Similar with the previous Fourier-based curve fitting, I look some papers, and write down the novel feedback representation of the object's shape based on **Bezier curve** and **nonuniform rational B-spline**.

1. Bezier-based

Bezier curve is a mathematical curve applied to 2D graphics applications (later I can extend it to 3D case). It creates and edits graphics by controlling some points on the curve (starting point, ending point, and separate intermediate points). It has the following series:

1.1 Linear formula

$$B(t) = P_0 + (P_1 - P_0)t \quad 0 \le t \le 1 \tag{1.1}$$

Given points P0, P1, the linear Bezier curve is just a straight line between two points

1.2 Quadratic formula

$$B(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2 \quad 0 \le t \le 1 \tag{1.2}$$

The path of the quadratic Bezier curve is given by the points P0, P1, and P2.

1.3 Cubed formula

$$B(t) = P_0(1-t)^3 + 3P_1 t(1-t)^2 + 3P_2 t^2(1-t) + P_3 t^3 \quad 0 \le t \le 1 \tag{1.3}$$

1.4 General formula

$$\begin{aligned} B(t) &= \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i \\ &= \binom{n}{0} P_0 (1-t)^n t^0 + \binom{n}{1} P_1 (1-t)^{n-1} t^1 + \cdots + \binom{n}{n-1} P_{n-1} (1-t)^1 t^{n-1} + \binom{n}{n} P_n (1-t)^0 t^n \end{aligned} \tag{1.4}$$

Where, $0 \le t \le 1$, P_i is coefficient of Bezier curve.

Where:

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \frac{n!(n-k+1)!}{k!} \tag{1.5}$$

(1.5) is binomial coefficient.

1.5 Used in the paper

Define:

$$c(\rho_k) = (x_k \quad y_k)^T \in R^2 \tag{1.6}$$

Where, $0 \le \rho_k \le L$, L is the cable length, where, $1 \le k \le 100$,as in the simulator, the 2D cable shape consists 100 sample points, while in 3D, it's 50 sample points, future, I can extend it to 3D case. A Bezier curve of *n* degree has *n* + 1 control points. I use n order Bezier curve as follows:

$$\begin{aligned}
\begin{pmatrix} x_k \\ y_k \end{pmatrix} &= \begin{pmatrix} B_1(\rho_k) \\ B_2(\rho_k) \end{pmatrix} \\
&= \begin{pmatrix} \sum_{i=0}^{n} \binom{n}{i} m_i (1-\rho_k)^{n-i} \rho_k^i \\ \sum_{i=0}^{n} \binom{n}{i} p_i (1-\rho_k)^{n-i} \rho_k^i \end{pmatrix} \\
&= \begin{pmatrix} \binom{n}{0} m_0 (1-\rho_k)^n \rho_k^0 + \binom{n}{1} m_1 (1-\rho_k)^{n-1} \rho_k^1 + \cdots + \binom{n}{n} m_n (1-\rho_k)^0 \rho_k^n \\ \binom{n}{0} p_0 (1-\rho_k)^n \rho_k^0 + \binom{n}{1} p_1 (1-\rho_k)^{n-1} \rho_k^1 + \cdots + \binom{n}{n} p_n (1-\rho_k)^0 \rho_k^n \end{pmatrix}
\end{aligned} \tag{1.7}$$

Define:

$$T(\rho_k) = \left[\binom{n}{0} (1-\rho_k)^n \rho_k^0 \quad \binom{n}{1} (1-\rho_k)^{n-1} \rho_k^1 \quad \cdots \quad \binom{n}{n} (1-\rho_k)^0 \rho_k^n \right] \in R^{1\times(n+1)} \tag{1.8}$$

Thus, combined with(1.7) and(1.8), we can have the following equation:

$$\begin{aligned}
c(\rho_k) &= \begin{pmatrix} x_k \\ y_k \end{pmatrix} \\
&= \begin{bmatrix} T(\rho_k) & 0 \\ 0 & T(\rho_k) \end{bmatrix} \begin{bmatrix} m_0 & m_1 & \cdots & m_n & p_0 & p_1 & \cdots & p_n \end{bmatrix}^T \\
&= G(\rho_k) s
\end{aligned} \tag{1.9}$$

Thus, we have:

$$\begin{aligned}
G(\rho_k) &= \begin{bmatrix} T(\rho_k) & 0 \\ 0 & T(\rho_k) \end{bmatrix} \in R^{2\times 2(n+1)} \\
s &= \begin{bmatrix} m_0 & m_1 & \cdots & m_n & p_0 & p_1 & \cdots & p_n \end{bmatrix}^T \in R^{2(n+1)}
\end{aligned} \tag{1.10}$$

Thus, we have:

$$\begin{aligned}
\bar{c} &= \left(c(\rho_1)^T \quad \cdots \quad c(\rho_N)^T \right)^T \in R^{2N} \\
\bar{G} &= \left(G(\rho_1)^T \quad \cdots \quad G(\rho_N)^T \right)^T \in R^{2N\times 2(n+1)}
\end{aligned} \tag{1.11}$$

Where, N is the total number of cable points (for 2D N=100, while 3D N=50).

Thus, we have:

$$\bar{c} = \bar{G} \cdot s \quad \Rightarrow \quad s = \left(\bar{G}^T \bar{G} \right)^{-1} \bar{G}^T \bar{c} \tag{1.12}$$

The rest content is similar with the Fourier-based method.

2. NURBS Curve-Based Image Feature

After introducing the Bezier curve-based image features, it should be noted that there are some complex curves that cannot be represented by the Bezier curve. For one thing, using high-order Bezier curves will cause a huge computational burden. For another, Bezier curves lack the ability of local shape modification. In addition, Bezier curves are not robust to regular and symmetric curves. Thus, I will introduce the NURBS curves. The NURBS curve has a local shape modifying property, and the number of its control points is independent of the order of curves. Hence, it can describe complex curves more accurately and efficiently. Moreover, the NURBS curve is invariant under translation, rotation, scaling, and perspective projection. Thus, the curve parameters can be used to design image features.

The definition of the NURBS curve is:

$$P(\rho_i) = \frac{\sum_{i=0}^{n} \omega_i p_i B_{ik}(\rho_i)}{\sum_{i=0}^{n} \omega_i B_{ik}(\rho_i)} \quad 0 \le \rho_i \le L \quad 1 \le i \le N \tag{1.13}$$

Where, N is the total number of cable points (for 2D N=100, while 3D N=50). It should be mentioned that there are $n+1$ k-order B-spline basis functions for a NURBS curve of k degree with $n+1$ control points.

And ω_i is the weight of the control coefficients p_i.

$B_{ik}(\rho_i)$ is the k order B-spline basis function as follows:

$$B_{ik}(s) = \frac{1}{k!}\sum_{j=0}^{k-i}\left((-1)^j \binom{k+1}{j}(s+k-i-j)^k\right) \tag{1.14}$$

Give the examples of the three order B-spline basis function:

$$\begin{aligned}
B_{03}(s) &= \frac{1}{3!}\sum_{j=0}^{3-0}\left((-1)^j \binom{4}{j}(s+3-0-j)^k\right) \\
B_{13}(s) &= \frac{1}{3!}\sum_{j=0}^{3-1}\left((-1)^j \binom{4}{j}(s+3-1-j)^k\right) \\
B_{23}(s) &= \cdots \\
B_{33}(s) &= \cdots
\end{aligned} \tag{1.15}$$

Within the paper, we use k order B-spline basis function and we define the following equation:

$$c(\rho_i) = \begin{pmatrix} x_i & y_i \end{pmatrix}^T \in R^2 \tag{1.16}$$

Where, $0 \le \rho_i \le L$

Thus, combined with (1.13) and (1.16), we have:

$$\begin{aligned} x_i &= \frac{\omega_0 m_0 B_{0k} + \omega_1 m_1 B_{1k} + \cdots + \omega_n m_n B_{nk}}{\omega_0 B_{0k} + \omega_1 B_{1k} + \cdots + \omega_n B_{nk}} \\ &= \begin{pmatrix} \omega_0 B_{0k} \\ \omega_1 B_{1k} \\ \vdots \\ \omega_n B_{nk} \end{pmatrix}^T diag\left(\omega_0 B_{0k} + \omega_1 B_{1k} + \cdots + \omega_n B_{nk}\right)^{-1} \begin{pmatrix} m_0 \\ m_1 \\ \vdots \\ m_n \end{pmatrix} \\ y_i &= \frac{\omega_0 p_0 B_{0k} + \omega_1 p_1 B_{1k} + \cdots + \omega_n p_n B_{nk}}{\omega_0 B_{0k} + \omega_1 B_{1k} + \cdots + \omega_n B_{nk}} \\ &= \begin{pmatrix} \omega_0 B_{0k} \\ \omega_1 B_{1k} \\ \vdots \\ \omega_n B_{nk} \end{pmatrix}^T diag\left(\omega_0 B_{0k} + \omega_1 B_{1k} + \cdots + \omega_n B_{nk}\right)^{-1} \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{pmatrix} \end{aligned} \tag{1.17}$$

For saving the display space, $diag\left(\omega_0 B_{0k} + \omega_1 B_{1k} + \cdots + \omega_n B_{nk}\right) \in R^{(n+1)\times(n+1)}$,it's a diagonal matrix, and its diagonal element is $\omega_0 B_{0k} + \omega_1 B_{1k} + \cdots + \omega_n B_{nk}$.

Define:

$$P = \begin{pmatrix} \omega_0 B_{0k} \\ \omega_1 B_{1k} \\ \vdots \\ \omega_n B_{nk} \end{pmatrix}^T diag\left(\omega_0 B_{0k} + \omega_1 B_{1k} + \cdots + \omega_n B_{nk}\right)^{-1} \in R^{1\times(n+1)} \tag{1.18}$$

Thus, we have:

$$c(\rho_i) = \begin{pmatrix} x_i \\ y_i \end{pmatrix} = diag(P, P)\begin{pmatrix} m_0 & m_1 & \cdots & m_n & p_0 & p_1 & \cdots & p_n \end{pmatrix}^T = G(\rho_i)s \tag{1.19}$$

Where:

$$\begin{aligned} G(\rho_i) &= diag(P, P) \in R^{2\times 2(n+1)} \\ s &= \begin{pmatrix} m_0 & m_1 & \cdots & m_n & p_0 & p_1 & \cdots & p_n \end{pmatrix}^T \in R^{2(n+1)} \end{aligned} \tag{1.20}$$

Thus, we have:

$$\begin{aligned} \bar{c} &= \left(c(\rho_1)^T \quad \cdots \quad c(\rho_N)^T\right)^T \in R^{2N} \\ \bar{G} &= \left(G(\rho_1)^T \quad \cdots \quad G(\rho_N)^T\right)^T \in R^{2N\times 2(n+1)} \end{aligned} \tag{1.21}$$

Thus, we have:

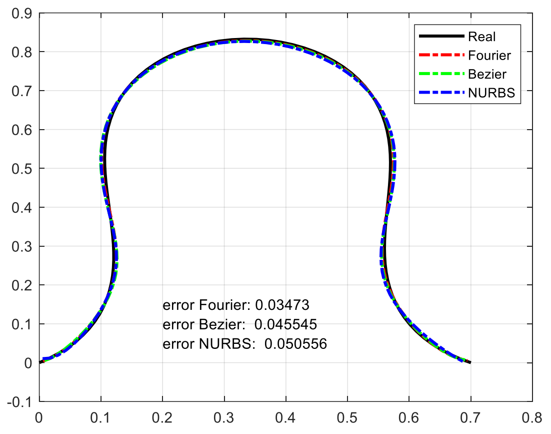
$$\bar{c} = \bar{G} \cdot s \quad \Rightarrow \quad s = \left(\bar{G}^T \bar{G}\right)^{-1} \bar{G}^T \bar{c} \tag{1.22}$$

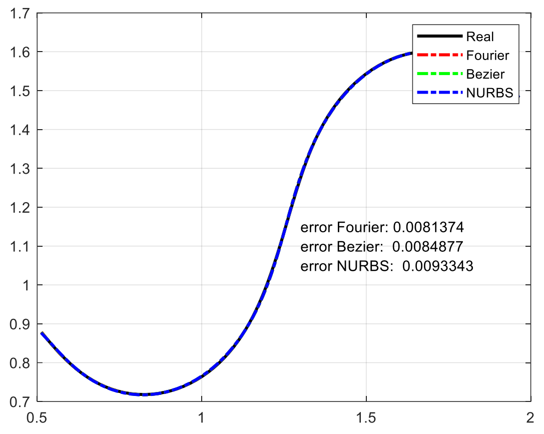
Where, N is the total number of cable points (for 2D N=100, while 3D N=50).

3. Validation

In this section, I use 2D simulator simulation. The cable length equals 2.

To compare the performance of the proposed techniques, I add Fourier-based method.





The error is between real shape and estimation shape(Fourier, Bezier and NURBS).

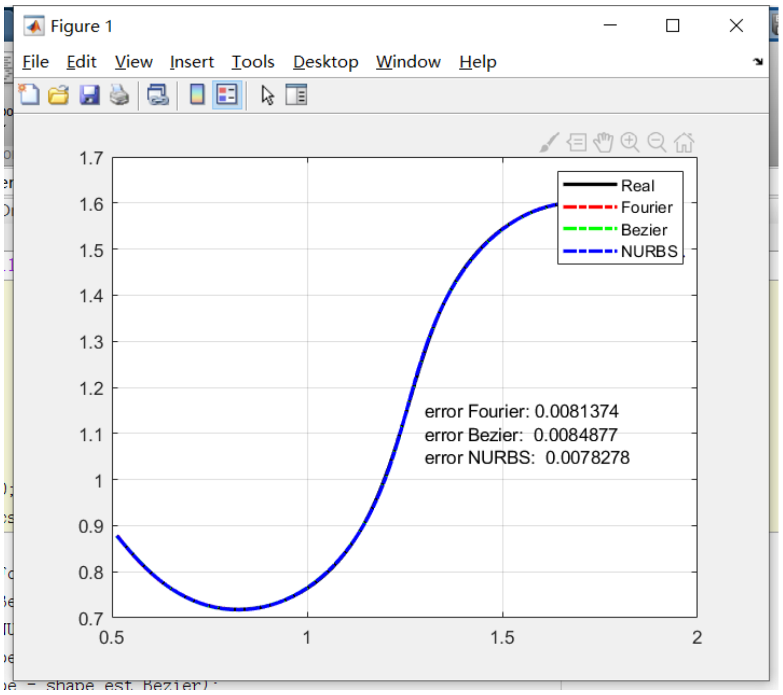
The order of Fourier method is 4, thus $G \in R^{200\times18}$ $s \in R^{18}$.

The order of Bezier method is 8, thus $G \in R^{200\times18}$ $s \in R^{18}$

The order of NURBS method is 8, thus $G \in R^{200\times18}$ $s \in R^{18}$

The reason why sets the above conditions is to compare the effectiveness of the presented algorithms fairly when the parameters dimensions are same.

Supplement: Because for NURBS method, I just select weight coefficients ω_i randomly, thus, NURBS fitting result is good sometimes good or sometimes bad. However, in most cases, NURBS fitting result is the best among three methods, as you see in this figure.



error Fourier: 0.0081374
error Bezier: 0.0084877
error NURBS: 0.0078278