpp
float QuarticPolynomial::calc_point(float t) {
    return a0 + a1 * t + a2 * pow(t, 2) + a3 * pow(t, 3) + a4 * pow(t, 4);
}
```

```cpp
fp_type QuarticPolynomial_fx::calc_point(fp_type t) {
    fp_type ans=a3+t*a4;
    ans=ans*t+a2;
    ans=ans*t+a1;
    ans=ans*t+a0;
    return ans;
}
```

```cpp
#define Standard_precision 15
typedef cnl::scaled_integer<int64_t, cnl::power<-Standard_precision>>  fp_type;
```

https://johnmcfarlane.github.io/cnl/

Experiment

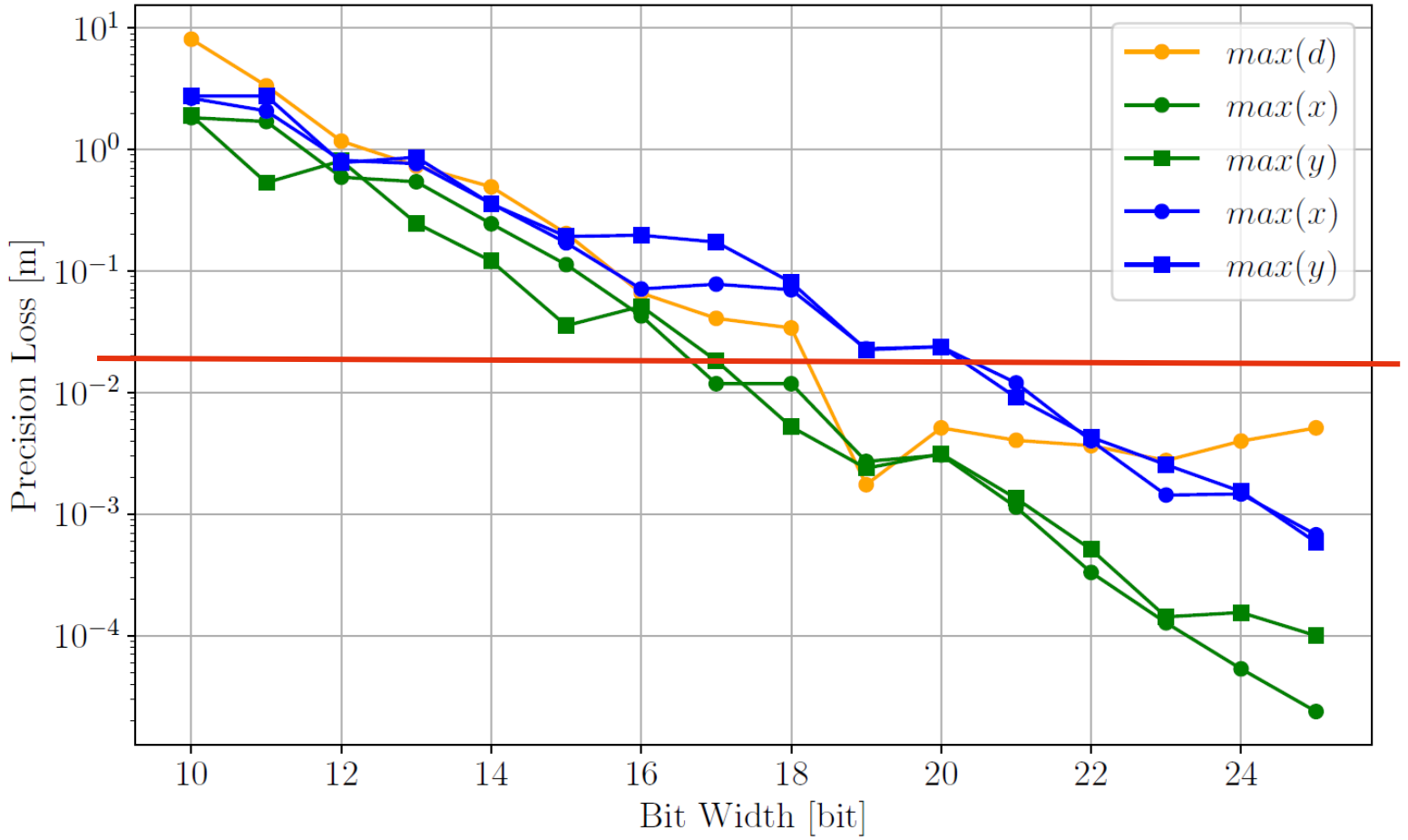
Implementation

```
fp_type QuinticPolynomial_fx::calc_point_fx(fp_type t) {
    fp_type ans = a4+t*a5;
    #ifdef USE_RECORDER
        Recorder::getInstance()->saveData<float>("QuinticPolynomial_fx::calc_point_fx::ans", static_cast<float>(ans));
    #endif
    ans = a3+t*ans;
    #ifdef USE_RECORDER
        Recorder::getInstance()->saveData<float>("QuinticPolynomial_fx::calc_point_fx::ans", static_cast<float>(ans));
    #endif
    ans = a2+t*ans;
    #ifdef USE_RECORDER
        Recorder::getInstance()->saveData<float>("QuinticPolynomial_fx::calc_point_fx::ans", static_cast<float>(ans));
    #endif
    ans = a1+t*ans;
    #ifdef USE_RECORDER
        Recorder::getInstance()->saveData<float>("QuinticPolynomial_fx::calc_point_fx::ans", static_cast<float>(ans));
    #endif
    ans = a0+t*ans;
    #ifdef USE_RECORDER
        Recorder::getInstance()->saveData<float>("QuinticPolynomial_fx::calc_point_fx::ans", static_cast<float>(ans));
    #endif
    return ans;
}
```

Record data

Results

Bit Width For Fraction

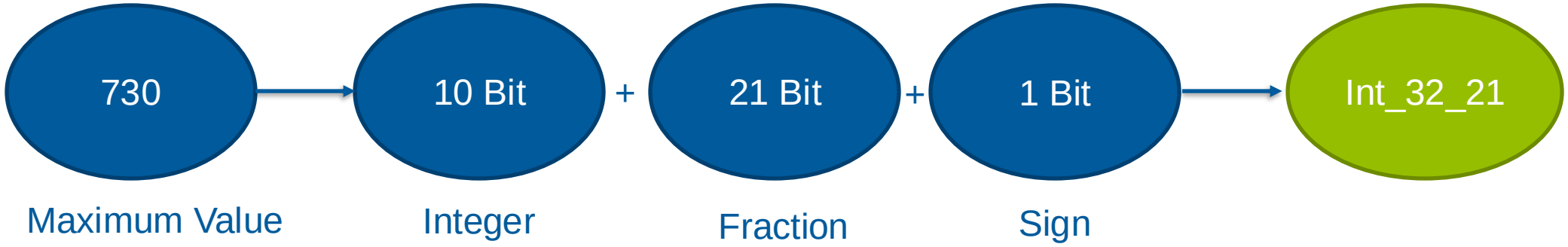


Why so many bits is needed ?

Results

Bit Width For Integer

$$d'''(t) = 4a_3 + 24a_4t + 60a_5t^2$$



Summary

This study

- explores the applicability of fixed-point arithmetic for trajectory planning
- analyzes how the fractional bit width of fixed-point numbers affects precision loss in trajectory planning
- propose to use Thomas algorithm to decouple the dependency of external matrix-library
- propose to use int_32_21 fixed-point representation to maintain a precision loss within 2 cm for a 50 meters-trajectory within 3 seconds