Advances in Applied Mathematics 应用数学进展, 2025, 14(9), 5-14
Published Online September 2025 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2025.149394

Hans汉斯

三次Bézier曲线的快速求值算法

贺思源，向仁婧

东华理工大学理学院，江西 南昌

收稿日期：2025年7月29日；录用日期：2025年8月22日；发布日期：2025年9月2日

摘　要

三次Bézier曲线的求值算法对于曲线的处理有重要的意义，为了提高三次Bézier曲线的计算效率，本文从矩阵分解的角度解读了De Casteljau算法，并在此基础上提出了一种新的快速求值算法。该算法利用线性插值计算曲线上的点，除第一次线性插值利用两个初始控制顶点外，其余每次线性插值仅利用一个初始控制顶点与上一次线性插值产生的一个新控制顶点，将中间控制顶点数量从传统算法的6个减少至3个，其仅使用控制顶点的凸组合，计算复杂度与控制顶点数量呈线性关系，从而提高了求值算法的计算效率。新算法只涉及线性插值，具有数值稳定性且易于实现，可以更快地求出曲线在特定参数值下的点，为三次Bézier曲线的实时绘制提供了新的解决方案。

关键词

Bézier曲线，De Casteljau算法，求值算法，线性复杂度

Fast Evaluation Algorithm for Cubic Bézier Curves

Siyuan He, Renjing Xiang

School of Science, East China University of Technology, Nanchang Jiangxi

Received: Jul. 29th, 2025; accepted: Aug. 22nd, 2025; published: Sep. 2nd, 2025

Abstract

The evaluation algorithm of cubic Bézier curves is of great significance for the processing of curves. In order to improve the computational efficiency of cubic Bézier curves, this paper interprets the De Casteljau algorithm from the perspective of matrix factorization, and proposes a new fast evaluation algorithm based on this. The algorithm uses linear interpolation to calculate the points on the curve. Except that the first linear interpolation uses two initial control vertices, the remaining linear interpolation only uses one initial control vertex and a new control vertex generated by the

文章引用: 贺思源, 向仁婧. 三次 Bézier 曲线的快速求值算法[J]. 应用数学进展, 2025, 14(9): 5-14.
DOI: 10.12677/aam.2025.149394

previous linear interpolation, which reduces the number of intermediate control vertices from 6 to 3. It only uses the convex combination of control points, and the computational complexity is linear with the number of control points, which improves the computational efficiency of the evaluation algorithm. The new algorithm only involves linear interpolation, which is numerically stable and easy to implement. It can quickly find the points of the curve under specific parameter values, and provides a new solution for real-time rendering of cubic Bézier curves.

Keywords

Bézier Curve, De Casteljau Algorithm, Evaluation Algorithm, Linear Complexity

1. 引言

计算机辅助几何设计(CAGD)是随着航空、造船、机械设计和制造等现代工业的蓬勃发展与计算机的出现而发生与发展起来的一门新兴交叉学科，曲线曲面的表示和逼近是 CAGD 的重要研究内容。Bézier 曲线曲面是 CAD/CAM 系统中广泛使用的造型工具，任意形状曲线曲面的 Bézier 表示与逼近在 CAGD 中有着重要的应用价值。

自 1962 年 Bézier 提出以逼近为基础的参数曲线曲面设计方法以来，该方法得到了广泛的研究和应用。国内关于 Bézier 方法的早期成果包括刘鼎元[1]的论文，介绍了三次 Bézier 曲线由端点处两条切向量唯一确定；1998 年黄有度[2]给出了一种逐点生成 Bézier 曲线的算法，其只用到整数加减法，有极高的效率；随后郑文明[3]等人通过对 Bézier 曲线的控制多边形作进一步细分，得到了一种生成 Bézier 曲线的细分算法；几年后，马月德[4]等人对简单割角法、升阶法、De Casteljau 方法进行了有效性分析与比较，得出 De Casteljau 算法是最佳的 Bézier 曲线几何作图法。国外研究的早期成果包括 Gordon、Riesenfeld [5]等人深入研究了 Bézier 方法，揭示了其与 Bernstein 多项式的联系，从而使 Bézier 方法具有更坚实的理论基础；随后，Wolfgang Boehm、Andreas Müller [6]等人的论文介绍了 De Casteljau 算法的有效性；2000 年 Huynh Ngoc Phien [7]指出 De Casteljau 算法虽然简单、稳定，但其复杂度为 $O\left(n^2\right)$ (n 为曲线次数)，因此效率并不高，且给出了根据 Bézier 曲线与两条广义 Ball 曲线之间的关系提出了三种更有效的算法；同时期，Peters J [8]给出了两种新算法对 Bernstein-Bézier 形式的 m 元全次数多项式的点网格的求值，并与 De Casteljau 算法进行了比较；随后 Licio Hernanes Bezerra [9]-[11]在 2011 年左右提出了计算 Bézier 曲线的新方法，其利用快速 Fourier 变换或快速 Pascal 矩阵向量乘法(由 Xiang Wang 和 Zhou Jituan [12]提出)与多项式计算策略相结合，生成 Bézier 曲线，运算速度得到有效提高；近几年，Paweł Woźny 和 Filip Chudy [13]提出了一种计算多项式或有理 Bézier 曲线上的点的新算法，该方法具有几何意义并且仅使用控制点的凸组合，相对于传统的算法，新算法的计算复杂度与控制顶点的数量呈线性关系；2024 年，Vijay 与 Gurunathan Saravana Kumar [14]等人探讨了在二维和三维环境中生成结构与 Bézier 曲线类似的曲线的各种技术。

三次 Bézier 曲线以其平滑性、可控性、计算高效性和广泛适用性在现代计算机图形学中占据着重要地位。它不仅在二维图形设计、三维建模、动画制作、字体设计等领域展现出卓越的性能，还在不断推动着这些领域的发展。随着计算机图形学技术的不断进步，三次 Bézier 曲线的应用前景将更加广阔。三

次 Bézier 曲线在 CAGD 中应用时，需要利用求值算法绘制和编辑复杂的曲线和形状，通过调整相应的控制顶点生成平滑的曲线，因此提供简单、高效的求值算法对于三次 Bézier 曲线处理具有重要的意义，特别是在一些复杂场景中，高效且高精度的三次 Bézier 曲线求值算法能够快速生成平滑的曲线，满足实时性要求。但随着应用场景的复杂化，生成三次 Bézier 曲线的各种算法也存在对初始控制顶点利用不够充分和计算效率、精度仍有待提高等不足，无法快速地计算出三次 Bézier 曲线。本文从矩阵分解的角度出发，利用线性插值的方法，针对三次 Bézier 曲线提出一种计算复杂度为 $O(n)$ (n 为曲线次数)的效率更高的算法，以达到快速求值的目的。

2. De Casteljau 算法解读

2.1. 三次 Bézier 曲线的求值算法

定义：三次 Bézier 曲线表达式为 $\boldsymbol{p}_3(t)=\sum_{i=0}^{3}\boldsymbol{V}_i B_i^3(t)$ ($i=0,1,2,3$)，其中 $\boldsymbol{V}_i$ 为控制顶点，$B_i^3(t)=C_3^i t^i(1-t)^{3-i}$ ($i=0,1,2,3$)为 Bernstein 基函数。

三次 Bézier 曲线具有如下性质：

1) 对称性：对于给定的控制多边形 $\boldsymbol{V}_0\boldsymbol{V}_1\boldsymbol{V}_2\boldsymbol{V}_3$，可以从 $\boldsymbol{V}_0$ 出发构造 Bézier 曲线，也可以从 $\boldsymbol{V}_3$ 开始反向构造曲线。这两条曲线形状完全相同但参数化方向相反。

2) 凸包性：Bézier 曲线完全被包容在由控制顶点形成的凸包内。

De Casteljau 算法通过重复线性插值来计算 Bézier 曲线。具体来说，它使用控制顶点之间的线性插值来逐步逼近曲线上的点。对于三次 Bézier 曲线，共需递归 3 次，其 De Casteljau 算法的递归关系式为：

$$\boldsymbol{V}_i^r(t)=(1-t)\boldsymbol{V}_i^{r-1}(t)+t\boldsymbol{V}_{i+1}^{r-1}(t) \tag{1}$$

其中 $r=1,2,3$，i 从 1 到 $3-r$，初始条件为 $\boldsymbol{V}_i^0(t)=\boldsymbol{V}_i$ ($i=0,1,2,3$)，r 表示递归的层数，$t\in[0,1]$ 是参数，且 $\boldsymbol{p}_3(t)=\boldsymbol{V}_0^3(t)$。

2.2. 算法解读

注：为书写方便，在不至于指代不明确时，下文省略自变量记号“t”。

记

$$\boldsymbol{V}=\begin{pmatrix}\boldsymbol{V}_0\\ \boldsymbol{V}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix},\quad \boldsymbol{B}=\begin{pmatrix}B_0^3 & B_1^3 & B_2^3 & B_3^3\end{pmatrix},$$

则对于三次 Bézier 曲线，可将 $\boldsymbol{p}_3(t)=\sum_{i=0}^{3}\boldsymbol{V}_i B_i^3(t)$ 写成矩阵形式，如下：

$$\boldsymbol{p}_3(t)=\boldsymbol{B}\boldsymbol{V},$$

令

$$\boldsymbol{M}_3=\begin{pmatrix}1-t & t\end{pmatrix},\quad \boldsymbol{M}_2=\begin{pmatrix}1-t & t & 0\\ 0 & 1-t & t\end{pmatrix},\quad \boldsymbol{M}_1=\begin{pmatrix}1-t & t & 0 & 0\\ 0 & 1-t & t & 0\\ 0 & 0 & 1-t & t\end{pmatrix},$$

有

$$\boldsymbol{B}=\boldsymbol{M}_3\boldsymbol{M}_2\boldsymbol{M}_1,$$

又有

$$\boldsymbol{M}_1\begin{pmatrix}\boldsymbol{V}_0\\ \boldsymbol{V}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix}=\begin{pmatrix}\boldsymbol{V}_0^1\\ \boldsymbol{V}_1^1\\ \boldsymbol{V}_2^1\end{pmatrix},\quad \boldsymbol{M}_2\begin{pmatrix}\boldsymbol{V}_0^1\\ \boldsymbol{V}_1^1\\ \boldsymbol{V}_2^1\end{pmatrix}=\begin{pmatrix}\boldsymbol{V}_0^2\\ \boldsymbol{V}_1^2\end{pmatrix},\quad \boldsymbol{M}_3\begin{pmatrix}\boldsymbol{V}_0^2\\ \boldsymbol{V}_1^2\end{pmatrix}=\boldsymbol{V}_0^3,$$

则有

$$\boldsymbol{p}_3(t)=\boldsymbol{BV}=\boldsymbol{M}_3\boldsymbol{M}_2\boldsymbol{M}_1\begin{pmatrix}\boldsymbol{V}_0\\ \boldsymbol{V}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix}=\boldsymbol{M}_3\boldsymbol{M}_2\begin{pmatrix}\boldsymbol{V}_0^1\\ \boldsymbol{V}_1^1\\ \boldsymbol{V}_2^1\end{pmatrix}=\boldsymbol{M}_3\begin{pmatrix}\boldsymbol{V}_0^2\\ \boldsymbol{V}_1^2\end{pmatrix}=\boldsymbol{V}_0^3. \tag{2}$$

由(2)可知，每进行一次矩阵乘法就是对曲线上特定的一系列控制顶点进行一次线性插值，一共进行三次矩阵乘法。第一次矩阵乘法对$\boldsymbol{V}_0$、$\boldsymbol{V}_1$、$\boldsymbol{V}_2$、$\boldsymbol{V}_3$进行线性插值，产生三个新控制顶点$\boldsymbol{V}_0^1$、$\boldsymbol{V}_1^1$、$\boldsymbol{V}_2^1$；第二次矩阵乘法对$\boldsymbol{V}_0^1$、$\boldsymbol{V}_1^1$、$\boldsymbol{V}_2^1$进行线性插值，产生两个新控制顶点$\boldsymbol{V}_0^2$、$\boldsymbol{V}_1^2$；第三次矩阵乘法对$\boldsymbol{V}_0^2$、$\boldsymbol{V}_1^2$进行线性插值，产生三次 Bézier 曲线在特定参数值t下的点$\boldsymbol{V}_0^3$。三次矩阵乘法共产生 6 个新控制顶点，除第一次插值利用初始控制顶点，其余每一次插值都完全利用上一次插值所产生的新控制顶点，计算效率偏低。

利用图可直观地表示 De Casteljau 算法，如图 1 所示：

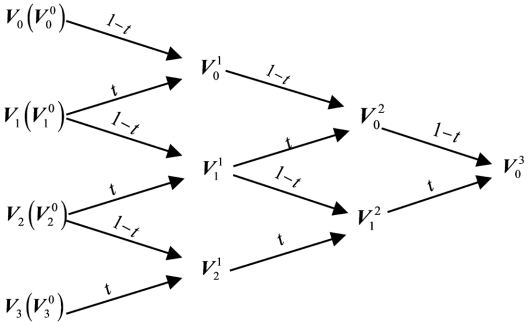

Figure 1. Principle of the De Casteljau algorithm
图 1. De Casteljau 算法原理

给定控制顶点$\boldsymbol{V}_0=(0,1)$，$\boldsymbol{V}_1=(0.3,0.2)$，$\boldsymbol{V}_2=(0.7,0.2)$，$\boldsymbol{V}_3=(1,1)$，利用 De Casteljau 算法生成$t=0.6$时三次 Bézier 曲线上对应点的递归过程如图 2 所示：

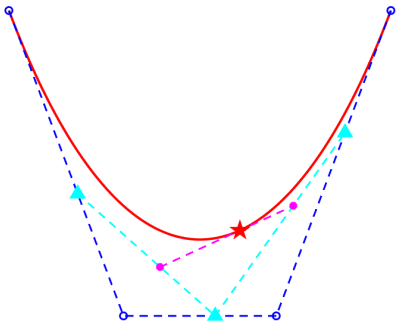

Figure 2. De Casteljau evaluation algorithm
图 2. De Casteljau 求值算法

图 2 中圆圈为初始控制顶点，三角形、实心圆、五角星表示取 $t=0.6$ 时算法产生的新控制点，圆圈从左往右依次为 $\boldsymbol{V}_0$、$\boldsymbol{V}_1$、$\boldsymbol{V}_2$、$\boldsymbol{V}_3$，三角形从左往右依次为 $\boldsymbol{V}_0^1$、$\boldsymbol{V}_1^1$、$\boldsymbol{V}_2^1$，实心圆从左往右依次为 $\boldsymbol{V}_0^2$、$\boldsymbol{V}_1^2$，五角星为 $\boldsymbol{V}_0^3$，且 $\boldsymbol{V}_0^3=\boldsymbol{p}_3(0.6)$。

在空间中生成三次 Bézier 曲线时，给定初始控制顶点 $\boldsymbol{V}_0=(0,1,0.7)$，$\boldsymbol{V}_1=(0.3,0.2,0.4)$，$\boldsymbol{V}_2=(0.7,0.2,0.6)$，$\boldsymbol{V}_3=(1,1,1)$，利用 De Casteljau 算法生成 $t=0.6$ 时三次 Bézier 曲线上对应点的递归过程如图 3 所示：

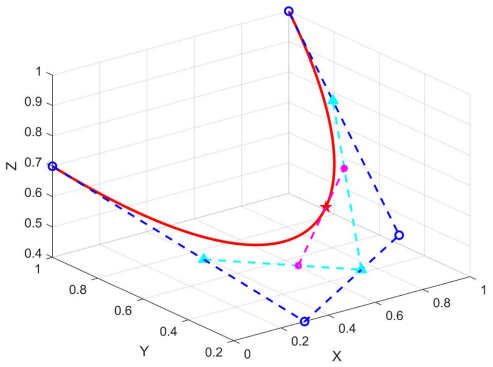

Figure 3. Three-dimensional example of De Casteljau evaluation algorithm
图 3. De Casteljau 求值算法三维示例

图 3 中圆圈、三角形、实心圆、五角星表示的控制顶点与图 2 相同。

3. 新的求值算法

3.1. 算法原理

定理：记

$$\begin{cases} h_i(t)=\dfrac{B_i^3(t)}{\sum_{j=0}^{i}B_j^3(t)},i=0,1,2,3 \\ \boldsymbol{Q}_i(t)=\left[1-h_i(t)\right]\boldsymbol{Q}_{i-1}(t)+h_i(t)\boldsymbol{V}_i,i=1,2,3 \end{cases} \tag{3}$$

初始条件为 $\boldsymbol{Q}_0(t)=\boldsymbol{V}_0$，参数 $t\in[0,1]$，则有 $\boldsymbol{p}_3(t)=\boldsymbol{Q}_3(t)$。

证明：由(3)知：

$$\begin{aligned} \boldsymbol{Q}_1&=(1-h_1)\boldsymbol{Q}_0+h_1\boldsymbol{V}_1=\frac{B_0^3}{B_0^3+B_1^3}\boldsymbol{Q}_0+\frac{B_1^3}{B_0^3+B_1^3}\boldsymbol{V}_1, \\ \boldsymbol{Q}_2&=(1-h_2)\boldsymbol{Q}_1+h_2\boldsymbol{V}_2 \\ &=\frac{B_0^3+B_1^3}{B_0^3+B_1^3+B_2^3}\boldsymbol{Q}_1+\frac{B_2^3}{B_0^3+B_1^3+B_2^3}\boldsymbol{V}_2 \\ &=\frac{B_0^3}{B_0^3+B_1^3+B_2^3}\boldsymbol{Q}_0+\frac{B_1^3}{B_0^3+B_1^3+B_2^3}\boldsymbol{V}_1+\frac{B_2^3}{B_0^3+B_1^3+B_2^3}\boldsymbol{V}_2, \\ \boldsymbol{Q}_3&=(1-h_3)\boldsymbol{Q}_2+h_3\boldsymbol{V}_3 \\ &=\left(B_0^3+B_1^3+B_2^3\right)\boldsymbol{Q}_2+B_3^3\boldsymbol{V}_3 \\ &=B_3^3\boldsymbol{V}_3+B_2^3\boldsymbol{V}_2+B_1^3\boldsymbol{V}_1+B_0^3\boldsymbol{Q}_0 \\ &=B_3^3\boldsymbol{V}_3+B_2^3\boldsymbol{V}_2+B_1^3\boldsymbol{V}_1+B_0^3\boldsymbol{V}_0 \\ &=\boldsymbol{p}_3(t). \end{aligned}$$

综上，$\boldsymbol{p}_3(t)=\boldsymbol{Q}_3(t)$，定理得证。

3.2. 算法解读

根据快速求值算法，也可将 $\boldsymbol{p}_3(t)=\sum_{i=0}^{3}\boldsymbol{V}_i B_i^3(t)$ 写成矩阵形式，如下：

$$\boldsymbol{p}_3(t)=\boldsymbol{B}\boldsymbol{V},$$

令

$$\boldsymbol{N}_3=\begin{pmatrix}1-h_3 & h_3\end{pmatrix},\ \boldsymbol{N}_2=\begin{pmatrix}1-h_2 & h_2 & 0\\ 0 & 0 & 1\end{pmatrix},\ \boldsymbol{N}_1=\begin{pmatrix}1-h_1 & h_1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix},$$

有

$$\boldsymbol{B}=\boldsymbol{N}_3\boldsymbol{N}_2\boldsymbol{N}_1,$$

又

$$\boldsymbol{N}_1\begin{pmatrix}\boldsymbol{V}_0\\ \boldsymbol{V}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix}=\begin{pmatrix}\boldsymbol{Q}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix},\quad \boldsymbol{N}_2\begin{pmatrix}\boldsymbol{Q}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix}=\begin{pmatrix}\boldsymbol{Q}_2\\ \boldsymbol{V}_3\end{pmatrix},\quad \boldsymbol{N}_3\begin{pmatrix}\boldsymbol{Q}_2\\ \boldsymbol{V}_3\end{pmatrix}=\boldsymbol{Q}_3,$$

则有

$$\boldsymbol{p}_3(t)=\boldsymbol{B}\boldsymbol{V}=\boldsymbol{N}_3\boldsymbol{N}_2\boldsymbol{N}_1\begin{pmatrix}\boldsymbol{V}_0\\ \boldsymbol{V}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix}=\boldsymbol{N}_3\boldsymbol{N}_2\begin{pmatrix}\boldsymbol{Q}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix}=\boldsymbol{N}_3\begin{pmatrix}\boldsymbol{Q}_2\\ \boldsymbol{V}_3\end{pmatrix}=\boldsymbol{Q}_3. \tag{4}$$

由(4)可知，每进行一次矩阵乘法就是对曲线上特定的一系列控制顶点进行一次线性插值，一共进行三次矩阵乘法。第一次矩阵乘法对 $\boldsymbol{V}_0$、$\boldsymbol{V}_1$ 进行线性插值，产生一个控制顶点 $\boldsymbol{Q}_1$；第二次矩阵乘法对 $\boldsymbol{Q}_1$、$\boldsymbol{V}_2$ 进行线性插值，产生一个新控制顶点 $\boldsymbol{Q}_2$；第三次矩阵乘法对 $\boldsymbol{Q}_2$、$\boldsymbol{V}_3$ 进行线性插值，产生三次 Bézier 曲线在特定参数值 t 下的点 $\boldsymbol{Q}_3$。三次矩阵乘法共产生 3 个新控制顶点，每一次插值实际上只利用了两个点，第一次线性插值利用两个初始控制顶点，其余两次线性插值利用的控制顶点包括一个初始的控制顶点和一个上次线性插值产生的新控制顶点。相较于 De Casteljau 求值算法，该算法减少了在线性插值过程中产生的控制顶点个数，且每一次线性插值都充分利用了原有的控制顶点，其复杂度为 $O(n)$ (n 为曲线次数)，在一定程度上提高了算法的计算效率。

利用图可直观地表示快速求值算法，如图 4 所示：

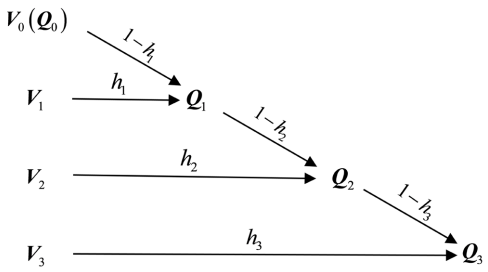

Figure 4. The principle of fast evaluation algorithm
图 4. 快速求值算法原理

给定与图 2 相同控制顶点，利用快速求值算法生成 $t=0.6$ 时三次 Bézier 曲线上对应点的递归过程如图 5 所示：

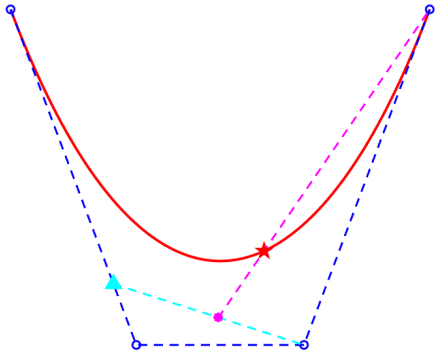

Figure 5. Fast evaluation algorithm
图 5. 快速求值算法

图 5 中圆圈为初始控制顶点，三角形、实心圆、五角星表示取 $t=0.6$ 时算法产生的新控制点，圆圈与图 2 相同，三角形为 $\boldsymbol{Q}_1$，实心圆为 $\boldsymbol{Q}_2$，五角星为 $\boldsymbol{Q}_3$，且 $\boldsymbol{Q}_3=\boldsymbol{p}_3(0.6)$。

当在空间中生成三次 Bézier 曲线时，给定与图 3 相同控制顶点利用快速求值算法生成 $t=0.6$ 时三次 Bézier 曲线上对应点的递归过程如图 6 所示：

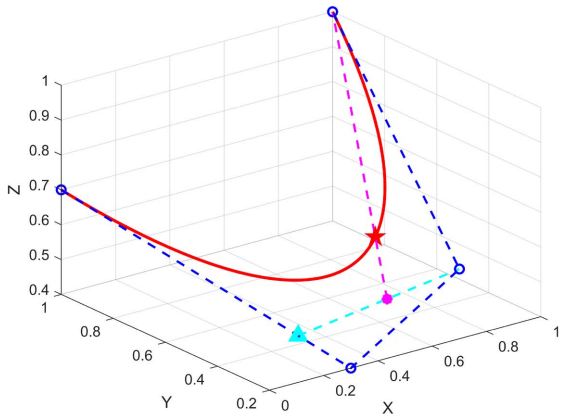

Figure 6. Three-dimensional example of fast evaluation algorithm
图 6. 快速求值算法三维示例

图 6 中圆圈、三角形、实心圆、五角星表示的控制顶点与图 5 相同。

由三次 Bézier 曲线的对称性可知：对于给定的控制多边形 $\boldsymbol{V}_0\boldsymbol{V}_1\boldsymbol{V}_2\boldsymbol{V}_3$，若从 $\boldsymbol{V}_3$ 开始构造曲线，得到的曲线形状与从 $\boldsymbol{V}_0$ 开始构造的曲线完全相同但参数化方向相反。故可知从 $\boldsymbol{V}_3$ 开始反向构造曲线的递推关系式如下：

$$\begin{cases} g_i(t)=\dfrac{B_{3-i}^3(t)}{\sum_{j=3-i}^{3}B_j^3(t)} \\ \boldsymbol{R}_i(t)=\left[1-g_i(t)\right]\boldsymbol{R}_{i-1}(t)+g_i(t)\boldsymbol{V}_{3-i} \end{cases}$$

其中，$i=0,1,2,3$，初始条件为 $\boldsymbol{R}_0(t)=\boldsymbol{V}_3$，利用递推关系可得可以得到曲线上与 $\boldsymbol{p}_3(t)$ 对应的点 $\boldsymbol{R}_3(t)$。

利用图可直观地表示反向快速求值算法，如图 7 所示：

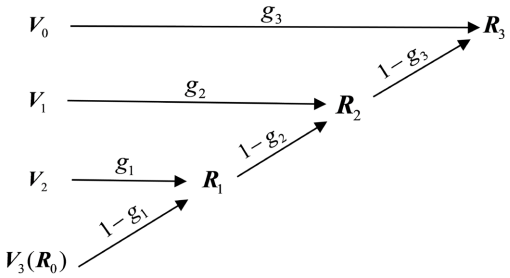

Figure 7. The principle of reverse fast evaluation algorithm
图 7. 反向快速求值算法原理

给定与图 2 相同控制顶点，利用反向快速求值算法生成 $t=0.6$ 时三次 Bézier 曲线上对应点的递归过程如图 8 所示：

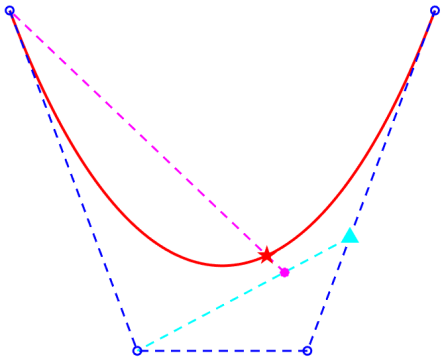

Figure 8. Reverse fast evaluation algorithm
图 8. 反向快速求值算法

图 8 中圆圈为初始控制顶点，三角形、实心圆、五角星表示取 $t=0.6$ 时算法产生的新控制点，圆圈与图 2 相同，三角形为 $\boldsymbol{R}_1$，实心圆为 $\boldsymbol{R}_2$，五角星为 $\boldsymbol{R}_3$，且 $\boldsymbol{R}_3=\boldsymbol{p}_3(0.6)$。

当在空间中反向生成三次 Bézier 曲线时，给定与图 3 相同控制顶点利用反向快速求值算法生成 $t=0.6$ 时三次 Bézier 曲线上对应点的递归过程如图 9 所示：

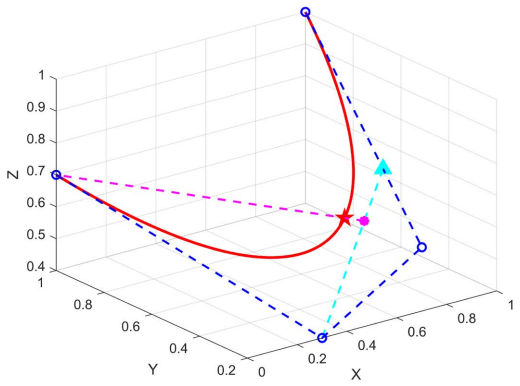

Figure 9. Three-dimensional example of reverse fast evaluation algorithm
图 9. 反向快速求值算法三维示例

图 9 中圆圈、三角形、实心圆、五角星表示的控制顶点与图 8 相同。

由凸包性可知：Bézier 曲线被 $\boldsymbol{Q}_3$ 分成两段，其中一段完全被包容在由控制多边形 $\boldsymbol{V}_0\boldsymbol{Q}_1\boldsymbol{Q}_2\boldsymbol{Q}_3$ 形成的凸包内。

4. n 次 Bézier 曲线快速求值算法及复杂度分析

4.1. n 次 Bézier 曲线的通用求值算法

定义：n 次 Bézier 曲线表达式为 $\boldsymbol{p}_n(t)=\sum_{i=0}^{n}\boldsymbol{V}_iB_i^n(t)$ $(i=1,2,\cdots,n)$，其中 $\boldsymbol{V}_i$ 为控制顶点，$B_i^n(t)=C_n^it^i(1-t)^{n-i}$ $(i=1,2,\cdots,n)$为 Bernstein 基函数。

定理：记

$$\begin{cases} h_i(t)=\dfrac{B_i^n(t)}{\sum_{j=0}^{i}B_j^n(t)},i=0,1,2,\cdots,n \\ \boldsymbol{Q}_i(t)=\left[1-h_i(t)\right]\boldsymbol{Q}_{i-1}(t)+h_i(t)\boldsymbol{V}_i,i=1,2,\cdots,n \end{cases}$$

初始条件为 $\boldsymbol{Q}_0(t)=\boldsymbol{V}_0$，参数 $t\in[0,1]$。则有 $\boldsymbol{p}_n(t)=\boldsymbol{Q}_n(t)$。定理的证明比较复杂，本文暂不概述。

4.2. 复杂度量化分析

设曲线次数为 n，单点求值所需基本运算量如下(见表 1)：

Table 1. The basic computation required for single point evaluation
表 1. 单点求值所需基本运算量

算法	浮点乘法次数	浮点加法次数	总运算量
De Casteljau	$n(n-1)/2$	$n(n-1)/2$	$O(n^2)$
新算法	$3n$	$3n$	$O(n)$

5. 总结

三次 Bézier 曲线在 CAGD 中具有重要的作用，传统的 De Casteljau 算法虽然稳定，但计算复杂度为 $O(n^2)$ (见表 1，n 为曲线次数)，且其每一次递归都会产生不同的控制顶点，第一次递归产生 3 个新的控制顶点，第二次递归产生 2 个新的控制顶点，最后一次产生三次 Bézier 曲线在特定参数值 t 下的点，每一次递归都利用上一次递归产生的控制顶点，涉及控制顶点相对较多，初始的控制顶点在第一次递归后就不再使用，对于初始控制顶点的利用不够充分且效率偏低。利用快速求值算法求值时，每一次递归只产生一个控制顶点，共产生 3 个新控制顶点，每一次递归都使用了初始控制顶点，一定程度上减少了递归过程中产生的新控制顶点，计算复杂度为 $O(n)$ (见表 1，n 为曲线次数)，效率更高且易于实现。De Casteljau 算法与快速求值算法均利用递归计算，且只涉及线性插值，具有数值稳定性，但快速求值算法涉及的中间结果更少，效率更高。快速求值算法可以方便地绘制三次 Bézier 曲线，对复杂图形的绘制和编辑非常有用，同时其不仅适用于二维图形，还可以扩展到三维图形中(如图 3、图 6、图 9 所示)。但本文仅针对三次 Bézier 曲线给出了一种效率更高的算法，后续可以讨论任意次 Bézier 曲线的求值算法，也可以讨论三次 Bézier 曲面或可表示成 Bézier 形式的曲线曲面的快速求值算法，并将其应用到相关领域中。

参考文献

[1] 刘鼎元. 三次参数曲线段和三次 Bézier 曲线形状控制[J]. 应用数学学报, 1981(2): 158-165.

[2] 黄有度. 三次 Bézier 曲线的快速生成算法[J]. 工科数学, 1998(4): 56-59.

[3] 郑文明, 吴清江. 三次 Bezier 曲线绘制的一种新的快速算法[J]. 华侨大学学报(自然科学版), 2001(4): 362-365.

[4] 马月德, 曹淑娟, 李玉清. Bézier 曲线的几何生成法及其有效性分析[J]. 西安工业大学学报, 2006, 26(2): 166-169.

[5] Gordon, W.J. and Riesenfeld, R.F. (1974) Bernstein-Bézier Methods for the Computer-Aided Design of Free-Form Curves and Surfaces. *Journal of the ACM*, **21**, 293-310. https://doi.org/10.1145/321812.321824

[6] Boehm, W. and Müller, A. (1999) On de Casteljau's Algorithm. *Computer Aided Geometric Design*, **16**, 587-605. https://doi.org/10.1016/s0167-8396(99)00023-0

[7] Phien, H.N. and Dejdumrong, N. (2000) Efficient Algorithms for Bézier Curves. *Computer Aided Geometric Design*, **17**, 247-250. https://doi.org/10.1016/s0167-8396(99)00048-5

[8] Peters, J. (1994) Evaluation and Approximate Evaluation of the Multivariate Bernstein-Bézier Form on a Regularly Partitioned Simplex. *ACM Transactions on Mathematical Software*, **20**, 460-480. https://doi.org/10.1145/198429.198434

[9] Bezerra, L.H. (2013) Efficient Computation of Bézier Curves from Their Bernstein-Fourier Representation. *Applied Mathematics and Computation*, **220**, 235-238. https://doi.org/10.1016/j.amc.2013.05.079

[10] Bezerra, L.H. and Sacht, L.K. (2011) On Computing Bézier Curves by Pascal Matrix Methods. *Applied Mathematics and Computation*, **217**, 10118-10128. https://doi.org/10.1016/j.amc.2011.05.007

[11] Bezerra, L.H. (2012) Vandermonde Factorizations of a Regular Hankel Matrix and Their Application to the Computation of Bézier Curves. *SIAM Journal on Matrix Analysis and Applications*, **33**, 411-432. https://doi.org/10.1137/100800300

[12] Wang, X. and Jituan, Z. (2006) A Fast Eigenvalue Algorithm for Pascal Matrices. *Applied Mathematics and Computation*, **183**, 711-716. https://doi.org/10.1016/j.amc.2006.05.093

[13] Woźny, P. and Chudy, F. (2020) Linear-Time Geometric Algorithm for Evaluating Bézier Curves. *Computer-Aided Design*, **118**, Article 102760. https://doi.org/10.1016/j.cad.2019.102760

[14] Vijay, Saravana Kumar, G. and Chand, A.K.B. (2024) A Comprehensive Discussion on Various Methods of Generating Fractal-Like Bézier Curves. *Computational and Applied Mathematics*, **43**, Article No. 368. https://doi.org/10.1007/s40314-024-02887-0