A Least-Squares Optimization Model for Quad Mesh Offset

Zitao Xu

Independent Scholar, Mukilteo, Washington, USA zitaoxu@gmail.com

Corresponding author: Zitao Xu, zitaoxu@gmail.com

Abstract. This article presents a least-squares optimization model for quadrilateral mesh offsets. Starting from an initial offset mesh generated from a base quad mesh, the method improves offset quality by introducing tangential corrections at offset vertices and solving for them globally. Each correction is represented by two scalar variables associated with two independent directions taken from the corresponding base quad, so that the optimization preserves the intended offset distance while allowing the offset mesh to adjust tangentially.

For each quad, the objective function is constructed from local shape-preservation terms together with a center-consistency term, so that the optimized offset quad remains as consistent as possible with the shape and size of the corresponding base quad while reducing local crimping and shearing. The resulting formulation is a global least-squares problem over the mesh and leads to a sparse linear system. Examples in both convex and concave regions show that the method can significantly improve the quality of quad-mesh offsets and can be applied iteratively for further improvement.

Keywords: quad mesh offset; least-squares optimization; geometric modeling; mesh quality improvement; sparse linear system
DOI: https://doi.org/10.5281/zenodo.21271404

1 INTRODUCTION

Offset construction is a fundamental operation in geometric modeling and computer-aided design, and offset surfaces or offset meshes arise in applications such as shell generation, toolpath generation, manufacturing, and geometric processing [1,2,4,11].

For mesh-based surfaces, direct offsetting may produce undesirable distortion, and the resulting offset mesh may contain severe crimping, shearing, or poorly shaped quads, especially in regions of high curvature or around concave features, therefore, optimization-based quality improvement has been studied for both offset surfaces and surface meshes [4,5,11].

Quadrilateral meshes play an important role in geometric modeling and geometry processing, and their generation, optimization, and structural improvement have been studied extensively [3,9,10].

This article studies the problem of improving the quality of a quad-mesh offset after an initial offset mesh has already been constructed. The initial offset mesh can be obtained by shifting each vertex in its normal direction – cross product defined by its 4 neighbor vertices, or other prescribed offset direction, see Fig. 1. The key observation is that the initial offset positions provide the desired offset distance in a first approximation, but they do not necessarily produce a satisfactory offset mesh in terms of quad shape and size, or bad crimping (Fig 1 (a)). Instead of recomputing the offset from scratch, the present work introduces a global optimization model that adjusts the offset vertices tangentially while preserving the overall offset configuration.

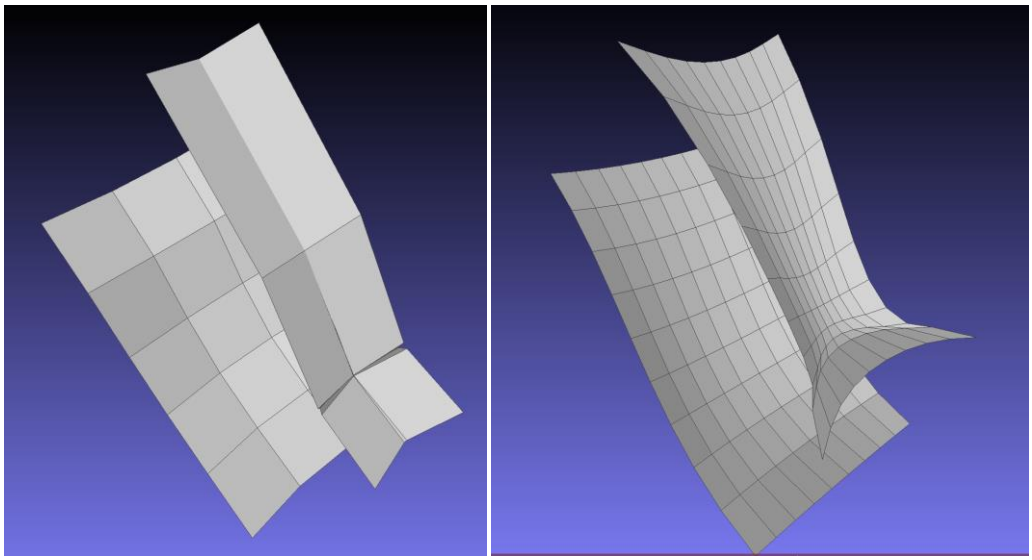

Figure 1: (a) left: 6x4 base mesh and its offset mesh; (b) right: 10x10 base mesh and its offset mesh

The method is an extension of the least-squares optimization idea previously developed by the author for two-dimensional polyline offset. In the two-dimensional setting, offset vertices are adjusted along tangent directions so that the offset polyline achieves a globally improved configuration. The present article extends that idea to quadrilateral surface meshes. The extension is nontrivial because each quad has four vertices and each offset vertex is allowed to move in a local tangent plane rather than along a single tangent direction. As a result, each quad contributes a local optimization problem involving eight scalar variables, and the global mesh optimization is assembled by coupling the local quad contributions through shared vertices.

The formulation begins with an initial offset mesh and represents the tangential correction at each offset vertex as a linear combination of two independent directions taken from the corresponding base-quad geometry. The optimization objective is then built from several local error measurements. The first group of terms attempts to preserve the shape and size relationship between a base quad and its offset quad, so that the optimized offset remains as consistent as possible with the local geometry of the original mesh. An additional center-consistency term constrains the average correction of the four quad corners and helps suppress crimping and shearing. By summing these local energies over all quads and minimizing the total energy, the method produces a global least-squares system whose solution yields an improved offset mesh.

The main contribution of this article is therefore a practical optimization framework for quad-mesh offset quality improvement. The method does not modify the initial offset distance directly; instead, it uses tangential vertex corrections to improve the local configuration of the offset mesh while retaining the intended offset structure. Because the formulation is expressed as a least-squares problem assembled over the whole mesh, it is suitable for iterative improvement and can be implemented efficiently.

The remainder of this article is organized as follows. Section 2 introduces the local quad formulation and the tangential correction model, and derives the least-squares objective function for a single quad as well as the global optimization system for the whole mesh. Section 3 presents implementation details and examples. Section 4 concludes the article and discusses possible extensions.

2 OPTIMIZATION

The present approach is closer in spirit to optimization-based mesh post-processing, in which an existing mesh is improved by minimizing a quality-related objective function rather than being reconstructed from scratch [5–8,12].

2.1 Initial Offset and Tangential Correction

Let $b_{i,j}$ denote a vertex of the base quad mesh, and let $p_{i,j}$ denote the corresponding vertex of an initial offset mesh, see Fig 2 (a). The initial offset mesh is assumed to have already been constructed, for example by moving the base vertices along prescribed offset directions. Although such an initial offset usually captures the intended offset distance, it may still contain severe local distortion such as shearing, crimping, or badly shaped quads.

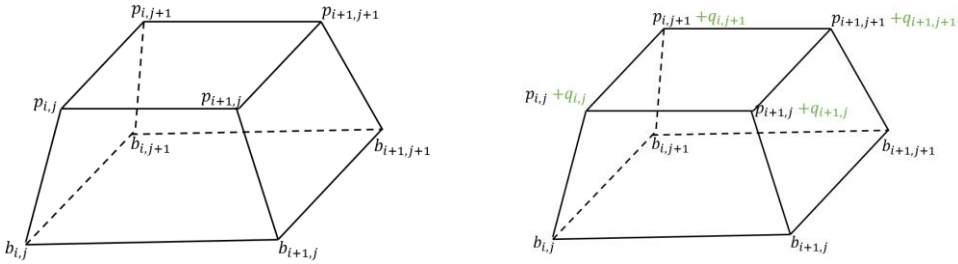

Figure 2: (a) a quad offset from bottom $b_{i,j}$ to top $p_{i,j}$; (b) adjust top from $p_{i,j}$ to $(p_{i,j}+q_{i,j})$

The purpose of the present optimization is therefore not to recompute the offset distance itself, but to improve the quality of the offset mesh by adjusting the offset vertices tangentially.

For each offset vertex $p_{i,j}$, let the optimized position be written as:

$$\hat{p}_{i,j} = p_{i,j} + q_{i,j}$$

where $q_{i,j}$ is a tangential correction vector. To preserve the original offset structure, the correction is restricted to two independent directions associated with the local base-quad geometry. Specifically, we write:

$$q_{i,j} = n_{i,j}\, v_{i,j} + t_{i,j}\, u_{i,j} \qquad (2.1)$$

where $t_{i,j}$ and $n_{i,j}$ are two non-parallel directions taken from the base quad, and $u_{i,j}, v_{i,j}$ are scalar unknowns to be solved by optimization.

Thus, each offset vertex contributes two scalar unknowns, and each quad contributes eight local unknowns through its four corner vertices.

2.2 Local Error Measurement Functions

As in optimization-based mesh smoothing and quality-improvement methods, local element-wise error measurements are defined and then accumulated into a global objective over the mesh [5,7,8,12].

For one quad of the base mesh, let the corresponding quad of the initial offset mesh be adjusted by the four tangential correction vectors defined in Eq. (2.1). The purpose of the optimization is to modify the offset quad so that it remains as consistent as possible with the

shape and size of the base quad, while still preserving the intended offset distance through tangential correction only.

To measure this consistency, three local error measurement functions are introduced for each quad. These terms are constructed from characteristic relative vectors of the base quad and the optimized offset quad. Their role is to penalize deformation of the offset quad with respect to the corresponding base quad, so that the optimized offset better preserves the local geometric structure of the original mesh.

More specifically, the three error terms compare selected corner-to-corner vectors of the optimized offset quad against the corresponding vectors of the base quad. In this way, both shape distortion and size distortion are reduced simultaneously. Although other definitions are possible, the present choice provides a simple least-squares formulation and works effectively in practice.

Let these three local error measurement functions be denoted by (see Fig 3 (b)):

$$f_{i,j+1},\ f_{i+1,j},\ f_{i+1,j+1}$$

where their explicit expressions are given in Eq. (2.2) and Eq. (2.3). Together, they form the main shape-preservation part of the quad energy.

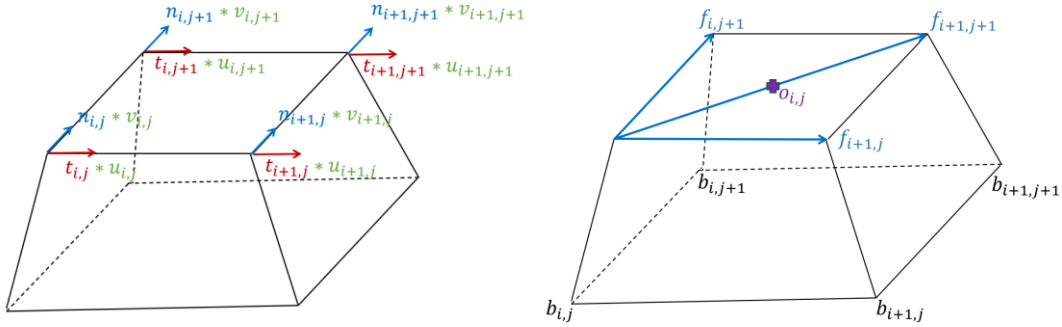

Figure 3: (a), left: adjustment $q_{i,j} = t_{i,j}\, u_{i,j} + n_{i,j}\, v_{i,j}$; same for $q_{i+1,j}, q_{i,j+1}, q_{i+1,j+1}$ (b), right: error measurement function $f_{i+1,j}$, $f_{i,j+1}$, $f_{i+1,j+1}$, $o_{i,j}$ (see definition below)

$$p_{xx} + q_{xx} = p_{xx} + n_{xx}\, v_{xx} + t_{xx}\, u_{xx}$$

where:

$$xx = (i,j), (i+1,j), (i,j+1), (i+1,j+1);$$

Also see Fig. 2 (b):

Define 3 error measurement functions f_{yy}, as below:

$$f_{yy} = (q_{yy} - q_{i,j} + c_{yy}) \tag{2.2}$$

where:

$$c_{yy} = (p_{yy} - p_{i,j}) - (b_{yy} - b_{i,j}) \tag{2.3}$$

$$yy = (i+1,j), (i,j+1), (i+1,j+1);$$

2.3 Center-Consistency Measurement

Preserving corner-to-corner relationships alone is not sufficient to prevent undesirable shearing or crimping of the offset quad. In particular, it is possible for an offset quad to satisfy part of the shape-preservation conditions while still becoming locally twisted or unbalanced.

To reduce this effect, an additional center-consistency measurement is introduced. For quad (i,j), define the average of the four corner correction vectors by

$$o_{i,j} = (q_{i,j} + q_{i+1,j} + q_{i,j+1} + q_{i+1,j+1})/4 \tag{2.4}$$

This quantity represents the average tangential displacement of the four corners of the offset quad. If the four corner adjustments become too unbalanced, the offset quad may exhibit excessive shearing or local crimping. The center-consistency term therefore acts as a stabilizing condition by constraining the average correction of the quad. In the present formulation, this term plays an important role in suppressing shearing and improving the overall offset quality.

Let the corresponding center error measurement be denoted by $o_{i,j}$,whose explicit definition is given in Eq. (2.4).

2.4 Quad Energy and Global Mesh Energy

Combining the three shape-preservation terms with the center-consistency term, the error measurement function for one quad is defined as:

$$E(i,j) = \| f_{i+1,j} \|^2 + \| f_{i,j+1} \|^2 + \| f_{i+1,j+1} \|^2 + \| o_{i,j} \|^2 \tag{2.5}$$

In the current implementation, all four terms are weighted equally. The first three terms are responsible for preserving the local shape and size relationship between the base quad and the offset quad, while the last term reduces local shearing and crimping by constraining the average quad correction.

Following the general spirit of optimization-based mesh quality improvement methods [7,12], the global error function for the entire quad mesh is defined as the sum of the local quad energies over all quads.

$$E = \sum_{i,j} E(i,j). \tag{2.6}$$

If the mesh contains $m \times n$ vertices, then the total number of quads is $(m-1)(n-1)$. Since each mesh vertex contributes two scalar unknowns through Eq. (2.1), the optimization problem contains $(2 * m * n)$ unknown variables in total. Because neighboring quads share vertices, the corresponding quad energies are coupled through the shared correction variables.

2.5 Least-Squares System for One Quad

For a single quad, the four corner vertices contribute eight scalar unknowns:

$$u_{i,j}, v_{i,j}, u_{i+1,j}, v_{i+1,j}, u_{i,j+1}, v_{i,j+1}, u_{i+1,j+1}, v_{i+1,j+1} \tag{2.7}$$

Since the correction vectors q are linear in these variables and the quad energy $E(i,j)$is defined as a sum of squared error terms, $E(i,j)$is a quadratic function of the eight local unknowns. Therefore, minimizing the total mesh energy reduces to a linear least-squares problem.

For each quad, the local normal equations are obtained by taking the partial derivatives of the total energy with respect to the eight local variables and setting them equal to zero. This produces eight linear equations associated with the quad. In implementation, these equations are written in matrix form, with the coefficient entries assembled directly from the derivative expressions of the local error function.

Because the same mesh vertex may belong to several neighboring quads, the corresponding variables appear repeatedly in different local quad equations. This is exactly what couples the optimization globally across the entire mesh.

3 IMPLEMENTATION

3.1 Local Matrix Form and Assembly of the Global Linear System

After the local quad energy functions have been defined, the optimization problem is solved by minimizing the total mesh energy in Eq. (2.6). Since each local correction vector $q_{i,j}$ is linear in the scalar variables $u_{i,j}$ and $v_{i,j}$ and each local quad energy is a sum of squared error terms, the total energy is a quadratic function of all unknown variables. Therefore, the minimization reduces to a linear least-squares problem.

For a single quad, the four corner vertices contribute eight scalar unknowns, as shown in (2.7)

The local normal equations are obtained by taking the partial derivatives of the total energy with respect to these eight variables and setting them equal to zero. This produces an 8×8 local linear system for the quad, together with a corresponding right-hand-side vector. In the present formulation, the coefficients of this local system are determined directly from the derivative expansion of the quad error measurement functions.

For clarity, the local derivative equations are written in matrix form as

$$A^{(quad)} x^{(quad)} = b^{(quad)}$$

where $x^{(quad)}$ is the vector of the eight local unknowns associated with $quad$, $A^{(quad)}$ is the corresponding local coefficient matrix, and $b^{(quad)}$ is the local right-hand-side vector. The coefficient pattern used in the implementation is shown in Table 1.

derivative	times	$q_{i,j}$	$q_{i+1,j}$	$q_{i,j+1}$	$q_{i+1,j+1}$	right-side-column
$\frac{\partial E_{i,j}}{\partial u_{i,j}}$	$t_{i,j}$	13/4	-3/4	-3/4	-3/4	$c_{i+1,j} + c_{i,j+1} + c_{i+1,y+1}$
$\frac{\partial E_{i,j}}{\partial v_{i,j}}$	$n_{i,j}$	13/4	-3/4	-3/4	-3/4	$c_{i+1,j} + c_{i,j+1} + c_{i+1,y+1}$
$\frac{\partial E_{i,j}}{\partial u_{i+1,j}}$	$t_{i+1,j}$	-3/4	5/4	¼	1/4	$-c_{i+1,j}$
$\frac{\partial E_{i,j}}{\partial v_{i+1,j}}$	$n_{i+1,j}$	-3/4	5/4	¼	1/4	$-c_{i+1,j}$
$\frac{\partial E_{i,j}}{\partial u_{i,j+1}}$	$t_{i,j+1}$	-3/4	1 /4	5/4	1/4	$-c_{i,j+1}$
$\frac{\partial E_{i,j}}{\partial v_{i,j+1}}$	$n_{i,j+1}$	-3/4	1 /4	5/4	1/4	$-c_{i,j+1}$
$\frac{\partial E_{i,j}}{\partial u_{i+1,j+1}}$	$t_{i+1,j+1}$	-3/4	1 /4	¼	5/4	$-c_{i+1,y+1}$
$\frac{\partial E}{\partial v_{i+1,j+1}}$	$n_{i+1,j+1}$	-3/4	1 /4	¼	5/4	$-c_{i+1,y+1}$

Table 1: Local coefficient pattern contributed by one quad during assembly of the global sparse linear system. Each quad contributes eight scalar unknowns (two per corner).

For each quad, the eight derivative equations obtained from the local normal equations can be written in matrix form as a local 8×8 linear system. The coefficient pattern of this system is fixed for the present formulation and is shown in Table 1. In implementation, these local coefficient contributions are assembled directly into the global sparse system by mapping the local quad variables to the corresponding global vertex-based unknowns.

Because neighboring quads share vertices, their local unknowns are not independent. When the local systems of all quads are accumulated, the coefficients associated with shared vertex variables are assembled into a single global linear system.

$$Ax = b$$

where x contains all scalar correction variables $u_{i,j}$ and $v_{i,j}$ over the mesh.

The global matrix A is sparse, because each quad contributes only to the variables associated with its own four corner vertices. Equivalently, each quad contributes a local block of coefficients to the rows and columns corresponding to those eight variables. In the present implementation, each quad contributes a fixed local 8×8 coefficient block with 72 nonzero scalar entries. These entries overlap with contributions from neighboring quads at shared vertices, and their accumulation over the mesh forms the global sparse system.

```
{
  // build working matrix
  int total = (_ht * _wd) * 2; // for both u & v
  _mxOp.initMatrix(total, total, _M);
  _B.resize(total);
  // for each input point set up a row in matrix
  for (int j = 0; j < (_ht-1); j++) {
    for (int i = 0; i < (_wd-1); i++) {
      vector<vector<double>> oo; // 8 X 9 matrix
      _compDuDv8(i, j, oo);
      int p0 = (j * _wd + i) * 2;  // 2 for both u & v (low-left-corner)
      int p1 = p0 + _wd * 2; // (up-left-corner)
      // first 4 rows for left-side & right-side
      for (int k = 0; k < 4; k++) {
        vector<double >& d = _M[p0 + k];
        vector<double >& o = oo[k];
        for (int j = 0; j < 4; j++) {
          d[p0 + j] += o[j];
        }
        for (int j = 0; j < 4; j++) {
          d[p1 + j] += o[j+4];
        }
        _B[p0 + k] += o[8];
      }
      // second 4 rows for left-side & right-side
      for (int k = 0; k < 4; k++) {
        vector<double >& d = _M[p1 + k];
        vector<double >& o = oo[k+4];
        for (int j = 0; j < 4; j++) {
          d[p0 + j] += o[j];
        }
        for (int j = 0; j < 4; j++) {
          d[p1 + j] += o[j+4];
        }
        _B[p1 + k] += o[8];
      }
    }
  }
  bool ret = true;
}
```

```
{
  _init(pnts, dist);
  // setup for M*u = B
  _setupMB();
  vector< vector<double> > mm = _M;
  bool ret = _mxOp.PivotInverse(mm, _iM);
  ret = _mxOp.verifyInv(_M, _iM, mt_global_epsilon);
  // below contains u & v
  vector<double> uv = _mxOp.multiply1D21(_iM, _B);
  int total = _wd * _ht;
  _uu.resize(_ht); _vv.resize(_ht);
  for (int k = 0; k < _ht; k++) {
    vector<double>& u = _uu[k]; u.resize(_wd);
    vector<double>& v = _vv[k]; v.resize(_wd);
    for (int j = 0; j < _wd; j++) {
      int id = (k * _wd + j) * 2;
      u[j] = uv[id]; v[j] = uv[id + 1];
    }
  }
  resize_matrix(_ht, _wd, qnts);
  for (int k = 0; k < _ht; k++) {
    for (int j = 0; j < _wd; j++) {
      Vec3 p = _PP[k][j] + _TT[k][j] * _uu[k][j] + _NN[k][j] * _vv[k][j];
      qnts[k][j] = p;
    }
  }
  pp = _PP;
  return (true);
}
```

Figure 4: (a) left: C++ code for all quads; (b) right: C++ main function

Once the global system is assembled, it is solved for the vector x of tangential correction coefficients. The optimized offset mesh is then obtained by substituting these coefficients back into Eq. (2.1) to recover the correction vector $q_{i,j}$ at each offset vertex.

The matrix form shown in Table 1 corresponds directly to the implementation used in the numerical examples. The C++ code shown in Fig. 4 is therefore not a separate heuristic

procedure, but a direct realization of the least-squares system derived from the quad energy function.

Figure 4 shows the actual corresponding implementation used to assemble the global system and solve for the correction coefficients.

3.2 Iterative Application

Once the linear system is solved, the optimized offset mesh is obtained by updating each offset vertex with its tangential correction vector. The same procedure may then be applied again by treating the updated offset mesh as the new current offset mesh. In this way, the method can be used iteratively to further improve quad quality.

In the examples presented in this article, repeated application of the optimization significantly reduces crimping and improves the shape quality of degenerated quads. In practice, only a small number of iterations is typically needed to obtain a visibly improved offset mesh.

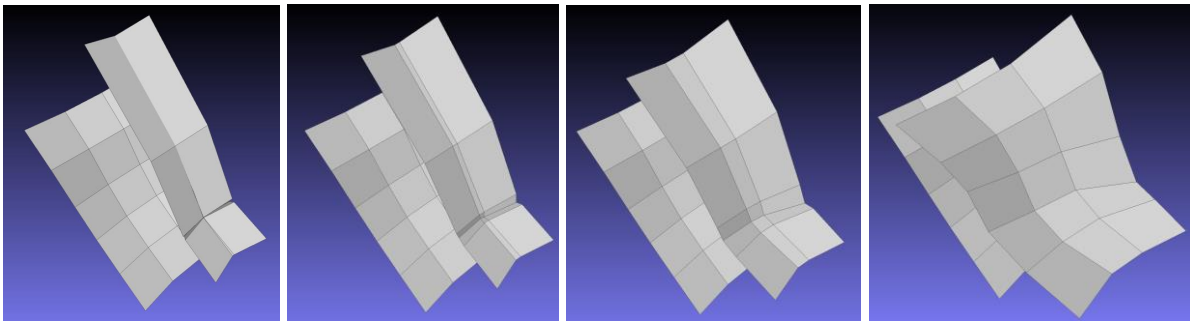

Figure 5: Iterative improvement of the offset mesh: (a) initial offset mesh; (b) result after one optimization step; (c) result after two optimization steps; (d) result after four optimization steps

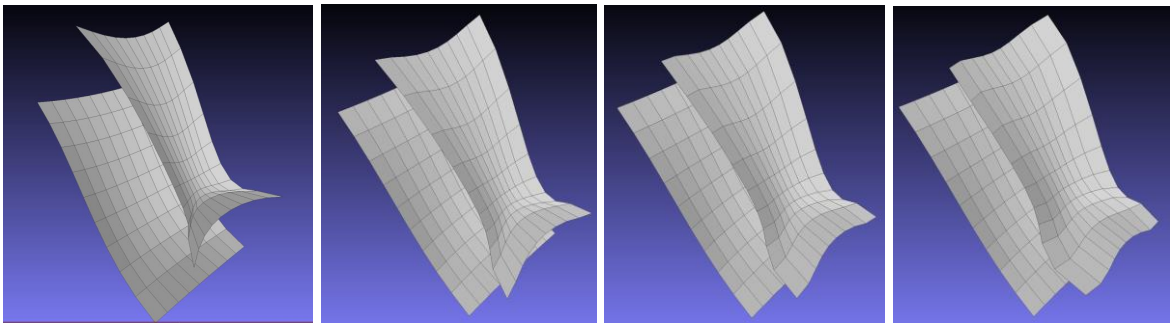

Figure 6: Iterative improvement of the offset mesh: (a) initial offset mesh; (b) result after one optimization step; (c) result after two optimization steps; (d) result after four optimization steps

For iterative application, the optimized offset mesh obtained from one solve is used as the current offset mesh for the next iteration.

Several variations of the present formulation may further improve performance or offset quality. For example, alternative diagonal constraints may be used in the local error terms, more edge-based directions may be incorporated into the quad energy, and a reprojection step to the base mesh may be introduced after each iteration. These extensions are not pursued here, since the present article is intended primarily to establish the basic least-squares optimization framework for quad-mesh offset improvement. They may be investigated in future work.

4 CONCLUSION

The present method complements existing work on polygonal offsetting and quadrilateral mesh optimization by focusing specifically on tangential least-squares improvement of an already constructed quad-mesh offset [3,4,7].

The present work introduces a least-squares optimization model for improving the quality of quadrilateral surface offsets. Starting from an initial offset mesh, the method improves offset quality by introducing tangential correction vectors at offset vertices and solving for their coefficients globally. The optimization is constructed from three local shape-preservation terms together with a center-consistency term, so that the optimized offset quad remains as consistent as possible with the shape and size of the corresponding base quad while reducing local crimping and shearing.

Because each correction vector is expressed in terms of two scalar variables, the quad-mesh optimization can be assembled into a global sparse linear system. The method is straightforward to implement and can be applied iteratively. The examples show that it can significantly improve offset quality for quad meshes with visibly distorted initial offsets.

The main contribution of this work is therefore a practical extension of the author's least-squares offset optimization idea from two-dimensional polylines to three-dimensional quadrilateral meshes. Possible future work includes alternative local error terms, improved directional choices for tangential correction, and extension of the same optimization framework to other surface offset settings.

REFERENCES

[1] Hoschek, Josef; Lasser, Dieter: Fundamentals of Computer Aided Geometric Design, AK Peters, Ltd, 1993.

[2] Piegl, Les; Tiller, Wayne, The NURBS Book, Springer,, 1995.

[3] Bommes, D.; Lévy, B.; Pietroni, N; Puppo, E; Silva, C; Tarini, M; Zorin, D: Quad-mesh generation and processing, A survey, *Computer Graphics Forum*, 32(6), 2013, pp. 51-76.

[4] Meng, W.; Chen, S.; Shu, Z.; Xin, S.-Q.; Fu, H.; Tu, C.: Efficiently computing feature-aligned and high-quality polygonal offset surfaces, *Computers & Graphics*, 70, 2018, pp. 62-70.

[5] Garimella, R. V.; Shashkov, M. J.; Knupp, P.M.: Triangular and quadrilateral surface mesh quality optimization using local parametrization, *Computer Methods in Applied Mechanics and Engineering*, 193(9–11), 2004, pp. 913–928.

[6] Canann, S. A.; Tristano, J. R.; M. L. Staten: An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral and quad-dominant meshes, in *Proceedings of the 7th International Meshing Roundtable*, 1998, pp. 479–494

[7] Knupp, P. M.: Smoothing by optimization for quadrilateral mesh generation, *Finite Elements in Analysis and Design*, 34(3–4), 2000, pp. 181–195.

[8] Natarajan, V.; Subburaj, K.; Raghavan, T.: A 3D constrained optimization smoother to post-process quadrilateral meshes for body-in-white, *Procedia Engineering*, 163, 2016, pp. 262–1998.

[9] Zhao, Z.; Fang, S.; Lei, N.; Liu, Y.; Zhu, Y.: Optimal surface quadrilateral mesh generation, in *Proceedings of the 32nd International Meshing Roundtable*, SIAM, 2024, pp. 15–34.

[10] Li, X.; Qian, X.; Yong, J.-H.; Zheng J.; Liu, Y.; Zhang, H.: Optimal base complexes for quadrilateral meshes, *Computer Aided Geometric Design*, 52–53, 2017, pp. 105–120.

[11] Portaneri, C.; Rouxel-Labbé, M.; Hemmer, M.; Cohen-Steiner, D.; Alliez, P.: Alpha wrapping with an offset, *ACM Transactions on Graphics*, 41(4), 2022, Article, 105.

[12] González, G. F. Flores; Barrera, P. Sánchez: New quality measures for quadrilaterals and new discrete functionals for grid generation, *Mathematical and Computational Applications*, 28(5), 2023, Article 95.