5.2 Applications to the existing surface interpolation algorithms

In this paper, we use the chord-length parametrization to compute the parameter values $\mathbf{T}$. Because it is quite satisfactory and adequate in most practical applications [8, 9], and it is assumed that the closed contours have been aligned beforehand.

5.2.1 Adapts Piegl and Tiller's algorithm based on Conjecture 1

The Conjecture-1 gives the knot some flexibility similar to the Schoenberg-Whitney condition[4]. Based on the idea of Piegl and Tiller's algorithm [12], B-spline curve interpolation algorithm based on existing knot vector, shown in Fig. 1, we develop a closed B-spline curve interpolation method that attempts to choose the knots from a given input knot vector without causing the numerical stability problem. Summarized below are its overall steps.

To apply Piegl and Tiller's algorithm to closed B-spline curve interpolation based on the existent knot vector, we need to define two vectors, called anchor domain knots $\mathbf{D}^p$ and bound-value vector $\mathbf{X}$, which store the corresponding values as follows:

$$u_i^p = \begin{cases} t_i & p \text{ is odd} \\ \dfrac{d_n}{2} + \dfrac{t_{i-1}+t_i}{2} & p \text{ is even } (i=1,\cdots,n) \end{cases} \qquad u_0^p = 0 \quad u_{n+1}^p = 1 \tag{16}$$

$$x_i = \begin{cases} \dfrac{t_i + t_{i+1}}{2} & p \text{ is odd} \\ \dfrac{d_n}{2} + t_i & p \text{ is even } (i=0,\cdots,n) \end{cases} \tag{17}$$

Because of $x_{i-1} < u_i^p < x_i$, analogous to Piegl and Tiller's [12] approach, we also define an interval (a, b), shown in Fig. 4, to give each knot some flexibility.

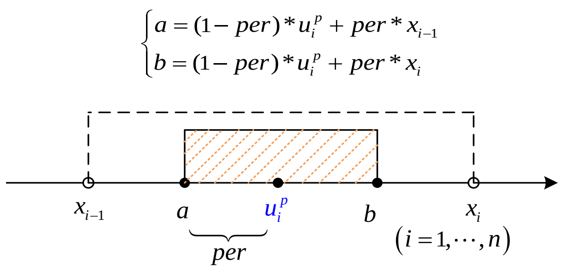
$\begin{cases} a = (1-per)*u_i^p + per*x_{i-1} \\ b = (1-per)*u_i^p + per*x_i \end{cases}$
x_{i-1}
a
u_i^p
b
x_i
$(i=1,\cdots,n)$
per

Figure 4: The diagram of knot interval used in closed B-spline curve algorithm

The factor *per* denotes the percentage of interval $[x_{i-1}, x_i]$ use. The corresponding pseudocode as shown in Procedure 2.

Listing 2: Algorithm for closed B-spline curve interpolation based on a given knot vector

```
Procedure 2 $\textbf{interpolate\_points\_by\_input\_knots}(\mathbf{Q}, \mathbf{T}, \bar{\mathbf{U}}, p, per)$
INPUT:
  $\mathbf{Q} = \{\mathbf{Q}_i\} \quad (i=0,\cdots,n) \leftarrow$ data points
  $\mathbf{T} = \{t_0,\cdots,t_n\} \leftarrow$ parameter values
  $\bar{\mathbf{U}} \leftarrow$ input knot vector
  $p \leftarrow$ required degree
  $per \leftarrow$ percentage of interval use
OUTPUT:
  $\tilde{\mathbf{R}}_i (i=0,\cdots,n+p) \leftarrow$ control points of closed B-spline curve
  $\tilde{\mathbf{U}} = \{\underbrace{0,\cdots,0}_{p}, \tilde{u}_0,\cdots,\tilde{u}_{n+1}, \underbrace{1,\cdots,1}_{p}\} \leftarrow$ clamped knot vector for closed B-spline curve
  $\bar{\mathbf{U}} \leftarrow$ updated input knot vector
ALGORITHM:
  $\mathbf{D}^p = \{u_0^p,\cdots,u_{n+1}^p\} \leftarrow$ anchor domain knots
  $\mathbf{X} = \{x_0,\cdots,x_n\} \leftarrow$ bound-value vector
  $\mathbf{D} = \{u_0,\cdots,u_{n+1}\} \leftarrow$ memory for domain knots
  for (i = 1; i <= n; i++) {
    $a = (1-per)*u_i^p + per*x_{i-1}$
    $b = (1-per)*u_i^p + per*x_i$
    $\bar{u}_k \leftarrow$ knot in $\bar{\mathbf{U}}$ that closest to $u_i^p$
    if ($\bar{u}_k \in (a,b)$) $u_i = \bar{u}_k$ else $u_i = u_i^p$
    }
    $\mathbf{U} \leftarrow$ cyclic knot vector based on $\mathbf{D}$ and Eq.(4)
    $\mathbf{R}_i \leftarrow$ control points by interpolating $\mathbf{Q}_i$ with $\mathbf{T},\mathbf{U}$ or $\tilde{\mathbf{T}},\mathbf{U}$
    $\tilde{\mathbf{R}}_i, \tilde{\mathbf{U}} \leftarrow$ clamp the closed curve via Procedure-1
    $\bar{\mathbf{U}} \leftarrow \mathbf{merge}(\bar{\mathbf{U}}, \tilde{\mathbf{U}})$ // update input knot vector
    return $\tilde{\mathbf{R}}_i, \tilde{\mathbf{U}}, \bar{\mathbf{U}}$
```

Procedure 2 accepts the data points and a knot vector as input. It then judges whether the knot in a given knot vector can be used to interpolate the data points. Firstly, it sets up the anchor domain knots $\mathbf{D}^p$ for degree p and another one called bound-value vector $\mathbf{X}$. If there are input knots in a flexibility interval, the algorithm chooses the one that nearest to the anchor domain knot u_i^p. Otherwise, the corresponding anchor knot u_i^p is used. After the curve interpolation is done, the input knot vector $\bar{\mathbf{U}}$ is updated by adding the anchor knots whose flexibility interval did not consist of input knots.

Listing 3: Algorithm for interpolating serial rows of data points

```
Procedure 2 interpolate_all_row_points($\mathbf{Q}_{i,j}, q, per$)
INPUT:
  $\mathbf{Q}_{i,j}(i = 0,\cdots,m; j = 0,\cdots,n_i) \leftarrow$ rows of data points
  $q \leftarrow$ row degree
  $per \leftarrow$ percentage of interval use
OUTPUT:
  $\mathbf{R}_{i,j}(i = 0,\cdots,m; j = 0,\cdots,n+q) \leftarrow$ control points of closed B-spline curves
  $\bar{\mathbf{V}} = \{\underbrace{0,\cdots,0}_{q}, \bar{u}_0,\cdots,\bar{v}_{n+1}, \underbrace{1,\cdots,1}_{q}\} \leftarrow$ common knot vector of all B-spline curves
ALGORITHM:
  $n_{\max} = \mathbf{max}\{n_0,\cdots,n_m\}$
  $\mathbf{T}_{\text{avg}} \leftarrow$ average of parameter values whose row index eqauls to $n_{\max}$
  $\mathbf{D} = \{\bar{v}_0,\cdots,\bar{v}_{n_{\max}+1}\} \leftarrow$ domain knots computed via $\mathbf{T}_{\text{avg}}$ and Eq.(16)
  $\bar{\mathbf{V}} = \{\underbrace{0,\cdots,0}_{q}, \bar{v}_0,\cdots,\bar{v}_{n_{\max}+1}, \underbrace{1,\cdots,1}_{q}\} \leftarrow$ initial input knots based on $\mathbf{D}$

    for (i = 0; i <= m; i++) {
      $\mathbf{T}^{(i)} \leftarrow$ parameter value of $\mathbf{Q}_{i,j}(j = 0,\cdots,n_i)$
      $\left\{\begin{array}{l} \tilde{\mathbf{R}}_{i,j}(j = 0,\cdots,n_i+q) \\ \mathbf{V}_i = \mathbf{V}^{n_i+q,q}_{\text{clamped}}(v_i) \\ \bar{\mathbf{V}} \leftarrow \text{updated input knot vector} \end{array}\right.$   $\leftarrow$ interpolate_points_by_input_knots($\mathbf{Q}_i, \mathbf{T}^{(i)}, \bar{\mathbf{V}}, p, per$)
    }
    $\mathbf{R}_{i,j} \leftarrow$ make B-spline curves compatible via knot refinement
    return $\mathbf{R}_{i,j}, \bar{\mathbf{V}}$
```

The idea that passing a knot vector into the closed B-spline curve interpolation procedure is very important when interpolating a sequence of rows of data points. That is, each row of data points is interpolated independently with a given knot vector passed in for each row and then updated subsequently. It leads to that each B-spline curve owns many knots in common and the explosion of the number of the common knot vector can be avoided when making these curves compatible. The overall procedure is summarized in Procedure-3. The value of the variable *per* is restricted in the interval [0, 1], represents the flexibility of a knot. A larger *per* could decrease the number of control net. In addition, when the factor *per* equals to 0, this algorithm degenerates to the traditional lofting method [2, 3].

5.2.2 Adapts Park and Wang's algorithm based on Conjecture 2

The key step of Park's approach is constructing a common knot vector. For the common cyclic knot vector, we just need to determine a common domain knots. Although this method generates an under-determined system, see Eq.(12), the corresponding control points $\tilde{\mathbf{P}}_{(\hat{n}+p+1)\times 1}$ of closed B-spline curve interpolation can be achieved via minimizing the energy of the curve. That leads to solving the below linear system:

$$\begin{pmatrix} \tilde{\mathbf{K}}_{(\hat{n}+p+1)\times(\hat{n}+p+1)} & \tilde{\mathbf{N}}^{\mathbf{T}}{}_{(\hat{n}+p+1)\times(n+p+1)} \\ \tilde{\mathbf{N}}_{(n+p+1)\times(\hat{n}+p+1)} & \varnothing_{(n+p+1)\times(n+p+1)} \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{P}} \\ \tilde{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \varnothing \\ \tilde{\mathbf{Q}} \end{pmatrix} \tag{18}$$

where $\tilde{\mathbf{K}}$ is the stiffness matrix defined in Eq.(10) with a cyclic knot vector. $\tilde{\mathbf{v}}_{(n+p+1)\times 1}$ is a vector storing Lagrange multipliers and $\tilde{\mathbf{N}}$ is determined via parameter values $\mathbf{T}$ and $\tilde{\mathbf{T}}$ for odd degree and even degree, respectively.

Listing 4: Algorithm for building the domain knots from given domain knots

```
Procedure 4 build_domain_knots_by_input_knots($\mathbf{T}, \mathbf{D}, p, per$)
INPUT:
  $\mathbf{T} = \{t_0, \cdots, t_n\} \leftarrow$ parameter values
  $\mathbf{D} = \{u_0, \cdots, u_{\hat{n}+1}\} \leftarrow$ input domain knots
  $p \leftarrow$ required degree
  $per \leftarrow$ percentage of interval use
OUTPUT:
  $\bar{\mathbf{D}} = \{\bar{u}_0, \cdots, \bar{u}_{n+1}\} \leftarrow$ updated input knot vector
ALGORITHM:
  $\mathbf{D}^p = \{u_0^p, \cdots, u_{n+1}^p\} \leftarrow$ anchor domain knots
  $\mathbf{X} = \{x_0, \cdots, x_n\} \leftarrow$ bound-value vector
  $\bar{\mathbf{D}} = \{\bar{u}_0, \cdots, \bar{u}_{n+1}\} \leftarrow$ memory for domain knots
  span = 0
  for (i = 1; i <= n; i++) {
    $a = (1 - per) * u_i^p + per * x_{i-1}$
    $b = (1 - per) * u_i^p + per * x_i$
    while ($u_{i+\text{span}} < a$) {
      span++
      if ($i + \text{span} > \hat{n} + 1$) break the for-loop // no sufficient knot to choose
    }
    if ($u_{i+\text{span}} < b$) $\bar{u}_i = u_{i+\text{span}}$ else { $\bar{u}_i = u_i^p$   span-- }
  }
  for (k = i; k <= n; k++) { $\bar{u}_k = u_k^p$ } // handle the remaining knots
  return $\bar{\mathbf{D}}$
```

Similar to Wang's [14] (see Theorem-1 and Procedure-2 of that paper) method that constructing a knot vector that guarantees the corresponding system matrix is *full-rank*, we use Conjecture-2 to build a domain knots that make the system matrix of closed B-spline curve interpolation full-rank. The related steps are summarized in Procedure 4. Furthermore, unlike the method of Piegl and Tiller [12], we notice that the clamping and knot refinement is not needed in Park and Wang's approach. The common domain knots can be determined by the following two steps: first, an initial common domain vector $\mathbf{D}$ is computed, then for each row, a domain vector $\mathbf{D}^{(i)}$ is obtained from Procedure 4 and $\mathbf{D}$ is updated by merging itself with $\mathbf{D}^{(i)}$. The algorithm is described as follows.

Listing 5: Algorithm for building common domain knots of rows of data points

```
Procedure 5 build_common_domain_knots($\mathbf{Q}_{i,j}, q, per$)
INPUT:
  $\mathbf{Q}_{i,j}(i = 0, \cdots, m; j = 0, \cdots, n_i) \leftarrow$ rows of data points
  $q \leftarrow$ row degree
  $per \leftarrow$ percentage of interval use
OUTPUT:
  $\mathbf{D} = \{u_0, \cdots, u_{\hat{n}+1}\} \leftarrow$ constructed common domain knots
ALGORITHM:
  $n_{\max} = \mathbf{max}\{n_0, \cdots, n_m\}$
  $\mathbf{T}_{\text{avg}} \leftarrow$ average of parameter values whose row index eqauls to $n_{\max}$
  $\mathbf{D} = \{\bar{v}_0, \cdots, \bar{v}_{n_{\max}+1}\} \leftarrow$ initial domain knots computed via $\mathbf{T}_{\text{avg}}$ and Eq.(16)
  for (i = 0; i <= m; i++) {
    $\mathbf{T}^{(i)} \leftarrow$ compute the parameter value of $\mathbf{Q}_{i,j}(j = 0, \cdots, n_i)$
    $\mathbf{D}^{(i)} \leftarrow$ build domain knots by input knots($\mathbf{T}^{(i)}, \mathbf{D}, p, per$)
    $\mathbf{D} \leftarrow$ merge($\mathbf{D}, \mathbf{D}^{(i)}$) // update the domain knots
  }
  return D
```

Once the common domain knots is built, we can use the linear system Eq.(18) and the common domain knots to compute all the control points of the B-spline curves that pass through all the rows of data points $\mathbf{Q}_{i,j}$. Lastly, the lofted B-spline surface can be achieved via interpolating a set of columns of control points.