High-performance mesh morphing with adaptive B-splines

Dmitry Pinaev and Sean Mauch

Abstract A novel mesh morphing method based on adaptive hierarchical B-splines is proposed. We discuss the mathematical background of the method and demonstrate the parallelization approach in an MPI environment. The performance and scalability data of the method is provided. The mesh deformation quality is compared with the industry-acknowledged RBF-based morpher.

1 Introduction

Mesh morphing plays an important role in computational fluid dynamics. A number of morphing methods have been developed to address this problem [9]:

- Finite-element-based elasticity morphing [8, 10]
- Spring-analogy morphing [11, 3, 12]
- Interpolation-based methods [4]

The finite-element-based methods can be numerically expensive, especially on polyhedral meshes. The spring-analogy method requires controlling many degrees of freedom by introducing spring elements inside the discretization cells [3]. Therefore, the family of various interpolation methods looks like a good trade-off between numerical complexity and final mesh quality.

In this paper we demonstrate an alternative to radial basis functions (RBFs) [5, 1, 2]. an interpolation-based morphing method based on hierarchical B-splines. We discuss the mathematical foundation of the method in the next section. Detailed information on the algorithm is presented in [6].

Dmitry Pinaev
Siemens PLM, Nordostpark 3, 90411 Nuremberg, e-mail: dmitry.pinaev@siemens.com

Sean Mauch
Siemens PLM, e-mail: sean.mauch@siemens.com

2 B-Spline Approximation

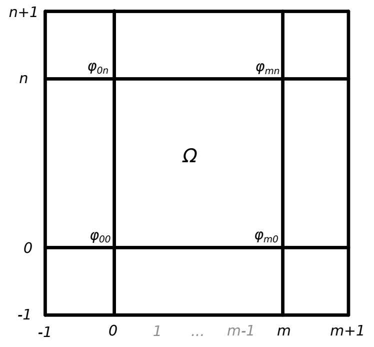


Fig. 1: Domain Ω with overlaid lattice Φ.

Consider a rectangular domain $\Omega = \{(x,y) \mid 0 \le x < m, 0 \le x < n\}$ (Fig. 1). There is a set of points $P = \{(x_c, y_c, z_c)\}$ in 3D space, where $\{x_c, y_c\}$ is a point in Ω. To approximate the data P, we introduce a lattice Φ, overlaid on Ω with dimensions $(0, m+1) \times (0, n+1)$. Let ϕ_{ij} to be a B-spline coefficient located at the ij-th position in the lattice Φ. The approximation function in Ω is defined then as follows:

$$f(x,y) = \sum_{k=0}^{3} \sum_{l=0}^{3} B_k(s) B_l(t) \phi_{(i+k)(j+l)} \tag{1}$$

where (i, j) are the indices at the lower corner of the voxel that contains the point (x, y), and (s, t) is the parametrization of the point within voxel, i.e. $i = \lfloor x \rfloor - 1$ and $s = x - i$. Equation (1) can be easily extended to 3D.

We use the following B-spline basis functions:

$$\begin{aligned} B_0(s) &= (1-s)^3/6, \\ B_1(s) &= (3s^3 - 6s^2 + 4)/6, \\ B_2(s) &= (-3s^3 + 3s^2 + 3s + 1)/6, \\ B_3(s) &= s^3/6. \end{aligned} \tag{2}$$

The 2D case considered below requires a lattice of 16 coefficients to define a spline approximation on the square. Imagine an arbitrary point z_c where the interpolation is defined as

$$z_c = \sum_{k=0}^{3} \sum_{l=0}^{3} w_{kl} \phi_{kl}, \tag{3}$$

where $w_{kl} = B_k(s)B_l(t)$ and $s \in (0.0, 1.0), t \in (0.0, 1.0)$. We call the stencil of ϕ's that surround a given point z_c its relevant coefficients (Fig. 2 a). There are many possible ϕ_{kl} that satisfy the equation. To uniquely define the coefficients we impose $\sum_{k=0}^{3}\sum_{l=0}^{3}\phi_{kl}^2 \to \min$.

Computing the coefficients ϕ_{kl} in the least square sense yields the equation:

$$\phi_{kl} = \frac{w_{kl}z_c}{\sum_{a=0}^{3}\sum_{b=0}^{3}(B_a(s)B_b(t))^2} = \frac{w_{kl}z_c}{\sum_{a=0}^{3}\sum_{b=0}^{3}w_{ab}^2}. \tag{4}$$

This minimizes the deviation of the function f from zero across the domain.

Now, we consider a region of scattered data points z_c (empty points). The data points in square influence the B-spline coefficient ϕ_{ij} (solid point). We call such a square the proximity data set P of control point ϕ_{ij} (Fig. 2 b).

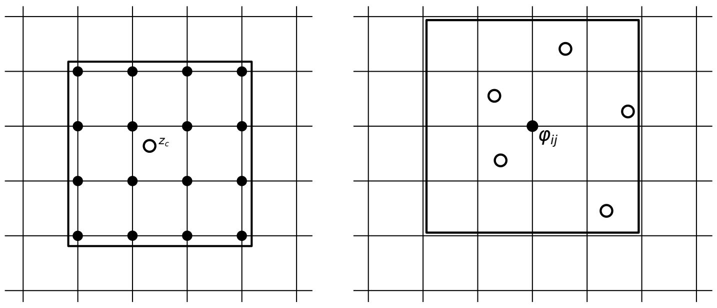


Fig. 2: (a) Relevant coefficients ϕ for some data point; (b) Proximity data set for a single B-spline coefficient.

When proximity data set contains just one data point the coefficient ϕ_{ij} is computed exactly according to Equation (4). In case of multiple points in the proximity data set, the control point gets several possible values:

$$\phi_c = \frac{w_c z_c}{\sum_{a=0}^{3}\sum_{b=0}^{3}w_{ab}}, \text{ where } c \in P. \tag{5}$$

In order to get a unique ϕ_{ij} value we minimize $\sum_c(w_c\phi_{ij} - w_c\phi_c)^2$ and obtain:

$$\phi_{ij} = \frac{\sum_c w_c^2\phi_c}{\sum_c w_c^2} = \frac{\delta_{ij}}{\omega_{ij}}. \tag{6}$$

Two important properties follow from Equation (6):

- The relevant data points for a given coefficient ϕ_{ij} are local and bound within a proximity data set.
- The numerator and denominator in Equation (6) are additive, hence, could be computed locally and reduced across MPI-paritions.

3 Adaptive interpolation

The adaptive B-spline interpolation consists in applying a sequence of refined control lattices. In a 2D case, the first lattice has dimensions 4×4, i.e. with parameters $m = 1$, $n = 1$. The next lattice is build with parameters $m = 2$, $n = 2$ which results in dimensions 5×5. The size of the lattice is defined as $(l * 2 + 3) \times (l * 2 + 3)$, where $l = 1, 2, 3 \ldots N$ is a refinement level.

Instead of finding the final correction right away, we apply a sequence of corrections at different scales. Further we consider current coordinates $\mathbf{x}$, target coordinates $\mathbf{t}$, and residual $\mathbf{r} = \mathbf{t} - \mathbf{x}$. Residual is nothing else but the displacement known at control points and unknown elsewhere. We can represent the target coordinates in terms of corrections I_i and residual r_i:

$$\begin{aligned} t &= x_0 + r_0, \\ t &= x_0 + I_1 + r_1, \\ t &= x_0 + \sum_{i=1}^{N} I_i + r_N, \end{aligned}$$

As the residual r_N goes to zero we obtain some data points in the domain Ω, where $||r_N|| < \varepsilon$. There are no further corrections required near to these data points. We store just non-zero B-spline coefficients ϕ to reduce the memory footprint.

In the evaulation phase the central part of the object will be fit with appropriate coefficients almost immediately (Fig. 3 a). With further refinement we would obtain non-zero B-spline coefficients just along the boundary (Fig. 3 b). The final refinement level yelds non-zero coefficients just at some rare locations (Fig. 3 c).

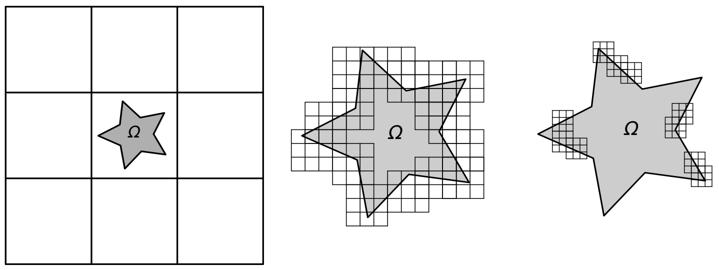


Fig. 3: (a) First coefficient lattice; (b) Control lattice at some higher refinement level with only non-zero knots; (c) Final refinement level

4 MPI Parallelization

For a simulation with morphing, the control points (a subset of the vertices) and the evaluation points (the rest of the vertices) are distributed according to domain decomposition.

Having each process calculate its contribution to the interpolation coefficients in terms of the δ and ω values (Equation (6)) is a good start. We use clustering to group the grid indices for the nonzero coefficient values into a relatively small number of buckets. Then we build bounding boxes in index space for each of the buckets. For the obtained lists of bucket bounding boxes, we perform a gather-to-all operation so that each process has an approximate description of the coefficient data held by all other processes. Each process performs intersection queries to determine which of the distributed buckets are relevant for interpolating displacements at the local evaluation points. Given the lists of relevant buckets, we use a point-to-point communication pattern to deliver the coefficient data.

Upon receiving several control point sets relevant for the same evaluation point, we add up the corresponding δ and ω values and find the final coefficients.

5 Morphing Quality Analysis

In the section we compare the discussed B-spline morpher to the RBF morpher with inverse multiquadric basis functions [7].

A hollow sphere is placed inside a meshed block (Fig. 4). The sphere is moved towards left upper corner of the block. The vertices that belong to the outer boundaries of the box are sliding within their boundary rectagles.

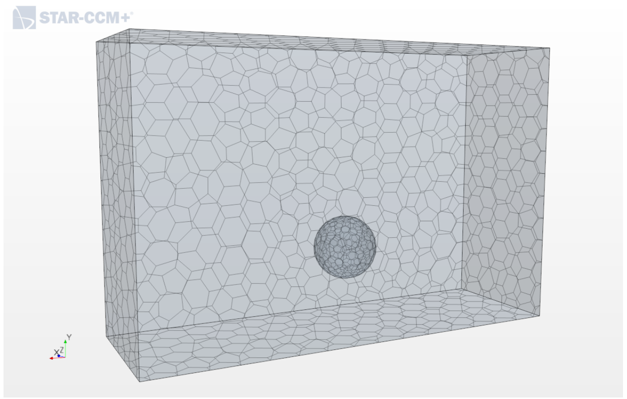


Fig. 4: A hollow sphere inside a box.

The displacement interpolated by the RBF morpher has the largest magnitude in the boundary layers of the sphere (Fig. 5) and quickly decays in the areas near the box boundaries. The prismatic boundary layers around the sphere are significantly stretched towards sphere's original position. The cells on the outer boundaries are skewed due to mesh motion.

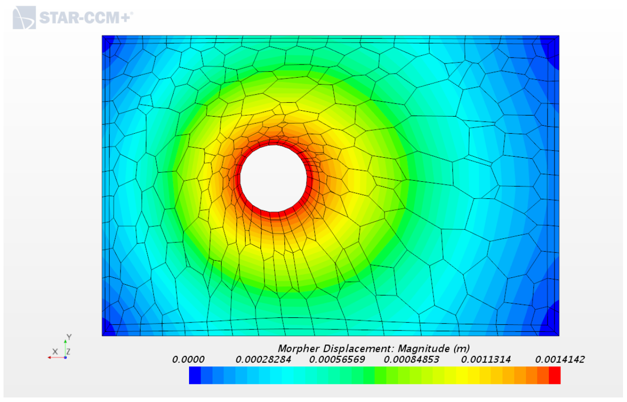


Fig. 5: Mesh deformation by RBF morpher.

In contrast to RBF, the B-spline morpher distributes the displacements more evenly and keeps the balance between very large and very small mesh cells (Fig. 6). The boundary layers next to the sphere are deformed less that in the RBF case. The prismatic cells at outer box boundaries remain tangential to the surface.

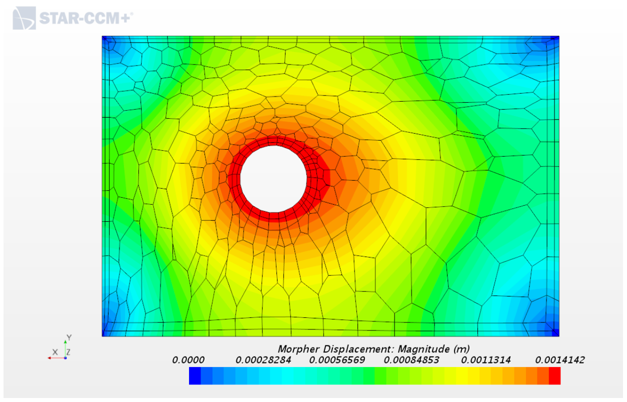


Fig. 6: Mesh deformation by B-spline morpher.

5.1 Scalability Analysis

To assess the parallel scalability, we use a mesh with 8 million cells. The deforming geometry is a cube (Fig. 7). The control points are defined on the top and bottom faces. The bottom boundary is fixed. The top boundary is rotated by 0.5 radians and is indented by pressing a hemisphere into it. Interpolation defines the displacement for all other points.

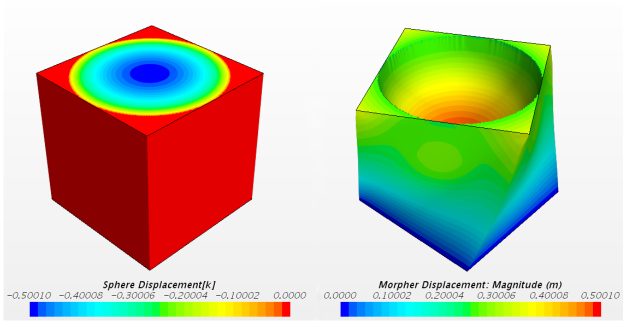
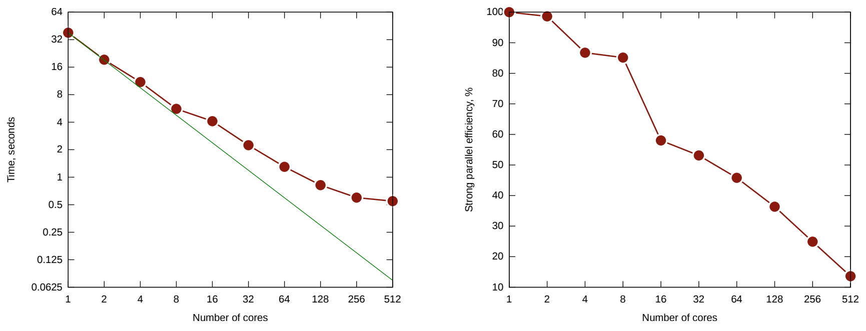


Fig. 7: (a) The prescribed cube deformation; (b) The result of morphing.



(a) label1 (b) label2

Fig. 8: Strong scalability (left), strong scalability efficiency (right), 8 million cells.

6 Summary

We demonstrated the application of a method of scattered data interpolation in the context of three-dimensional mesh morphing. The B-Spline interpolation appeared to be easily parallelizable in an MPI environment and exhibited mesh deformation patterns similar to RBF. B-Splines tend to preserve the properties of boundary-layers better that RBF.

References

1. S. Aubert, F. Mastrippolito, Q. Rendu, M. Buisson, and F. Ducros. Planar slip condition for mesh morphing using radial basis functions. *Procedia Engineering*, 203:349 – 361, 2017. 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain.
2. M. E. Biancolini. Mesh morphing accelerates design optimization, ansys advantage. *ANSYS Advantage*, 4, 2010.
3. C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163(1):231 – 245, 1998.
4. N. A. Gumerov and R. Duraiswami. Fast radial basis function interpolation via preconditioned krylov iteration. *SIAM Journal on Scientific Computing*, 29(5):1876–1899, 2007.
5. S. Jakobsson and O. Amoignon. Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids*, 36(6):1119 – 1136, 2007.
6. S. Lee, G. Wolberg, and S. Y. Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, Jul 1997.
7. M. Mongillo. Choosing basis functions and shape parameters for radial basis function methods. 2011.
8. S. M. Shontz and S. A. Vavasis. A robust solution procedure for hyperelastic solids with large boundary deformation. *Engineering with Computers*, 28(2):135–147, Apr 2012.
9. M. L. Staten, S. J. Owen, S. M. Shontz, A. G. Salinger, and T. S. Coffey. A comparison of mesh morphing methods for 3d shape optimization. In *IMR*, 2011.
10. K. Stein, T. E. Tezduyar, and R. Benney. Automatic mesh update with the solid-extension mesh moving technique. *Computer Methods in Applied Mechanics and Engineering*, 193(21):2019 – 2032, 2004. Flow Simulation and Modeling.
11. J. T. Batina. Unsteady euler airfoil solutions using unstructured dynamic meshes. 28, 09 1990.
12. Y. Yang. Application of spring analogy mesh deformation technique in airfoil design optimization. Master's thesis, 07 2015.