Supplementary Material: Space-Partitioning RANSAC

Daniel Barath[1] and Gabor Valasek[2]

[1] ETH Zürich, Computer Vision and Geometry Group, Switzerland
danielbela.barath@inf.ethz.ch
[2] Eötvös Loránd University, Budapest, Hungary
valasek@inf.elte.hu

1 Polynomial Approximation

This section gives a brief overview of three classic polynomial approximation schemes that we experimented with to bound the range of nonlinear transformations. The derivation of the results below are detailed in standard books on numerical analysis, we list these here for implementation reference and to make our paper self-contained. Similarly, we briefly summarize polynomial basis conversion via fitting for the sake of convenience.

The literature of polynomial approximations is rich, our selection of Taylor, Hermite, and Lagrange interpolation was motivated by ease of implementation and the existence and conciseness of error terms for these polynomials when used in the context of function approximation.

In terms of convenience, Lagrange interpolation in Bernstein basis is the least obtrusive solution as it only requires the evaluation of the target function. Hermite and Taylor expansions require higher order derivatives, which have to be either computed formally, via automatic differentiation, or numerical differentiation. However, all solutions require the ability to bound the magnitude of certain derivatives, if conservative bounds are to be computed.

1.1 Taylor Approximation

We denote the n-dimensional Euclidean space by $\mathbb{R}^n$ and $\|\cdot\|_2$ is the Euclidean norm. The partial derivatives of an $f : \mathbb{R}^2 \to \mathbb{R}$ function are $\partial_1 f, \partial_2 f$ or f_x, f_y.The scalar product of vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ is written as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^\mathrm{T}\boldsymbol{b}$.

Definition 1. *Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ be a multi-index. Then we define the following operations:*

- $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + \cdots + \alpha_n$
- $\boldsymbol{\alpha}! = \alpha_1! \cdot \alpha_2! \cdot \ldots \cdot \alpha_n!$, *where* $0! = 1$
- $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot \ldots \cdot x_n^{\alpha_n} \qquad (\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n)$
- $\partial^{\boldsymbol{\alpha}} f = \partial_1^{\alpha_1} \partial_2^{\alpha_2} \ldots \partial_n^{\alpha_n} f \qquad (f : \mathbb{R}^n \to \mathbb{R})$

Definition 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be $f \in C^{k+1}$. The degree k multivariate Taylor approximation of f about $\boldsymbol{x}_0$ is*

$$T_{k,\boldsymbol{x}_0}(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}| \le k} \frac{\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x}_0)}{\boldsymbol{\alpha}!} (\boldsymbol{x} - \boldsymbol{x}_0)^{\boldsymbol{\alpha}} . \tag{1}$$

Theorem 1 (Taylor Approximation Theorem). *Let $f : \mathbb{R}^n \to \mathbb{R}, S \subset \mathbb{R}^n$ open and convex, $f \in C^{k+1}[S]$. If $\boldsymbol{a}, \boldsymbol{a} + \boldsymbol{h} \in S$, then*

$$f(\boldsymbol{a} + \boldsymbol{h}) = T_f^{(k)}(\boldsymbol{a} + \boldsymbol{h}) + R_{\boldsymbol{a},k}(\boldsymbol{h}) \tag{2}$$

where the residual $R_{\boldsymbol{a},k}$ can be expressed using an adequate $c \in (0,1)$:

$$R_{\boldsymbol{a},k}(\boldsymbol{h}) = \sum_{|\boldsymbol{\alpha}| = k+1} \partial^{\boldsymbol{\alpha}} f(\boldsymbol{a} + c \cdot \boldsymbol{h}) \frac{\boldsymbol{h}^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} \tag{3}$$

or, with an integral form, as

$$R_{\boldsymbol{a},k}(\boldsymbol{h}) = (k+1) \sum_{|\boldsymbol{\alpha}| = k+1} \frac{\boldsymbol{h}^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} \int_0^1 (1-t)^k \partial^{\boldsymbol{\alpha}} f(\boldsymbol{a} + t\boldsymbol{h}) dt . \tag{4}$$

Corollary 1 (Error bound of Taylor approximation). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be such that $f \in C^{k+1}[S]$ and $M > 0$ such that $\forall \boldsymbol{x} \in S : \forall |\boldsymbol{\alpha}| = k+1 : |\partial^{\boldsymbol{\alpha}} f(\boldsymbol{x})| \le M$. Then*

$$R_{\boldsymbol{a},k}(\boldsymbol{h}) \le \frac{M}{(k+1)!} ||\boldsymbol{h}||_1^{k+1} . \tag{5}$$

In the two-dimensional case, the Taylor polynomials are written as

$$T_f^{(k)}(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{i} \frac{\partial_1^j \partial_2^{i-j} f(a,b)}{j!(i-j)!} (x-a)^j (y-b)^{i-j} \tag{6}$$

In the case of vector valued functions of two variables, the Taylor expansion naturally generalizes to

$$\boldsymbol{T}_{\boldsymbol{f}}^{(k)}(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{i} \frac{\partial_1^j \partial_2^{i-j} \boldsymbol{f}(a,b)}{j!(i-j)!} (x-a)^j (y-b)^{i-j} \in \mathbb{R}^n \tag{7}$$

1.2 Hermite Interpolation

Definition 3 (Hermite interpolant). *Let $f : \mathbb{R} \to \mathbb{R}$ be such that $f \in C^{k+1}$. The polynomial $h_k(x)$ of degree $2k+1$ is an order k Hermite interpolant at $a, b \in \mathbb{R}$ if and only if*

$$h_k^{(i)}(a) = f^{(i)}(a) \tag{8}$$

$$h_k^{(i)}(b) = f^{(i)}(b) \tag{9}$$

holds, $i = 0, \ldots, k$ and $f^{(i)}$ denotes the i-th derivative.

The above two-point Hermite interpolation is sometimes described as dense in the sense that all derivatives and function values are prescribed up to a fixed order and there are no gaps, that is, missing derivatives. It can be easily seen that the Hermite interpolation polynomial is unique. More importantly, its error characteristics are given by

Theorem 2 (Error bound of Hermite interpolation). *Let $f : \mathbb{R} \to \mathbb{R}$ be such that $f \in C^{k+1}$ and let $h_k(x)$ be an order k Hermite interpolant at $a, b \in \mathbb{R}$. Then for all $x \in [a, b]$ exists a $\xi \in [a, b]$ such that*

$$f(x) - h_k(x) = \frac{f^{(k+1)}(\xi)}{(k+1)!}(x-a)^{k+1}(x-b)^{k+1} \, . \tag{10}$$

Oftentimes, it is more convenient to bound the above as a function of the $b-a$ width of the domain. The maximum of the function is attained at the midpoint of the interval and straightforward substitution gives the resulting modified bound. The above holds for vector valued functions as well but similarly to the Taylor case, the 1-norm has to be used.

In our case, we approximate the image of the cell boundary curves, thus the single variable error term is sufficient.

1.3 Lagrange Interpolation

By Lagrange interpolation we refer to the interpolation of a $f : [a, b] \to \mathbb{R}, [a, b] \subset \mathbb{R}$ function at some prescribed $a = x_0 < x_1 < \cdots < x_k = b$ points by polynomials. Then the following holds

Theorem 3. *If $p_k(x)$ is a polynomial that interpolates $f : [a, b] \to \mathbb{R}, [a, b]$ at $a = x_0 < x_1 < \cdots < x_k = b$ and $f \in C^{k+1}[a, b]$, then for any $x \in [a, b]$, there exists a $\xi \in (a, b)$ such that the following holds:*

$$f(x) - p_k(x) = (x - x_0) \cdot \dots \cdot (x - x_k) \frac{f^{(k+1)}(\xi)}{(n+1)!} \, . \tag{11}$$

There are ways to re-phrase the above in terms of differences, should the target function not meet the continuity assumptions of the theorem but we did not experiment with the practical applicability of these.

As we have no control over the magnitude of $f^{(k+1)}(\xi)$, the only way to minimize the error in (11) is to find $x_i \in [a, b]$ nodes that minimize $\Pi_{i=0}^{k}(x - x_k)$ over $[a, b]$.

Definition 4 (Chebyshev polynomials). *The Chebyshev polynomials over $[-1, 1]$ are defined recursively as*

$$T_0(x) = 1, \tag{12}$$
$$T_1(x) = x, \tag{13}$$
$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \tag{14}$$

for $k \geq 1$.

Theorem 4 (Interpolation at Chebyshev nodes). *If $p_k(x)$ is a polynomial that interpolates $f : [-1,1] \to \mathbb{R}, f \in C^{k+1}[a,b]$ at the roots of $T_{k+1}(x)$, that is, $x_i = \cos\left(\frac{2i+1}{2k+2}\right)$, $i = 0,1,\ldots,k$, then*

$$|f(x) - p_k(x)| \leq \frac{1}{2^k(k+1)!} \max_{t\in[-1,1]} \left| f^{(k+1)}(t) \right| . \tag{15}$$

This is the best upper bound if we can only vary the location of the x_i interpolation nodes.

If the function is defined over an arbitrary $[a,b]$ interval, the Chebyshev nodes simply have to be affinely mapped from $[-1,1]$ to $[a,b]$ to compute the necessary Chebyshev nodes as follows:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2i+1}{2k+2}\right) . \tag{16}$$

2 Bounding Polynomials

2.1 Properties of Bézier Curves

Let $\boldsymbol{b}_i \in \mathbb{R}^d, (\ i = 0,1,\ldots,n)$ denote the control points of a d-dimensional Bézier curve. The parametric equation of the curve is

$$\boldsymbol{b}(t) = \sum_{i=0}^{n} \boldsymbol{b}_i B_i^n(t) \ , \ \ t \in [0,1] \ , \tag{17}$$

where $B_i^n(t)$ are the Bernstein polynomials over $[0,1]$, *i.e.*

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \ . \tag{18}$$

As the Bernstein basis is positive and forms a partition of unity (i.e. $B_i^n(t) \geq 0, t \in [0,1]$ and $\sum_{i=0}^n B_i^n(t) = 1$), it follows that all points of the curve are contained within the convex hull of its $\boldsymbol{b}_i$ control points. Consequently, the axis aligned bounding box of the control points is a conservative bound on the range of the curve.

Similarly, if we want to bound the magnitude of a $\boldsymbol{f} : \mathbb{R} \to \mathbb{R}^n$ vector valued function, we can construct a Bézier approximation to the $\|\boldsymbol{f}(x_i)\|$ magnitude values (in arbitrary norm) via interpolation and use the value of the largest control point (here, scalar) to infer an approximate upper bound on the magnitude.

2.2 Interpolating Data

Recall that the evaluation of a function in a basis such as in Equation (17) can be written in matrix form as

$$\boldsymbol{b}(t) = [B_0^n(t), B_1^n(t), \ldots, B_n^n(t)] \cdot \begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \ldots \\ \boldsymbol{b}_n \end{bmatrix} \tag{19}$$

As such, when given $n+1$ parameter values $t_0 < t_1 < \cdots < t_n$ and corresponding points in space $\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$, we can compute the $\boldsymbol{b}_i, (i = 0, 1, \ldots, n)$ Bézier control points that interpolate them by solving

$$\begin{bmatrix} B_0^n(t_0) & B_1^n(t) & \ldots & B_n^n(t) \\ B_0^n(t_1) & B_1^n(t) & \ldots & B_n^n(t) \\ \ldots & \ldots & \ldots & \ldots \\ B_0^n(t_n) & B_1^n(t_n) & \ldots & B_n^n(t_n) \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \ldots \\ \boldsymbol{b}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_0 \\ \boldsymbol{p}_1 \\ \ldots \\ \boldsymbol{p}_n \end{bmatrix} \tag{20}$$

for $[\boldsymbol{b}_0, \ldots, \boldsymbol{b}_n]^T$. One can either use a linear solver for better robustness, or use interpolation nodes that yield a small condition number for the matrix on the left. Chebyshev nodes are such a choice, up to moderate degrees (that is, up to 10), making direct inversion possible which reduces the interpolation problem to a simple matrix-vector multiplication. Note that if $t_0 = 0, t_n = 1$, the first and the last rows of the matrix are $\boldsymbol{e}_1, \boldsymbol{e}_{n+1}$ respectively, where $\boldsymbol{e}_i$ are the canonical basis vectors of dimension $n+1$.

In our tests on Lagrange interpolation, we used the roots of the Chebyshev polynomials over closed intervals, i.e. $t_i = \frac{1}{2}\left(1 - \cos(\frac{i\pi}{n})\right) \in [0, 1], i = 0, \ldots, n$.

2.3 Converting Hermite to Bézier Control Data

Since we approximate our mapped boundary curves from endpoint derivative data, *i.e.* we use Hermite interpolation, we have to convert the Hermite basis polynomial data to Bernstein basis. This can be done by brute-force interpolation, as in evaluating the Hermite polynomial in $n+1$ points and multiplying the resulting vector by the inverse of the Bernstein evaluation matrix of at the sample parameters, as shown in the previous subsection.

A simpler approach is possible, however, by recalling that the derivatives of Bézier curves at the endpoints are

$$\boldsymbol{b}^{(k)}(0) = \frac{n!}{(n-k)!} \Delta^k \boldsymbol{b}_0 \tag{21}$$

$$\boldsymbol{b}^{(k)}(1) = \frac{n!}{(n-k)!} \Delta^k \boldsymbol{b}_{n-k} \tag{22}$$

where the Δ forward differences are defined as

$$\Delta^j \boldsymbol{b}_i = \Delta^{j-1} \boldsymbol{b}_{i+1} - \Delta^{j-1} \boldsymbol{b}_i \tag{23}$$

for $j = 1, 2, \ldots$ and $\Delta^0 \boldsymbol{b}_i = \boldsymbol{b}_{i+1} - \boldsymbol{b}_i$. These allow us to compute the control points directly from the raw derivatives.

Let $\boldsymbol{m}_i^{(k)}, i = 0, 1$ denote the appropriate k-th directional derivatives at the two endpoints of the boundary curve. Then from requiring

$$\boldsymbol{m}_i^{(k)} = \boldsymbol{b}^{(k)}(i) \quad , \ (i = 0, 1) \tag{24}$$

to hold, we have

$$\boldsymbol{m}_0^{(k)} = \frac{n!}{(n-k)!} \Delta^k \boldsymbol{b}_0 \tag{25}$$

$$\boldsymbol{m}_1^{(k)} = \frac{n!}{(n-k)!} \Delta^k \boldsymbol{b}_{n-k} \tag{26}$$

This allows us to progressively compute the control points from the derivatives such that the resulting curve will reconstruct them at the endpoints.

For example, the first three derivatives at $t = 0$ determine the $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$ control points from

$$\boldsymbol{m}_0^{(1)} = n(\boldsymbol{b}_1 - \boldsymbol{b}_0) \tag{27}$$

$$\boldsymbol{m}_0^{(2)} = n(n-1)(\boldsymbol{b}_2 - \boldsymbol{b}_1 - \Delta \boldsymbol{b}_0) \tag{28}$$

$$\boldsymbol{m}_0^{(3)} = n(n-1)(n-2)(\boldsymbol{b}_3 - \boldsymbol{b}_2 - \Delta \boldsymbol{b}_1 - \Delta^2 \boldsymbol{b}_0) \tag{29}$$

as

$$\boldsymbol{b}_1 = \frac{\boldsymbol{m}_0^{(1)}}{n} + \boldsymbol{b}_0 \tag{30}$$

$$\boldsymbol{b}_2 = \frac{\boldsymbol{m}_0^{(2)}}{n(n-1)} + \Delta \boldsymbol{b}_0 + \boldsymbol{b}_1 \tag{31}$$

$$\boldsymbol{b}_3 = \frac{\boldsymbol{m}_0^{(3)}}{n(n-1)(n-2)} + \Delta^2 \boldsymbol{b}_0 + \Delta \boldsymbol{b}_1 + \boldsymbol{b}_2 \tag{32}$$

and at $t = 1$ endpoint from

$$\boldsymbol{m}_1^{(1)} = n(\boldsymbol{b}_n - \boldsymbol{b}_{n-1}) \tag{33}$$

$$\boldsymbol{m}_1^{(2)} = n(n-1)(\Delta \boldsymbol{b}_{n-1} - \boldsymbol{b}_{n-1} + \boldsymbol{b}_{n-2}) \tag{34}$$

$$\begin{aligned} \boldsymbol{m}_1^{(3)} &= n(n-1)(n-2)(\Delta^2 \boldsymbol{b}_{n-2} - \Delta^2 \boldsymbol{b}_{n-3}) \\ &= n(n-1)(n-2)(\Delta^2 \boldsymbol{b}_{n-2} - \Delta \boldsymbol{b}_{n-2} + \boldsymbol{b}_{n-2} - \boldsymbol{b}_{n-3}) \end{aligned} \tag{35}$$

as

$$\boldsymbol{b}_{n-1} = \boldsymbol{b}_n - \frac{\boldsymbol{m}_1^{(1)}}{n} \tag{36}$$

$$\boldsymbol{b}_{n-2} = \boldsymbol{b}_{n-1} - \Delta \boldsymbol{b}_{n-1} + \frac{\boldsymbol{m}_1^{(2)}}{n(n-1)} \tag{37}$$

$$\boldsymbol{b}_{n-3} = \boldsymbol{b}_{n-2} - \Delta \boldsymbol{b}_{n-2} + \Delta^2 \boldsymbol{b}_{n-2} - \frac{\boldsymbol{m}_1^{(3)}}{n(n-1)(n-2)} \tag{38}$$