A Projective Framework for Polyhedral Mesh Modeling

Amir Vaxman†

1Institute of Discrete Mathematics and Geometry, TU Vienna

Figure 1: Subdivision of planar-quad meshes (left) and editing of polyhedral meshes with map prescription (right).

Abstract

We present a novel framework for polyhedral mesh editing with face-based projective maps, that preserves planarity by definition. Such meshes are essential in the field of architectural design and rationalization. By using homogeneous coordinates to describe vertices, we can parametrize the entire shape space of planar-preserving deformations with bilinear equations. The generality of this space allows for polyhedral geometric processing methods to be conducted with ease. We demonstrate its usefulness in planar-quadrilateral mesh subdivision, a resulting multi-resolution editing algorithm, and novel shape-space exploration with prescribed transformations. Furthermore, we show that our shape space is a discretization of a continuous space of conjugate-preserving projective transformation fields on surfaces. Our shape space directly addresses planar-quad meshes, on which we put a focus, and we further show that our framework naturally extends to meshes with faces of more than four vertices as well.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Picture/Computational Geometry and Object Modeling—Polyhedral Shape modeling, Shape editing, Shape space, PQ meshes, design exploration

1. Introduction

Polygonal meshes with planar faces, denoted as Polyhedral meshes, have attracted recent attention, mainly due to their benefits in industrial and architectural design. Planar plates are considerably easier to produce than general continuous surfaces, using materials like glass or wood. A designing tool for polyhedral mesh editing is expected to allow the prescription of positional, rotational, or scaling transformations to components of the mesh, and compute the resulting mesh, conforming to the planarity constraints. In addition, a designer may prefer to explore through a space of possible constrained meshes instead of adhering to one result. These meshes are also expected to accommodate fairness measures, like smoothness over curves, or maintaining the similarity of faces. Mesh editing generally begins by sculpting the general shape of the objects, usually with a coarse mesh, and then adding small details and refining. Therefore, it is usually beneficial to edit complicated meshes in a multi-resolution manner, where coarse editing operations that act on a low-complexity mesh are propagated onto a fine mesh, and thus save time and space. Multi-resolution editing either begins with the simplification of a base fine mesh, or the
subdivision of a coarse one. In this work, we focus on the latter.

Unfortunately, the planarity constraint of a polygon is a third-order polynomial in the position of its vertices. Thus, it is difficult to efficiently define and generalize editing methods that are commonly practiced with well-established formulations within the trivially-planar subclass of triangular meshes. Therefore, it makes sense to search for alternative surface representations which reduce the load of such constraints. In this paper, we show that \textit{projective maps} are an attractive tool to work with in this context. Projective maps are linear transformations acting on homogeneous coordinates (and are therefore \textit{linear-fractional} transformations in the Euclidean space). Every triangle can be mapped to every other triangle with an affine map, and every planar quad, in general position, can be mapped into any other planar quad using a single nonsingular projective map. General position in this context means that no three points are collinear. With a set of compatibility conditions, bilinear in the homogeneous coordinates and a scale factor, we can define a shape space of planarity-preserving transformations for planar-quad (hereby abbr. PQ meshes), and meshes of polygonal faces bigger than 4 vertices each (that we hereby denote as \textit{higher-order faces and meshes}). In practice, we often deal with near-polyhedral meshes, in which faces are only approximately flat. The geometric operations we define preserve the relative planarity of such meshes, by regularizing the projective maps in a simple manner. We measure the planarity of a quad by the percentage of the absolute distance between its diagonals and the average diagonal length. A value of 1% is the usual acceptable limit. The planarity of a higher-order face is typically measured by an average of the (absolute) planarity values of all consecutive four vertices.

1.1. Our Contributions

We present a general technique of working with compatible projective maps that are assigned per face, and develop the following novel designing tools:

- Subdivision of PQ meshes.
- Multi-resolution editing of PQ meshes.
- Shape exploration by prescription of special classes of deformations.
- Interpolation.

Our framework generalizes existing methods for triangular and polygonal surfaces, burdening them with mere alternating linear constraints on homogeneous coordinates. Moreover, we show that our discrete formulation has a meaningful theoretical basis in continuous projective differential geometry. We base and exemplify our work on the important case of PQ meshes, for which our shape space is trivially complete, and for which subdivision methods are readily available in the literature. However, our framework applies for higher-order faces of polyhedral meshes, and we exemplify such editing capabilities as well.

2. Related Work

Modeling with Polyhedral Meshes. Applications of polyhedral non-triangular meshes have been commonly studied in recent literature within the context of architectural geometry. Liu \textit{et al.} [LPW’06] obtained PQ meshes by a process of nonlinear optimization, which involved alternating Catmull-Clark subdivision and planarization. Other remeshing methods [ZSW10, LXW’11] established conjugate vector fields on triangular surfaces in order to create PQ(-dominant) meshes by parametrization and subsequent optimization. Editing polyhedral meshes from existing ones by means of deformation is studied in several contexts. The linear shape space of \textit{parallel meshes}, where all meshes are edge-wise parallel to a given base mesh, was studied in [PLW’07] for the purpose of multi-layered construction. The subspace of \textit{mesh offsets}, where component-wise (e.g. vertex) distances between parallel meshes are equal, were given the greater attention, as they corresponded nicely with notions in continuous differential geometry, such as curvature [BPW10], minimal and constant mean-curvature surface [PLW’07, Mue11]. Meshes with offsets are also identified with \textit{circular} and \textit{conical} meshes [BS08a], and as discrete versions of curvature-line parametrizations. However, parallel and offset meshes form a rather restrictive shape space, used mostly for transforming meshes with low vertex degree (the ideal meshes in this group are hexagonally-dominant 3-webs [Mue11, WLY’08]). Triangular meshes usually induce only trivial parallel meshes, by uniform scaling [PW08].

Other editing methods for polyhedral meshes rose from a more general context of editing constrained meshes. In [YYPM11], a nonlinear shape space is viewed as a Riemannian manifold, defined by the intersection of the zero sets of local constraints (for instance, face planarity). This shape space is explored locally by computing the closed-form tangent space and second-order osculatant at any given base point on the manifold. The exploration of the environment of the base mesh then leads to a bounded deviation from the original shape space, which can be alleviated by nonlinear projections. Several approaches attempted to handle the burden of global nonlinear optimizations by finding a local and sparse set of moving vertices for every editing operation [HK12, DBD’13, ZTY’13], and reduce the problem accordingly. The generality of these approaches still results in applying nonlinear optimization methods. Another approach to handling nonlinear constraints is by applying a local-global scheme [BDS’12, SA07]. Such a scheme alternates between a local step, in which every constrained element (e.g. a face that must be planar) is projected individually into an ideal element which is similar to the original, and a global step, which integrates the mesh by a least-squares method. While being quite effective, this method does not guarantee the fulfillment of exact constraints in every step (albeit providing good convergence results), and it does not...
provide any meaningful parametrization of the polyhedral shape space.

We specifically address the special and interesting case of polyhedral meshes by defining a simple space. A recent approach [Vax12] defines a linear shape space of polyhedral meshes by assigning affine maps per face that are compatible at the edges of the mesh. These compatible maps produce a mesh deformation that preserves planarity exactly, and represents a meaningful linear subspace of the entire polyhedral shape space. Their method allows for easy as-rigid-as-possible [SA07] and as-similar-as-possible [LZX08] deformations, and exact shape space exploration. In addition, their method is a generalization of triangular deformation methods relying on deformation gradients (surveyed in [BS08b]) and frame-based methods (such as [LSLCO05]). Unfortunately, the class of affine transformations of faces, while parametrizing the entire space of triangle-mesh deformations, is still rather prohibitive and unsuitable for applications such as polyhedral subdivision (see Figure 2) or for modeling higher-order faces. Interesting enough, this is exactly the complementary disadvantage relative to parallel meshes. Our projective framework, for which the affine framework is a subspace, allows for greater freedom in modeling, while not being considerably more difficult to implement, as we iteratively alternate between linear subspaces of the full polyhedral shape space.

**Multi-Resolution Mesh editing.** Editing meshes with subdivision topology can be done by creating a subdivision hierarchy of meshes, coarsening positional and otherwise constraints made on a finer mesh, deforming a coarse mesh and then refining the results with fairness energy minimization. The survey [BMZB05] summarizes recent advances in this subject. However, other common methods prefer to create the coarse-to-fine editing either by surrounding the object with a coarse cage and transforming its inner ambient space (e.g. [JMD07]), or by simplifying a given mesh, and inverting such simplifications upon refining [KCVS98, SYBF06, SSP07, MS11]. We base our work on subdivision surfaces, but apply a similar approach as the latter work [MS11], where rigid or otherwise deformations, that are conducted on a coarser version of the mesh, are propagated by some manner of prescription to finer levels.

3. The Projective Shape Space

Given a source polyhedral base mesh $M = \{V,E,F\}$, we express every source vertex position in $\mathbb{R}^3$ as $p_i = (x_i, y_i, z_i)$, with a trivial extension to homogeneous coordinates $\overline{p}_i = (x_i, y_i, z_i, 1)$. We use $\overline{p}_i \in \mathbb{R}^4$ to represent any point in the projective space $P^3$ on the line $(w_1 x_i, w_1 y_i, w_1 z_i, w_1)$, projectively-equivalent to $p_i$. Given two quadrilateral planar faces $f = (\overline{p}_1, \overline{p}_2, \overline{p}_3, \overline{p}_4)$, $f' = (\overline{p}_1', \overline{p}_2', \overline{p}_3', \overline{p}_4')$, it is well known that if both are planar, there is a projective map, represented by a nonsingular matrix $A_f \in \mathbb{R}^{4 \times 4}$, so that:

$$\overline{p}_i A_f = \overline{p}_i', \quad 1 \leq i \leq 4 \quad (1)$$

Four points on a two-dimensional plane, no three of which are collinear, uniquely define the transformation of that plane (and any quad within). Moreover, the projective transformation $f \rightarrow f'$ defines matrix $A_f$ up to a uniform scale, i.e., $A_f, w A_f, w \neq 0 \in R$ define the same transformation. Our ambient space is $P^3$, which is a four-dimensional homogeneous space, yet we act on planar quads, each residing in a $P^2$ three-dimensional homogeneous space. Therefore, our transformations have redundant degrees of freedom that manifest in the way that points outside of the quad planes transform, not affecting the quads themselves. We exploit these degrees of freedom in Section 6. An important resulting property is that the $w_i$ (homogeneous weight) values in $\overline{q}_i$ are totally determined (again, up to a global scale) by the geometric positions of $q_i$. For instance, when all $w_i = 1$, the map is affine, and $A_{41} = A_{42} = A_{43} = 0, A_{44} = 1$ accordingly.

The complete shape space of PQ deformations, stemming from the given base mesh $M$, is then defined as the collection of face-based projective maps, such that vertices that are shared between two adjacent maps are projectively equivalent. This space can be parametrized by assigning two extra scalar variables $s_{f,i}, s_{g,i}$ for two faces $f, g$ sharing a vertex $p_i$. 

Figure 2: The approximative subdivision of a cube and the pipes meshes in several methods. Since our method shares the local-global mechanisms of $[BDS12]$, we compare iterations and final planarity measures in total (summing over all subdivision levels). We stopped iterating when the maximum planarity went below 0.1% in the cube example, and 1.0% in the pipes example. It is clear that we get comparable and intuitive results to the nonlinear method $[LPW06]$ and to Shape-up $[BDS12]$, and improve upon planarity with less iterations, since our projective maps preserve planarity by nature. In both cases, affine maps are not rich enough to conduct this subdivision, and the closest possible affine maps $[Vax12]$ are the trivial ones (identity maps which left a trivial subdivision).
In this work, we utilize normalized matrices, so that:

\[
\begin{align*}
    s_{f,j} \cdot \mathbf{p}_i A_f = s_{g,j} \cdot \mathbf{p}_k A_g = (q_i, 1),
\end{align*}
\]  

which is evidently quadratic in homogeneous coordinates.

In this work, we utilize normalized matrices, s.t. \( A_{4,4} = 1 \). Note again that this does not restrict the shape space, since projective matrices are defined up to a global scaling factor.

**Linear Subspaces:** Upon any prescribed scale factors \( \{s_{f,j}\} p_i \in f, f \in F \), our shape space, parametrized by the set of compatible face-based projective maps \( A = \{A_f|f \in F\} \), becomes linear. The constraints in 2 can then be summarized in a linear matrix form: \( S = \{A \mid C_i \cdot A = d_i\} \), where \( A \in R^{16[F]} \) is a column vector of all face-based transformations \( A_f \) by order. Any positional constraint imposed on the mesh can be easily incorporated into this linear subspace: if a vertex \( p_i \), adjacent to a face \( f \), is constrained to transform to \( q_i \), then we add the constraint \( s_{f,j} \cdot (p_i, 1) A_f = (q_i, 1) \).

### 3.1. General Polyhedral Meshes

It is possible to edit general polyhedral higher-order meshes with a single projective map per face, using the same framework. We employ this solution in our work. However, a single projective map per face only parametrizes a subset of the possible planarity-preserving transformations of higher-order faces. A possible solution is to assign several compatible maps per face, so that every map covers four vertices, and every vertex is covered by at least one map. This however is out of our scope. We do not address the subdivisions of general meshes because we focus on quad-based subdivision, yet we utilize higher-order meshes in the map-prescription editing of section 7.

### 4. Editing Methodology

Within our framework, we define two types of operations which we use interchangeably in our applications: a local step that fits a local per-face projective transformation from a source face to a target face, and a global step that projects given prescribed transformations (such as those computed with the local step) onto the polyhedral shape space.

#### 4.1. Local Step

Given a face \( f \), comprising source vertices \( p_i, i \in [1 \ldots 4] \), and desired vertex positions \( q_i \), we attempt to find the best fitting projective transformation. By introducing scaling coefficients \( s_{f,j} \) as variables, a perfectly fitting projective transformation holds:

\[
\begin{align*}
    s_{f,j} (p_i, 1) T_f - (q_i, 1) = 0.
\end{align*}
\]

These equations are linear with regards to the reciprocal to \( s_{f,j} \): \( s_{f,j} = \frac{1}{s_{f,j}} \). In order to avoid degenerate solutions, we always use normalized transformation matrices in which \( T_{4,4} = 1 \). We identify two cases of importance rising from this local formulation:

- When \( p_i \) and \( q_i \) both form perfectly planar faces, the null space to these equations spans the space of ambient transformations that transform \( p_i \) in face \( f \) into \( q_i \) exactly (and differ on their action on points in the ambient space). In these cases, we try to find the valid transformation closest to the identity matrix (projecting the identity matrix on the null space can be easily achieved by the QR-factoring of Equation system 3).

- When \( p_i \) and \( q_i \) are only nearly planar or worse, there is always a projective transformation between them, since we work in the ambient space \( P^3 \). This case might lead to solutions that are correct, but unwanted. We therefore try to avoid it by regularizing the input in our applications.

All other possible cases (e.g. planar quad \( p \) to nonplanar \( q \)) may lead to an overdetermined system which would then be solved in a least-squares manner.

#### 4.2. Global Step

Given a set of (normalized) “ideal” face-based maps \( T_f \), we attempt to project them onto the polyhedral shape space. In order to avoid direct projection on the entire quadratic shape space, we approximate the result by reducing to a linear subspace. This subspace is defined by setting the \( s_{f,j} \) values, induced on the original coordinates by the prescribed \( T_f \) in Equation 3, as constant. We can then construct the matrices \( C_i, d_i \) as in Section 3, and proceed by solving the following convex problem:

\[
\begin{align*}
    A = \text{argmin}_f \sum_f |T_f - A_f|^2, \text{ s.t. } C_i A = d_i,
\end{align*}
\]

using the Frobenius norm. This is a quadratic minimization problem with linear constraints that can be solved in a single matrix form. The linear conditions \( C_i A = d_i \) might be over-constrained in certain configurations. Solving directly would then lead to a least-squares solution. However, we found that in practice it is more stable numerically to introduce a small tolerance \( \varepsilon \), so that we solve for \( d_i - \varepsilon \leq C_i A \leq d_i + \varepsilon \), opting for the smallest \( \varepsilon \) for which these linear-bound constraints are satisfied. Fortunately, we can play with the \( \varepsilon \) value quite efficiently within an iteration, since the matrices are constant throughout this step and can be prefactored, and since the optimization problem remains convex. It is also evident that this algorithm converges immediately when both meshes are perfectly polyhedral to begin with (since the local maps are compatible, and thus form a trivially-integrable shape), and that it parametrizes the target mesh by the projective maps, with relation to the first (base) mesh, which is useful in our applications. We next utilize our editing methodology for several editing applications within the shape space.
5. PQ Mesh Subdivision

Common mesh subdivision methods can be cast as deformation problems: the mesh is first subdivided in a trivial manner, and then the vertices are deformed within the new topology. Trivial subdivision of planar quads is made by simply connecting the mid-edge points to the mid-face point in the standard quad-refinement step used by primal subdivision methods. It is clear that near-planarity is also preserved by this averaging. Our method proceeds by trying to imitate a given linear (or otherwise) subdivision. The vertices of the trivially-subdivided mesh are extended as \((p_i, 1)\), and the vertices of the target linearly-subdivided mesh as \((q_i, 1)\). We proceed with the following algorithm within the editing methodology of Section 4:

1. For each face \(f\), project local points \(p_{f,i}\) (resp. \(q_{f,i}\)) on the best-fitting plane.
2. **Local step:** find the set of local “ideal” projective transformations \(\{T_f\}\) that transform between the planarized faces, and extract the induced scale factors \(s_{f,i}\).
3. **Global step:** construct \(C_i, d_i\) and project \(T_f\) onto the compatible projective-map set \(\{A_f\}\) to find the new mesh. Re-use new mesh positions as \(q_i\) points for the next iteration.
4. If the planarity bound is met, and if the mesh has not changed much from a previous iteration (we usually demand \(\max|V^k - V^{k-1}| < 10^{-3}\), terminate. Otherwise, re-iterate.

The reason for using planarized faces instead of the originals is to avoid the ambient projective problem mentioned in Section 4.2. In the global step (3), the \(\varepsilon\) value is modified while solving the convex functional: it is first set to be a fraction of the constraint error induced by the \(T_f\) computed in the local step (2). If this global solver fails (overconstrained), then \(\varepsilon\) is successively increased until the solver succeeds. Though we do not supply any theoretical proof of convergence, it is evident from the results in Figures 3 and 4 that this algorithm performs well in practice. Our model linear subdivisions are the tensor-product four-points scheme [Kob96] for interpolatory subdivision, and the Catmull-Clark scheme [CC98] for approximate subdivision. In all our experiments, we usually begin with the initially-induced \(\varepsilon = 10^{-2}\max|C_i T_f - d_i|\), and expand it by multiplications with 10.

6. Multi-Resolution Editing

With the algorithm of the previous section, we subdivide an initial coarse polyhedral mesh \(M_0\) into several finer meshes \(M_1, \ldots, M_n\). Next, we devise a multi-resolution editing algorithm, which intuitively propagates projective maps from coarse meshes to fine meshes, for the sake of efficient mesh editing. A depiction summarizing our algorithm, as we detail it further, is in Figure 5.

(a) Several subdivision levels for the Aquadom mesh. Notice how a mildly nonplanar face is planarized throughout the subdivision by the regularization.

(b) Subdivisions of the Half-tunnel mesh.

Figure 3: Approximative subdivisions.

(a) Subdivision of the pipes mesh.

(b) The Six Model.

Figure 4: Interpolative subdivisions. The upper planarity limit is relaxed to 0.5% to allow for more smoothness, as interpolative subdivisions are more constrained by nature.

6.1. Single-Level Deformation

In our setting, the user edits a single fine mesh \(M_n\), whereas transformations are done on a coarser level \(M_k, k \leq n\). The reasoning for our algorithm is straightforward and intuitive: we assume that the user solely cares for the result in the finest mesh, but wishes to edit the mesh in the coarsest level possible, in order to optimize the editing speed (see Table 1 for comparisons). For clarity, we base our definition on interpolatory subdivision, where a vertex \(p_i\) in mesh \(M_k\) also exists in all \(M_l, l > k\) (with the same index), and then extend the algorithm to handle approximative subdivisions in Section 6.5. Throughout this section we denote the set of all quads that stem from the (recursive) refinement of a single quad as its “progeny”. This quad is further denoted as the “ancestor” of this set. The basic algorithm for single-level deformation takes after the face-based affine maps described in [Vax12], and based on as-similar-as-possible methods, such as [SA07], extended to multi-resolution editing in the spirit of [MS11]. We summarize its steps for completeness:
Figure 5: The steps of the full multi-resolution algorithm, from upper-left in clockwise order: Choosing ROI and handles on the fine level, coarsening fine fixed vertices to match fixed vertices on the coarse mesh, and fine handles to positions in the ambient space. The handles are attached to the ancestors of the fine quads (purple). Next, deforming the coarse mesh, and then prolonging the deformation, while keeping all the constraints of the fine mesh in position.

1. Establish Region-of-interest (ROI) and editing handles to form positional constraints.
2. Alternate between the next two steps until convergence is gained (measured by difference in maximal vertex displacement between two consequent iterations):
3. **Local step:** Find an “ideal” map $T_f$ that matches the desired property exactly (e.g., a rigid or conformal transformation), by first computing the best fitting projective map via Equation 3, and then the closest ideal map (in least-squares) to it.
4. **Global step:** Solve for the global projection step, while maintaining the positional constraints.

The major difference between our basic editing algorithm and that of [Vax12], is that we use projective maps instead of affine maps, which gives us more freedom. Notice, however, that this freedom is of little use when rigid or similar transformations are sought after, as these are subclasses of affine maps anyhow, and this extension is not in full capacity. The projective framework is however essential for the multi-resolution deformation, since it is the basis of the subdivision algorithm. The computation of the ideal matrices $T_f$, derived in [Vax12], is summarized in Appendix A. We next explain how this basic algorithm is modified in our multi-resolution setting.

### 6.2. Coarsening Constraints

The user should be able to operate on the finest mesh while being oblivious to the coarser level in which the deformation is actually carried. Therefore, the fixed (non-ROI) positions and the handles chosen do not generally correspond to existing coarse vertices. However, since our face-based transformations are defined in $P^3$, and therefore not confined to their respective original quad plane, we can practically constrain the transformation of every point in the ambient space with the projective map of a given nearby face (see Figure 5). We choose this face as the ancestor in $M_k$ of a face adjacent to the handle in the fine mesh $M_n$. The constraints are coarsened as follows:

- Fixed coarse vertices are fixed if and only if their fine counterparts are fixed.
- For every fine-mesh handle $p_i$ moved to a new position $q_i$, we choose the coarse faces $\{f_k\}$ which are the ancestors of the fine faces $\{f_n\}$ adjacent to $p_i$, and define positional constraints on their associated (coarse) projective maps.

Note that a projective map in $P^3$ has only one degree of freedom (of moving a single $R^3$ point, i.e. 3 degrees of freedom) left in the ambient space outside of the plane of the transformed quad. Thus, every coarse quad can only satisfy one extra positional constraint exactly. In light of this, we devise the rule according to which the best coarse-deformation level is chosen: it is the coarsest level for which no face is constrained with more than one handle, or a handle and a fixed point (there is no problem with multiple fixed points which leave a face unchanged).

### 6.3. Prolonging Maps

Given the final set of maps $\{A_k\}$ for the level $M_k$, We prolong (i.e., propagate coarse-to-fine) them in a trivial manner: the prescribed map $T_g$ for a given face $g$ in $M_k$ is exactly $A_k$, if $f$ is the ancestor of $g$ in the subdivision. Therefore, we aspire to move the entire progeny of a single coarse quad as a
solid. In the general case, the maps \( \{ T_g \} \) are not already in the projective shape space of \( M_n \), and so we solve a global step (Equation 4) as before, with the user-prescribed positional constraints in the fine level.

### Smoothness regularization

Projecting prescribed maps into the shape space does not guarantee that the result would be a smooth deformation, especially as the energy relates to the transformation of individual faces, and does not contain bending terms. To accommodate for bending smoothness, we add an energy term that reduces the variation in the map across faces: for all pairs of adjacent faces \( f, g \) we add the energy term:

\[
E_{\text{smooth}} = \alpha \sum_{f, g} \| A_f - A_g \|_F^2 \quad (\text{Frobenius norm})
\]

Such a subtraction of matrices is meaningful since the maps are always normalized. We used \( \alpha = 3 - 4 \) in all our experiments.

### 6.4. Full Multi-resolution Algorithm

Our full algorithm is produced in the following simple steps:

1. The user selects a ROI and editing handles from any vertex in the finest mesh \( M_n \).
2. The coarsest possible level \( k \) is chosen automatically as explained in Section 6.2
3. The mesh is deformed according to the basic algorithm, and the deformation is prolonged as in Section 6.3

We show our results in Figures 6, 7 and 8, and the results of the improvement are summarized in Table 1

### 6.5. Approximative Subdivisions

Approximative subdivision differ only in the sense that the fixing of a vertex \( p_i \) in the coarse mesh does not correspond to a prescribed position in the finest mesh (and vice versa).

Table 1: Statistics of multires deformations vs. full deformations. The first two rows are the sizes of the mesh in quads. The “Fine Deform” field denotes a full deformation without multi-resolution and with the same constraints. Times are in milisecs, and the improvement (last row) is the ratio of multires deform to full fine deformation (equivalent to [Vax12]). Lower value is better. It is naturally evident that multi-resolution deformation is superior when the fine mesh is bigger.

However, that does not change our algorithm. The only relevant change is that when we coarsen the fixed (unmoving) points, we constrain them to their actual positions in the coarse mesh, which do not correspond with their position in the fine mesh as in interpolatory subdivisions. That is however in accordance with the fact that the fixed faces stay stationary with the identity as projective maps. The moving handles are assumed to reside in the ambient space of the coarse mesh, rather than correspond to actual coarse vertices, in any case.

### 7. Editing by Direct Map Prescription

We offer a different approach to shape exploration with map prescriptions \( T_f \). Rather than exploring the eigenspaces of constrained Hessians of fairness energies, as done in [YYPM11, Vax12], we take an approach that is more in-

---

© 2014 The Author(s)
© 2014 The Eurographics Association and Blackwell Publishing Ltd.
spired by frame-field editing, as done in [LSLCO05]. By fixing some positions and prescribing simple classes of maps with the global step of our editing methodology, we can easily obtain a rich set of new shapes. We prescribe three types of transformations: rigid (rotations), perspective warps (transforming boxes into Frustums and rectangles into trapezoids), and uniform scaling. Uniform scaling is an interesting unique subspace: By prescribing uniform scale matrices $T_f = diag[t, t, t, 1]$, for instance, we can “inflate” and “deflate” the mesh. By setting $t = 0$ and constraining several fixed positions, we obtain a surface for which every face tries to shrink as much as possible, and therefore can be thought of being “as-minimal-as-possible”.

Since scaling matrices are affine, we always choose the linear subspace for which $s_f = 1$ in order to project prescribed scaling maps (and all affine maps in general) in the global step. Therefore, the matrices $T_f(t)$ form a line in the ambient projective map space, which remains a line after projecting to our compatible shape space hyperplane. Thus, we only need to solve for two matrix sets $\{A_f(t_1)\}, \{A_f(t_2)\}$, projected from the respective $T_f(t)$, and linear interpolation between them produces another compatible set. A scaling examples is depicted in Figure 11, and other examples of map prescription exploration are in Figures 9 and 10.
7.1. Interpolation and Extrapolation

Any (normalized) projective transformation matrix $A$ can be uniquely factorized into an affine map and a "pure" projective matrix in the following manner:

$$T = T_{proj}T_{aff} = \begin{pmatrix} I_{3 \times 3} & J \\ 0 & 1 \end{pmatrix} \begin{pmatrix} H_{3 \times 3} & 0 \\ Y & 1 \end{pmatrix} \quad (5)$$

The affine-part matrix $H$ can be further factorized as $H = QR$, where $R$ is an orthogonal matrix, and $Q$ is positive-definite. In order to interpolate between shapes related by affine maps in an as-rigid-as-possible manner, [Vax12] prescribed matrices $H(t) = (tQ + (1 - t)J)R$ for interpolated meshes $t \in [0, 1]$ (and extrapolated meshes for all $t \in \mathbb{R}$). We enhance this approach to projective maps, which makes interpolation possible between any two PQ meshes with the same topology, by considering the interpolated projective maps corresponding to $Y(t) = (1 - t)Y, J(t) = (1 - t)J$ as well, in order to linearly modify the translation and the perspective deformation. These matrices are used for the global step map prescription, as before. An example is in Figure 12.

8. Conjugate-preserving Projective Transformations

We next show that the discrete framework we defined has a meaningful continuous counterpart. PQ meshes are often considered as discretizations of *conjugate parametrizations* [BS08a]. Every subset of a quad mesh that does not include a singularity can be parametrized on a lattice s.t. distinct $u$- and $v$-polylines can be identified, corresponding with $(u, v)$ parameter lines on continuous surfaces. We identify two types of discrete differential forms [DKT05] on such a quad mesh:

**Vertex-based scalar functions $G(p)$, or 0-forms.** The discrete differentials of 0-forms are primal 1-forms defined on the oriented edges of the mesh as $d\alpha|e = G(p_2) - G(p_1), e = (p_1, p_2)$. The discrete derivative of the function $G_{uv}$ on the edges (according to the polyline and edge orientation) is proportional to the differential, when equipped with a metric (e.g., the edge length).

We next look at the coordinate function (i.e., the vertex-based function $G : V \rightarrow R^3$). Since we assumed that edges construe the $u$- and $v$- polylines, we get that $p_u$ (resp. $p_v$) on a $u$- (resp. $v$-) edge is proportional to the edge vector in $R^3$, and that $p_{uv}$ is proportional to the diagonal (see Figure 13).

**Face-based scalar functions $H(f)$, or 2-forms.** We can dually define $u$- (resp. $v$-) quad strips, and define the codifferential on face-based values (2-forms) as values (dual 1-forms) on dual edges $\hat{e} = (f, g)$ between two adjacent quads $f, g$. In a similar argument as above, the codifferential $d_{\hat{e}}\beta = H(g) - H(f)$ is proportional to the derivatives of quad-based pointwise (non-integrated) functions. Again, we omit the inherent Hodge star definition that is part of the codifferential, as the metric is irrelevant in our formulation (we only deal with linear dependence). We treat our per-face projective maps as the 2-forms in this section.

It is easy to see verify that a quad $p_{1,4}$ in homogeneous coordinates is planar if and only if $\det[p_1, p_2, p_3, p_4] = 0$. This is in fact the definition of planarity, as the supporting-plane equation is the null space to this matrix. By row additions, we also arrive at $\det[p_1, p_2 - p_1, p_3 - p_1, p_4 - p_1] \propto \det[p_{uv}, p_u, p_v, p_{uv}] = 0$, corresponding with the continuous projective definition of a conjugate parametrization of a conjugate parametrization $P(u, v)$:

$$P_{uv} = \alpha P_u + \beta P_v + \gamma P.$$  \hspace{1cm} (6)
Alluding to the fact that planar quads are considered as discretizations of conjugate parametrizations [BS08a].

If all points are transformed by a single projective transformation $A$, it is obvious that this determinant is multiplied by $\det(A)$, and thus a uniform projective transformation of the entire mesh clearly preserves conjugate parametrizations, as is well-known. This is a classical invariant in the field of projective differential geometry [Eis09]. Our framework defines compatible local projective transformations based on the compatibility conditions in Equation 2. Suppose, for the rest of the section (as in Figure 13), that the two quads $f, g$ are related by an edge $e = (\vec{p}_1, \vec{p}_2)$, belonging without loss of generality to a $v$-polylines, and then the dual edge $\hat{e} = (f, g)$ is therefore in a quad $u$-strip. We next show that by a process of local rescaling, available in both the discrete and the continuous settings, we can show an immediate connection between the compatibility conditions of our frameworks and conjugate-preserving transformations.

**Local rescaling:** Two adjacent maps $A_f, A_g$ are compatible on an edge $e$ if there are scaling factors on both sides of the edge so that the common points transform into projectively-equivalent points. However, we want to reduce the problem locally to simpler conditions, in order to make our point, and get rid of the scaling factors. Given the target Euclidean positions $q$ of a single face $f$, when trying to determine the transformation $A_f$, as in the local step (Equations 3), we get 16 equations of projective equivalence, with 20 variables (16 variables of map $A_f$, and 4 scaling variables $s_f, p$). That means that the null space of projectively-equivalent transformations has at least 4 degrees of freedom. Therefore, it is always possible to locally choose $B_f, B_g$ that are equivalent to $A_f, A_g$ (respectively) in their action on the quads, differing in their actions on the ambient space and the global scale factor. We can thus always choose $B_f, B_g$ to induce unit scale factors on vertices $p_1, p_2$ of the mutual edge $e$, and the local compatibility conditions can be simplified into:

$$\vec{p}_1(B_g - B_f) = 0, \quad \vec{p}_2(B_g - B_f) = 0 \Rightarrow \quad (\vec{p}_1 - \vec{p}_2)(B_g - B_f) = 0. \quad (7)$$

Note that there are not enough degrees of freedom for this $A_f, A_g \rightarrow B_f, B_g$ reparametrization to be globally possible, since it would then imply that our entire shape space is simply linear (all maps become affine). Put in a discrete-differential form we write that:

$$\langle d_0(p), d_2(B) \rangle \propto \langle p, B_u \rangle = 0, \quad \langle p_B, B_u \rangle = 0, \quad \langle p, B_u \rangle = 0, \quad \langle p, B_u \rangle = 0. \quad (8)$$

We remind again that the definition of a discrete hodge star (usually necessary for the definition of an inner product) is irrelevant here since we only deal with proportionate terms. The inner product is taken column-wise in the matrix, as evident in equation 7.

We next look at the continuous counterpart to our compatibility formulation. The face-based projective maps translate into a matrix field on a conjugate-parametrized surface $P(u, v), A(u, v) : R^2 \rightarrow R^4$. We want to show that we have a similar condition on the orthogonality of the gradients of the matrix field and the gradients of the parametrization. For each $(u, v)$ we look at a small neighborhood $X$. The next step is to find the continuous counterpart of $B$, which is a transformation equivalent to $A$, but with a locally unifying scale factor. This is in fact simple: suppose that $w(u, v)$ is the homogeneous weight induced by $A$ (the fourth coordinate). We simply take $B(u, v) = \frac{A(u, v)}{w(u, v)}$. Then, the following lemma holds:

**Lemma 8.1** If $B$ holds that $\forall (u, v) \in X, P \cdot B_u = P \cdot B_v = 0$, then the parametrization of surface $Q(u, v) = P(u, v)A(u, v) = P(u, v)B(u, v)$ is also conjugate.

We simply generalized the discrete derivatives to their continuous counterparts as defined in this section.

**Proof** We derive $Q(u, v)$, use the given identities provided by the input in the Lemma, formed as the continuous conditions derived in Equation 9, and check for conjugacy:

$$Q_u = (PB)_u = PB_u + P\cdot B = P\cdot B, \quad Q_v = (PB)_v = PB_v + P\cdot B = P\cdot B, \quad Q_{uw} = (P_B)_u = P_uB_u + P_uB_v = P_uB = P_uB \Rightarrow \quad Q_{uw} = (P_v)_u = P_uB_u + P_uB_v = P_uB = P_uB \Rightarrow \quad \det(P, P_u, P_v, P_{uv}) = \det([Q, Q_u, Q_v, Q_{uv}], B) = 0 \quad \square$$

We have therefore defined a subclass of continuous projective transformation-matrix fields that are conjugate-preserving, and for which our framework is a meaningful discretization. For transformation fields which are purely affine, it is straightforward to check that we are also forming the continuous version of the framework in [Vax12].

9. Discussion

Our projective framework opens up the way for new and generalized applications in polyhedral modeling. The simple linearization of the local and the global steps allows for intuitive and easy editing of meshes with mere solving of convex functionals in homogeneous coordinates. However, working with a projective space comes at the price of limiting our ability to effectively deal with applications that require the explicit transformation of vectors (e.g., subtraction of points), or the definitions of metric-based functionals, such as the area gradient. That limits our capacity to define discrete differential quantities on our shape space,
e.g., Laplace operators. Furthermore, we must deal with object translation, because while translation is irrelevant to the shape of the object, it is not projectively-invariant, and therefore our matrices must incorporate this component, which might hamper the numerical results. An interesting direction for future research is the combination of homogeneous and Euclidean coordinates, perhaps by alternating iterations of algorithms. It is also important to note that, while remaining linear, projective transformations add 15 variables per face, and the amount of variables becomes prohibitive for big meshes.

Another shortcoming is the adherence to a fixed topology, which is an ubiquitous trait of shape-space-based editing frameworks. Working with changing topologies is a challenge for future work. We are further interested in adapting other geometric modeling methods to polyhedral meshes, such as simplification, remeshing, and direct reconstruction from point clouds.

Implementation Details: We implemented linear solving and matrix factorization in MATLAB for all of our examples. Measured times relate to an i7 CPU machine with 16Gb memory.

10. Acknowledgements
This work has been supported by Austrian Science Fund (FWF) under grant P23735-N13 and under Lise-Meitner grant M1618-N25. We would like to thank Helmut Pottmann and Johannes Wallner for their major support and the anonymous reviewers for their feedback.

References

© 2014 The Author(s)
© 2014 The Eurographics Association and Blackwell Publishing Ltd.
Appendix A: Best Approximating Rigid and similarity maps

We wish to determine the least-squares closest map $T_f$ for a given map $B_f$, minimizing $\|B_f - T_f\|^2$. For that matter, we compute the positive singular value decomposition of $B_f$, i.e. $B_f = U\Sigma V^T$. We can then obtain:

- **Closest Rigid Map:** A.k.a. the Orthogonal Procrustes Problem, the solution is obtained at $T_f = UV^T$.
- **Closest Similar Map:** The solution is obtained at $T_f = USV^T$, $S = sI$. The scalar value $s$ is the average of the singular values of the 2D transformation of the planar face restricted to the plane. A simple way to approximate it, is to average the change in edge lengths following the transformation.