DOI: 10.1111/cgf.13903

COMPUTER GRAPHICS *forum*
Volume 39 (2020), number 6 pp. 49–65

Hyperspectral Inverse Skinning

Songrun Liu,[1] Jianchao Tan,[1] Zhigang Deng[2] and Yotam Gingold[1]

[1]Department of Computer Science, George Mason University, Fairfax, VA, USA
{songruner, tanjianchaoustc}@gmail.com, ygingold@gmu.edu
[2]Department of Computer Science, University of Houston, Houston, TX, USA
zdeng4@uh.edu

Abstract

In example-based inverse linear blend skinning (LBS), a collection of poses (e.g. animation frames) are given, and the goal is finding skinning weights and transformation matrices that closely reproduce the input. These poses may come from physical simulation, direct mesh editing, motion capture or another deformation rig. We provide a re-formulation of inverse skinning as a problem in high-dimensional Euclidean space. The transformation matrices applied to a vertex across all poses can be thought of as a point in high dimensions. We cast the inverse LBS problem as one of finding a tight-fitting simplex around these points (a well-studied problem in hyperspectral imaging). Although we do not observe transformation matrices directly, the 3D position of a vertex across all of its poses defines an affine subspace, or flat. We solve a 'closest flat' optimization problem to find points on these flats, and then compute a minimum-volume enclosing simplex whose vertices are the transformation matrices and whose barycentric coordinates are the skinning weights. We are able to create LBS rigs with state-of-the-art reconstruction error and state-of-the-art compression ratios for mesh animation sequences. Our solution does not consider weight sparsity or the rigidity of recovered transformations. We include observations and insights into the closest flat problem. Its ideal solution and optimal LBS reconstruction error remain an open problem.

Keywords: linear blend skinning, deformation, animation, affine geometry, hyperspectral unmixing

ACM CCS: • Computing methodologies → Mesh geometry models, Motion processing

1. Introduction

Linear blend skinning (LBS) is one of the most widely used techniques for animation and deformation due to its simplicity and efficiency. The deformed position of a vertex is determined by the weighted average of a set of transformations T_j:

$$\mathbf{v}'_i = \sum_{j=1}^{h} w_{i,j} T_j \mathbf{v}_i, \tag{1}$$

where $\{w_{i,j}\}$ are the skinning weights, $\{T_j\}$ is the set of h affine transformation matrices and $\mathbf{v}_i$ is the undeformed position of the i-th vertex (with the homogeneous coordinate = 1). The skinning weights for each vertex are typically non-negative ($w_{i,j} \geq 0$) and sum-to-one ($\sum_{j=1}^{h} w_{i,j} = 1$). During animation, only the transformation matrices change. This reduces the amount of information needed for creating and editing. Conceptually, the transformations serve as the *handles* and may represent the 'bones' of a character. The weights and transformations of a model are often called its rig. See [JDKL14] for a recent survey.

Given the utility and ubiquity of LBS, it is desirable to obtain an LBS rig given only a set of observed deformations. The observations may come from physical simulation, direct mesh editing, motion capture or another deformation rig. In this example-based inverse skinning problem, the desired output is a set of transformation matrices, skinning weights and possibly also the rest pose mesh itself. In the most challenging version of this problem, the observations come in the form of uncorrelated point clouds [CZ11, LCY*18]. Often a set of meshes with corresponded vertices are given [JT05, KSO10, LD12], and the problem is to find a corresponding LBS rig. In other words, given a set of deformed poses (vertices $\mathbf{v}'_{p,i}$), find the rig ($w_{i,j}$,$T_{p,j}$) and possibly also the undeformed positions $\mathbf{v}_i$. Inverse LBS can be formulated as a constrained least squares problem [JDKL14]:

$$\min_{w,R,\mathbf{t},\mathbf{v}} \sum_{p=1}^{\#poses} \sum_{i=1}^{n} \left\| \mathbf{v}'_{p,i} - \sum_{j=1}^{h} w_{i,j} T_{p,j} \mathbf{v}_i \right\|^2 \tag{2}$$

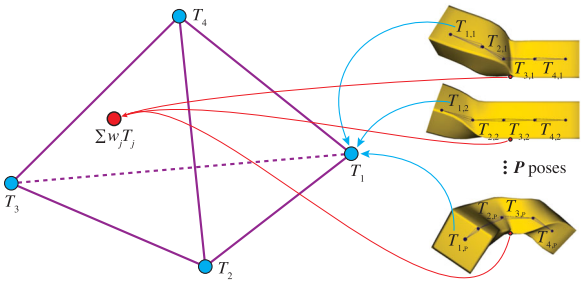

Figure 1: *Transformation matrices as points in $\mathbb{R}^{3\cdot4\cdot\#poses}$. Given several poses (e.g. animation frames) of a linear blend skinned model (right), each bone's transformation matrices across all poses are vertices of a simplex in $\mathbb{R}^{3\cdot4\cdot\#poses}$ (left). Any vertex's transformation (red circle) is the weighted sum of the bone transformations. The weights w_i of the LBS rig are the barycentric coordinates with respect to the simplex.*

subject to:

$$w_{i,j} \geq 0 \quad \text{and} \quad \sum_{j=1}^{h} w_{i,j} = 1. \tag{3}$$

The constraints in Equation (3) are convex combination constraints on the skinning weights that ensure the blended transformations interpolate the handle transformations. Two additional optional constraints are per-vertex weight sparsity ($\|w\|_0 \leq K$) and rigid handle transformations ($T_j \in SE(3)$). In this work, our *assumptions* are that an undeformed mesh is given with vertex correspondences between the undeformed mesh and deformed poses. We do not consider weight sparsity or transformation rigidity.

Our key *contribution* is a re-formulation of inverse skinning as a problem in high-dimensional space. The transformation matrices applied to a mesh vertex $\mathbf{v}_i$ across all poses can be thought of as a point in $\mathbb{R}^{3\cdot4\cdot\#\text{poses}}$. All such points that can be generated by a given LBS rig define a simplex whose vertices are the handle transformations across all poses (Figure 1). The simplex lives in an $(h-1)$-dimensional affine subspace of $\mathbb{R}^{3\cdot4\cdot\#\text{poses}}$, where h is the number of handles. As per the preferred terminology in mathematics, we call an affine subspace a *flat*. (See Appendix A for definitions and useful identities.) Therefore, our challenge is to fit a simplex to the observed data. This is related to a well-studied problem in the field of hyperspectral imaging, where the vertices of the smallest-volume simplex enclosing all observed points are the 'pure', unmixed data [Cra94]. Although, in our setting, we do not observe any transformation matrices directly, each vertex $\mathbf{v}_i$, based on its deformed position in other poses $\mathbf{v}'_{p,i}$, defines a $9 \cdot$ #poses-dimensional flat in $\mathbb{R}^{3\cdot4\cdot\#\text{poses}}$. Our algorithm can be summarized as follows:

1. We first estimate a point in $\mathbb{R}^{3\cdot4\cdot\#\text{poses}}$ for each vertex. We do this by intersecting vertices' flats (Section 4.1).
2. For a rig with h handles, all points must lie on an $(h-1)$ dimensional flat in $\mathbb{R}^{3\cdot4\cdot\#\text{poses}}$. We search for the $(h-1)$-dimensional flat whose distance to the vertices' flats is minimized (Section 4.2).
3. Finally, we compute the closest point on the $(h-1)$-dimensional flat to each vertex's flat. The smallest enclosing simplex for these points provides us with the transformations (its vertices) and skinning weights (trivially computed as barycentric coordinates) (Section 5).

One advantage of our approach is that the simplex encloses the observations and so has the lowest possible error of any set of handles of given size (i.e. our approach can generally achieve smaller reconstruction errors than the state of the art). By contrast, the clustering used in previous inverse LBS algorithms may choose handles inside the simplex which, due to the convexity constraints on the weights, cannot contribute to all vertices. We re-frame inverse LBS as two simple sub-problems: minimizing flat/flat distances and finding the smallest-volume simplex. This has additional benefits: (i) The reconstruction error is completely determined by the answer to the first problem, enabling an efficient bisection search for the optimal number of handles. (ii) The second sub-problem is actively studied in the field of hyperspectral unmixing; improved algorithms can, therefore, be immediately applied to improve inverse LBS results. (iii) Our per-vertex transformation matrix initial guess (Section 4.1) could be used by other inverse LBS algorithms.

We evaluate our algorithm on 16 distinct models from the literature and newly created. Some models are based on performance-capture data and some have known ground-truth rigs. We provide numeric comparisons in all cases and video comparisons when available. We also evaluate choices made in our algorithm's design, such as the impact of our initial guess.

Our approach to minimize flat/flat distances is not guaranteed to find the global optimum. However, in posing inverse skinning as a flat/flat distance minimization problem, we provide a fresh direction for future progress on this problem. In Section 4 and Appendix C, we describe numerous (inferior) alternative approaches to minimize flat/flat distances and experiments with scenarios in which the global optimum can be reliably found. We consider this to be an important contribution of our work.

2. Related Work

2.1. Skinning decomposition

An early form of inverse LBS was first studied by Wang and Phillips [WP02] as a way to overcome LBS artefacts like joint collapse and the so-called candy-wrapper artefacts [GB08]. To overcome these limitations, with the assumption that each pose is associated with a bone skeleton, they solve for skinning parameters in a more general model than LBS in which vertices have independent weights for each entry in the transformation matrices. James and Twigg [JT05] were the first to study the inverse LBS problem with a single weight per transformation matrix. They proposed to extract rotations from mesh triangles in correspondence via polar decomposition, and then apply mean-shift clustering on the rotations to obtain an initial estimate of bone transformations. They then progressively correct skinning weights and bone transformations to better match the input. Various follow-up works also estimate full bone skeleton hierarchies from example poses [SY07, DATTS08, HTRS10]. All of these approaches are based on an analysis of 3D motion, often including a clustering step. In contrast, our approach is based on the convex geometric structure of transformation matrices as high-dimensional points. We do not consider skeleton hierarchies or rigid

transformations. Our handles are defined as the vertices of the simplest possible convex hull—a simplex. Handles found by clustering make a sub-optimal use of the affine subspace, since they cannot contribute to points outside the convex hull of the handles. As a result, our approach is able to achieve lower errors with the same number of bones.

Kavan *et al.* [KSO10] proposed to view the inverse skinning problem as a special case of matrix factorization. They achieve fast performance in part by first reducing the dimensionality of the mesh positions, followed by iterative quadratic optimization for all vertices' transformations, skinning weights and rest pose positions. In particular, they solve small isolated problems or a system dependent on the number of poses, not the number of vertices, and take the advantage of their weight sparsity assumption. Le and Deng [LD12, LD13, LD14] presented a suite of works that address skinning decomposition, weight compression and automatic rigging with a bone skeleton, respectively. Our problem statement is similar to Le and Deng [LD12], which computes handle transformations and skinning weights given a set of posed meshes. They optionally enforce rigid handle transformations and sparse weights, which our approach does not. In the absence of sparsity considerations, our approach is faster and achieves lower errors. Recently, Mukai and Kuriyama [MK16] proposed a practical method to synthesize plausible and dynamic skin deformation based on an auxiliary helper bone rig. Holden *et al.* [HSK15] introduced a method to solve for high-level rig parameter given bone skeleton transformations. Thiery and Eisemann [TE18] introduced a method to replicate nonlinear as-rigid-as-possible deformations, and optional given poses, given a mesh and skeleton. They support skeleton joint limits and provide interactive tools for editing weights. Loper *et al.* [LMR*15] described SMPL, a skinning and blend shape-based model for human body shapes and poses. They obtain model parameters via optimization (training), to minimize the error from observed data. They have additional degrees of freedom (blend shape parameters) but initialize and regularize their optimization with a known human rig. Our approach could be useful within their optimization. We do not consider these additional kinds of rigs or bones. Jacobson *et al.*'s [JBK*12] FAST method takes as input a rigged model (including skin weights) and transformations for a subset of the handles. Our work is orthogonal, as it finds the rig and transformations. For a recent survey of inverse LBS approaches, see [JDKL14] part IV.

In sharp contrast with all previous approaches, we solve the inverse LBS problem from a fresh point of view: finding a simplified convex hull of transformation matrices as points in a high-dimensional geometric space. We accomplish this via flat estimation followed by hyperspectral unmixing. We provide new insights into the flat estimation problem and are quickly able to generate solutions within a given reconstruction error.

2.2. Surface registration

Both our approach and the aforementioned inverse LBS approaches assume vertex correspondences across all mesh poses. However, several related works do not require this assumption. Chang and Zwicker [CZ08, CZ09, CZ11] presented a series of works on surface registration for articulated shapes; their solution takes the form of LBS with binary weights, which is akin to segmenting the surface at joints and restricting transformations to rigid motions.

Surface registration is also a fundamental problem in computer graphics. A large body of literature has been published on surface registration via iterative closest point (ICP). Most closely related to our needs, Amberg *et al.* [ARV07] extend the ICP framework to non-rigid registration using different regularizations with an adjustable stiffness parameter. Like us, they also find per-vertex affine transformation matrices. In contrast with our approach, they solve a large, sparse system of equations with a smoothness term between neighbouring vertices. Schneider and Eisert [SE09] describe a new cost function for ICP by minimizing a first-order approximation of its cost function. Yoshiyasu *et al.* [YMYK14] proposed to find geometric as well as semantic consistencies with user-specified features while preserving mesh topology. Among various surface registration applications, deformable 3D shape registration with a known mesh topology draws our special attention. For example, Cao *et al.* [CTA*14] proposed to first find the reliable correspondences using a spin-image method, and then minimize an energy consisting of rotation, regularization and correspondence terms. Papazov and Burschka [PB11] compute an as-rigid-as-possible deformation based on local similarity transforms. We aim to address a much simpler problem, since we are given the correspondence, with the restriction that our solution lies in a low-dimensional LBS subspace.

2.3. Sub-space clustering

Several existing works considered the problem of clustering onto flats [ZSWL12, HYL*03] or clustering flats via closest points [GLS10, LS13]. None of these works consider the scenario where both the input data and output clusters are flats. In contrast, in our scenario, the input data are flats (defined by each vertex's undeformed and deformed positions) and the output is a single flat, since in LBS all transformations must lie on the same flat. Our flat optimization (Section 4) must solve the flat/flat distance problem.

2.4. Hyperspectral unmixing

Blind hyperspectral unmixing, also known as unsupervised hyperspectral unmixing, is an active area of research in signal processing. The goal is to recover the spectral signatures of the constitutive materials present in a hyperspectral image along with their abundances in each pixel [MBDC*14]. In the terminology of computer graphics, the input is an image with more than three channels per pixel, and each pixel is assumed to be the linear mixture of an unknown palette. The goal is to recover the palette, which they call *endmembers*, and mixing weights, which they call *abundances*. In our problem, we want to decompose per-vertex transformation matrices into a set of handle transformations (pure materials) and skinning weights (abundances). Craig [Cra94] conjectured that the no-pure-pixel hyperspectral unmixing problem could be solved by finding the minimum-volume simplex enclosing the set of observed points. Since then, increasingly efficient and more robust-to-outlier algorithms have been proposed [CCHM09, BD09, ACMC10, ALBDP14, LCWC16]. In practice, these algorithms work extremely well, despite it being well known that the minimum-volume enclosing simplex problem, in general, has local

minima and even infinite families of minimal solutions [HGPP13]. In the last few years, two independent sufficiency proofs have explained why the optimal simplex can be exactly recovered under surprisingly mild conditions [LML*15, FMHS15].

Hyperspectral unmixing is equivalent to non-negative matrix factorization (NMF), whose solution in general is NP-complete [Vav09]. Fortunately, if the NMF is separable, then the time complexity becomes polynomial [AGKM12]. An NMF problem $M = UV$ with $(U, V) \geq 0$ is called *separable* if there exists a factorization where each column of U equals a column of M. Separability corresponds to the so-called *pure-pixel* assumption in hyperspectral unmixing, in which every material is observed not mixed with any others in at least one pixel. We do not make this assumption, and we also do not want our handle transformations to be non-negative.

3. Background

The 3×4 affine transformation matrices applied to a rest pose vertex $\mathbf{v}_i$ in every pose can be (row-major) vectorized and stacked vertically, one pose above the other, to form a very tall column matrix in $\mathbb{R}^{12 \cdot \#\text{poses}}$. The set of all transformation matrices obtainable by an LBS rig, $\sum_{j=1}^{h} w_j T_j$ such that $\sum w_j = 1$, forms an $(h-1)$-dimensional affine subspace, or flat, in $\mathbb{R}^{12 \cdot \#\text{poses}}$. With the constraint that weights are positive ($w_j \geq 0$), the set defines a simplex in the flat. See Figure 1 for an illustration. In our inverse skinning problem, this *handle transformation flat* is unknown.

Flats generalize the concept of a line or plane (a linear subspace offset from the origin) to higher dimensions. In 2D (respectively, 3D), two lines (respectively, planes) almost always intersect, because they are both hyperplanes, or flats whose dimension is exactly one less than the ambient space. The general scenario, in which the flat dimension is at least two less than the ambient dimension, is akin to lines in 3D, which 'rarely' intersect. The distance between two flats is the distance between their closest points. See Appendix A for a detailed description of flats. Briefly, a flat can be defined explicitly as $\mathcal{L} = \{\mathbf{p} + B\mathbf{z}\}$, where the columns of the matrix B span directions parallel to the flat, $\mathbf{z}$ is the vector of parameters and $\mathbf{p}$ is again a point on the flat. A flat can also be written as $\mathcal{L} = \{F\mathbf{w}\}$, where the columns of the matrix F are points in the flat and the parameters $\mathbf{w}$ must sum to 1. Finally, every matrix equation defines a flat implicitly: $\mathcal{L} = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a}\}$.

In our inverse skinning setting, vectorizing the expression that the transformation matrices applied to an undeformed vertex $\mathbf{v}_i$ equals its position in all poses $\{\mathbf{v}'_{p,i}\}$ implicitly defines a flat:

$$\bar{V}_i \mathbf{x} = \begin{bmatrix} \mathbf{v}'_{1,i} \\ \vdots \\ \mathbf{v}'_{p,i} \end{bmatrix} = \mathbf{v}'_i, \tag{4}$$

where $\mathbf{x}$ is a point in the space of transformation matrices ($\mathbb{R}^{12 \cdot \#\text{poses}}$) and $\bar{V}_i = I_{3 \cdot \#\text{poses}} \otimes \mathbf{v}_i^\top$. The $\mathbf{v}_i$ and $\mathbf{v}'_{p,i}$ are column matrices, and $\otimes$ is the Kronecker product. $\mathbf{v}_i$ is in homogeneous coordinates, while $\mathbf{v}'_{p,i}$ is not. $\bar{V}_i$ is a $(3 \cdot \#\text{poses}) \times (12 \cdot \#\text{poses})$ block diagonal matrix with orthogonal rows. The implied flat $\bar{V}_i \mathbf{x} = \mathbf{v}'_i$ is $(9 \cdot \#\text{poses})$-dimensional. The point $\mathbf{x}$ is the (row-major) vectorization of the 3×4 affine transformation matrices stacked vertically one pose above the other to form a column matrix. These per-vertex flats are known based on the input to the inverse skinning problem. The points in these flats are the matrices which transform an undeformed vertex to its corresponding deformed positions. The flats are high dimensional, because the transformations are not unique (e.g. pure translations).

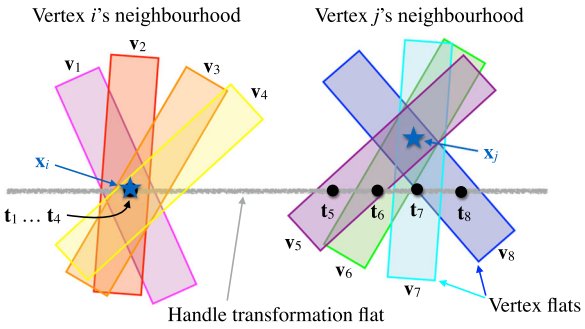

Figure 2: *We seek the $(h-1)$-dimensional handle transformation flat whose distance to all vertex flats is minimal. Because $h-1$ is typically smaller than the vertex flat dimension ($9 \cdot \#\text{poses}$), we visualize the handle transformation flat as a line. The closest point in transformation-space to a set of vertex neighbourhood flats is close to the desired handle transformation flat. If the vertex neighbourhood flats all intersect at the same point, then the vertex neighbourhood has constant skinning weights. See the text for details.*

Our first goal (Section 4) is to find a handle flat that minimizes the distance to all vertex flats (Figure 2). In general, this distance will be non-zero, because the handle flat has a much lower dimension than the ambient space or the vertex flats. Our second goal (Section 5) is to find an appropriate simplex in the handle flat containing all vertices' transformations (Figure 1). This is a well-studied problem in hyperspectral imaging. The simplex provides the solution to our inverse skinning problem. It determines the skinning weights and handle transformations for each pose.

4. Per-Vertex Transformations

The goal of this section is to find a flat $\mathcal{L}$ passing through or minimizing the distance to the vertices' flats, $\bar{V}_i \mathbf{x} = \mathbf{v}'_i$ (Figure 2). If $\mathcal{L}$ is $(h-1)$-dimensional, then any h affinely independent points in $\mathcal{L}$ can be affinely combined to reproduce all points in $\mathcal{L}$. (It is easy to choose points for which the weights are convex combinations.) Therefore, the LBS reconstruction error is determined by the flat, not by the choice of points. Rather than searching for the handles directly in $\mathbb{R}^{12 \cdot \#\text{poses}}$, we search instead for a handle transformation flat $\mathcal{L}$. The goal of Section 5 will be to find points on the flat to serve as the handle transformations.

The dimension of $\mathcal{L}$ is $h-1$, where h is the number of handles. If $h = 1$, then $\mathcal{L}$ is a single point, and the solution can be found in closed-form by inverting a small 4×4 square matrix. However, when $h > 1$, we know of no guaranteed solution to the problem of finding the flat $\mathcal{L}$ that minimizes the distance to a given set of flats

$\bar{V}_i\mathbf{x} = \mathbf{v}'_i$:

$$\underset{\mathcal{L}}{\operatorname{argmin}} \sum_i D(\mathcal{L}, \bar{V}_i\mathbf{x} = \mathbf{v}'_i)^2, \tag{5}$$

where D measures the Euclidean distance between flats. We experimented with many ways to express and minimize the flat/flat distance: direct gradient and Hessian-based optimization for an explicit representation of $\mathcal{L}$, including optimization on the Graff manifold; gradient-based optimization based on projection matrices; global optimization via basin hopping; computing the Karcher mean; and several alternating optimization strategies, one of which exhibited the best performance of all approaches. We describe this superior approach in detail in Section 4.2. We describe alternative approaches in Appendix C.

A straightforward expression for Equation (5) is

$$\min_{\mathbf{p},\mathbf{B}} \sum_i \|\bar{V}_i(\mathbf{p} + B\mathbf{z}_i) - \mathbf{v}'_i\|^2, \tag{6}$$

where $\mathbf{p} \in \mathbb{R}^{12\cdot\#\text{poses}}$ and $B \in \mathbb{R}^{(12\cdot\#\text{poses})\times(h-1)}$ are the explicit representation for $\mathcal{L}$, and $\mathbf{z_i} \in \mathbb{R}^{h-1}$ are the parameters for the closest point on $\mathcal{L}$: $\mathbf{z}_i = -(B^\top\bar{V}_i^\top\bar{V}_iB)^{-1}B^\top\bar{V}_i^\top(\bar{V}_i\mathbf{p} - \mathbf{v}'_i)$. The pseudoinverse should be used when $3 \cdot \#\text{poses} < h - 1$.

The $\bar{V}_i$ are defined in Equation (4) not to be orthonormal. As a result, Equation (6) measures the distance from $\mathcal{L}$ to vertex i's flat in terms of the 3D position error, rather than the distance in $\mathbb{R}^{12\cdot\#\text{poses}}$.

Equation (5) is, in general, non-convex; to see this, consider a cube in $\mathbb{R}^3$. Let the given flats be the lines through all 12 edges of the cube. The line with the minimum distance to all given lines passes through opposite corners of the cube; there are four equivalent solutions separated by inferior solutions.

Figure 3 shows the results of an experiment in which we generate sets of 100 random d-dimensional flats in $\mathbb{R}^{24}$ that intersect a k-dimensional flat, and then optimize for the k-dimensional flat parameters from a random initial guess. We use a straightforward explicit expression for the energy (Equation 6) optimized with a Hessian-based trust region solver constraining B to lie on the Grassmann manifold [TKW16]. The solver terminated after a maximum of 200 iterations. (200 iterations took 20 min on average. We achieved qualitatively similar results in $\mathbb{R}^{12}$ at up to 1000 iterations.) We report the energy of the local minimum found. All experimental setups have a known zero-energy solution. When $d = 0$, the given flats are points and the problem degenerates to a simple linear least-squares problem; a perfect solution is always found. More generally, when $d + k < 24$, a zero-energy solution can often be recovered, though with decreasing success and rapidly increasing computation time as $d + k$ approaches 24. When $d + k \geq 24$, the problem is trivial. The set of unknown flats that do *not* intersect all given flats has measure-zero, and so a random initial guess almost surely has zero energy. In our inverse LBS scenario, the ambient space is $\mathbb{R}^{12\cdot\#\text{poses}}$, the given flats have dimension $\mathbb{R}^{9\cdot\#\text{poses}}$ and the number of handles is a small number independent of #poses. This combination seems to place our problem into the zone where solutions are challenging to find naively even when a known zero-energy solution exists. See Figure 4 and our Supplemental Materials for a visualization of the $k = 3$ scenario. See our Supplemental Materials for an animation of the optimization iterations for the $d = k = 1$ scenario in 3D.

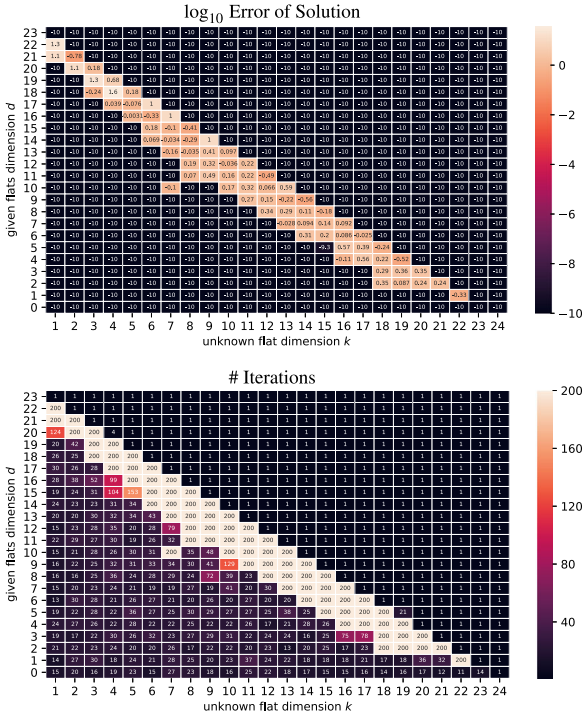

Figure 3: *Solutions obtained from numerical optimization with varying given and unknown flat dimensions in $\mathbb{R}^{24}$. A known zero-error solution exists for all scenarios. The errors are shown at $\log_{10}$ scale; error is capped from below to 10^{-10}. The number of solver iterations was capped at 200. (We achieved qualitatively similar results in $\mathbb{R}^{12}$ at up to 1000 iterations.) See the Supplemental Materials for animations of optimization iterations.*

4.1. Initial guess

In the absence of a closed form solution to this problem, we conjecture that nearby vertices are likely transformed by similar transformation matrices. (In LBS, this will be exactly true when vertices share the same skinning weights for all bones.) For each vertex and its neighbourhood, we can compute the point in $\mathbb{R}^{12\cdot\#\text{poses}}$ whose distance to the vertices' flats is minimized in a least squares sense:

$$\mathbf{x}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{j\in\{i\}\cup\mathcal{N}(i)} \left\| \frac{1}{\|\mathbf{v}_j\|^2}\bar{V}_j^\top\bar{V}_j(\mathbf{x} - \mathbf{t}_j) \right\|^2, \tag{7}$$

where the divisor normalizes the rows of $\bar{V}_j$, $\mathcal{N}(i)$ are the one-ring neighbours of vertex i, and $\mathbf{t}_j$ is any valid transformation matrix in vertex j's flat as a point in $\mathbb{R}^{12\cdot\#\text{poses}}$. Neighbours are needed because a single vertex does not uniquely determine a transformation. A trivial valid transformation matrix can be obtained as the pure translation matrix mapping $\mathbf{v}_j$ to $\mathbf{v}'_{p,j}$. $\bar{V}^\top\bar{V}$ is a block diagonal matrix whose diagonal blocks are each simply $\mathbf{v}^\top\mathbf{v}$. When scaled by the divisor, it is a projection matrix. Minimizing the above entails solving a 4×4 system of equations.

Since the metric we ultimately care about in inverse LBS is the 3D error, we minimize a version of Equation (7) that applies the transformation to $\mathbf{v}_j$ (left-multiplying by $\bar{V}_j$), which leads to a simpler

Table 1: *Keeping different fractions of our per-vertex initial guess results in different final error after 10 iterations of our bi-quadratic flat optimization (Section 4.2).*

	Transformation errors/vertex errors E_{RMS}									
Model	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
cylinder	0.23	0.27	0.22	0.01	8e-3	0.02	0.13	0.2	0.16	0.16
	21.59	15.89	17.77	1.0	0.88	2.34	17.37	13.22	12.88	12.67
cube	0.07	0.11	0.10	0.09	0.10	0.11	0.11	0.11	0.11	0.13
	5.55	6.87	6.25	6.37	6.12	6.28	6.49	6.65	8.55	7.58
cheburashka	0.04	0.02	0.03	0.02	0.02	0.02	0.02	0.02	0.03	0.06
	1.24	0.90	0.82	0.85	0.92	0.75	0.95	0.81	0.94	1.16
wolf	38.74	108.14	21.99	0.61	0.2	0.14	0.15	0.14	0.12	0.19
	0.04	0.09	2e-3	4e-8	2e-8	1.5e-9	2e-9	1.2e-9	1.3e-9	8.9e-10
cow	1.81	0.46	0.39	0.36	0.41	0.66	0.96	0.65	0.68	0.72
	0.49	0.18	0.21	0.14	0.20	0.31	0.23	0.25	0.25	0.29

The initial guess was computed via the unconstrained one-ring neighbourhood. The lowest error is found at approximately 50%.

expression:

$$\mathbf{x}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{j \in \{i\} \cup \mathcal{N}(i)} \|\bar{V}_j \mathbf{x} - \mathbf{v}'_j\|^2 \tag{8}$$

because $\frac{1}{\|\mathbf{v}_j\|^2} \bar{V}_j \bar{V}_j^\top = I$. Due to the block diagonal structure of $\bar{V}_i$, minimizing the expressions amounts to solving 4×4 systems of equations.

A vertex's flat has a $(9 \cdot \#\text{poses})$-dimensional nullspace and $(3 \cdot \#\text{poses})$-dimensional row-space, so any four (or more) non-planar vertices lead to a full-rank system. Intuitively, this is because the action of a 3D affine transformation matrix cannot be determined from its action on a plane; the scale factor in the direction orthogonal to the plane remains unconstrained. When the sum of squared distances (Equation 7 or 8) is exactly 0, we know that all flats intersect at a single point, the minimizer $\mathbf{x}_i$. This will be the case for vertices whose neighbourhoods always undergo the same affine transformations. This occurs in LBS whenever weights are locally constant. This occurs at any rigid part of a shape. When the sum is non-zero, the intersection point is still likely to be near the optimal handle transformation flat (Figure 2).

Given the set of minimizers $\mathbf{x}_i$, one for each vertex with an associated full-rank system (Equation 8), we can compute an initial guess for the desired flat $\mathcal{L}_{guess}$ via principal component analysis (PCA). PCA directly provides an explicit representation for a flat: the mean is a point on $\mathcal{L}_{guess}$, and the first h principal components form an orthonormal basis for directions parallel to it. Note that this PCA projection error is only approximate. It does not exactly capture the desired flat/flat distance in $\mathbb{R}^{12 \cdot \#\text{poses}}$ or in 3D (Equation 6). The approximation is due to the fact that the distance from a vertex flat to $\mathcal{L}_{guess}$ may be smaller than the distance from a particular point on that flat. The distance could also be larger, if the minimizers $\mathbf{x}_i$ of Equations (7) and (8) are not restricted to lie on the vertex i's flat. Since vertices whose reconstruction error is large should be considered outliers, we experimented with the fraction of vertices to keep before computing PCA (Table 1). The lowest error was obtained near the 50th percentile, which we use for our results. See Appendix B for additional experiments comparing variants of Equations (7) and (8).

In the following section, we improve the PCA approximation $\mathcal{L}_{guess}$ via numerical optimization of Equation (6).

4.2. Flat optimization

We seek to minimize the sum of squared flat/flat distances (Equation 6) given an initial guess $\mathcal{L}_{guess}$. To do so, we use the explicit affine expression for the flat:

$$\min_F \sum_i \|\bar{V}_i F \mathbf{w}_i - \mathbf{v}'_i\|^2 \tag{9}$$

subject to $\mathbf{1}^\top \mathbf{w}_i = 1$. This expression is quadratic in each of F, $\mathbf{w}_i$ and even the rest pose positions $\bar{V}_i$. The quadratic expressions for each $\mathbf{w}_i$ are independent of each other and entail solving an $(h+1) \times (h+1)$ system of equations. (The extra row and column are for enforcing the sum-to-one constraint with a Lagrange multiplier.) The quadratic expressions for each $\mathbf{v}_i$ are independent of each other and entail solving a 3×3 system of equations. The minimizer for F results in a linear matrix equation:

$$\sum_i \lambda_i \big(I_{3 \cdot \#\text{poses}} \otimes (\mathbf{v}_i \mathbf{v}_i^\top)\big) F \big(\mathbf{w}_i \mathbf{w}_i^\top\big) = -\lambda_i \sum_i \bar{V}_i \mathbf{v}_i'^\top \mathbf{w}_i, \tag{10}$$

where λ_i are per-vertex weights which we set to the smallest singular value of the system used to solve for $\mathbf{w}_i$. Via the row-major vectorization identity $\text{vec}_{\text{row}}(ABC) = (A \otimes C^\top)\text{vec}_{\text{row}}(B)$, and the observation that the identity matrix in Equation (10) leads to a repeated-block diagonal system matrix, this results in a $4h \times 4h$ system of equations. (Only a single block needs to be solved with multiple right-hand sides.) We set the initial F to $\mathcal{L}_{guess}$ and then alternately solve for $\mathbf{w}_i$ and F (and optionally $\bar{V}_i$) for a few iterations (i.e. no more than a specified maximum number of iterations, e.g. 2, or until the Frobenius norm between successive iterations of F falls below a threshold, e.g. 0.2).[1]

[1] To prevent tangential drift, we normalize W's columns to be unit length from their average. To measure convergence with principal angles, F could

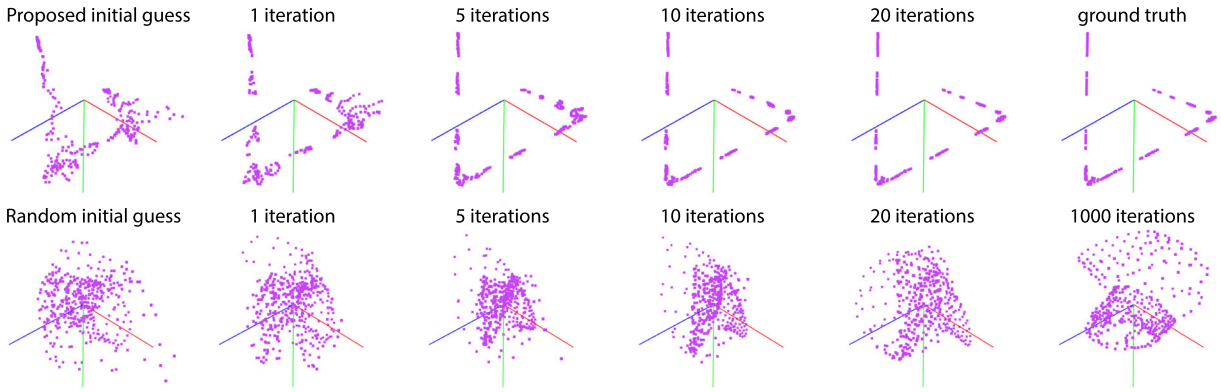

Figure 4: *Visualization of bi-quadratic flat optimization on the cylinder example, with and without our proposed initial guess. In this example, there are four bones, each with the vertex of a tetrahedron, so the desired handle flat is 3D. The points visualized here are the closest points on the handle flat to each vertex's flat (computed via projection). Because the orientation of the visualized 3D space is arbitrary (obtained via PCA from the ambient $\mathbb{R}^{48}$), we apply a procrustes transformation between adjacent iterations. Top row: From our initial guess (Section 4.1), our bi-quadratic optimization (Section 4.2) rapidly recovers a result close to the ground truth in only a few iterations. Bottom row: From a random initial guess, our bi-quadratic optimization reaches a state dissimilar to the ground truth from which little progress is made. See the Supplemental Materials for videos of the optimization progress.*

Figure 5 compares our optimization approach with a variety of alternatives. These approaches are detailed in Appendix C. Our alternating bi-quadratic approach immediately achieves low error and makes little further progress. Figure 4 and the Supplemental Materials visualize the iterative optimization of a four-bone model (cylinder), with and without our proposed initial guess. The handle flat for this model is a 3D affine sub-space. We visualize the closest points on this flat to each vertex's flat.

The result of this optimization is a flat $\mathcal{L}$. This process is quick, and exactly captures the LBS error for a rig with h handles. When the desired number of handles is unknown, we run a binary search on h to find the smallest number of handles whose LBS reconstruction error is below a desired threshold.

The closest point on $\mathcal{L}$ to each vertex flat is given by $F\mathbf{w}_i$. These points are the input to the next stage of our algorithm, which computes the final LBS rig (skinning weights and handle transformations).

5. Handle Transformation and Skinning Weights Estimation

Given a set of points in $\mathbb{R}^{12\cdot\#\text{poses}}$ all lying on a flat with dimension $h-1$, the goal of this section is to compute a tight-fitting $(h-1)$-simplex around them. The h vertices of the simplex will serve as the handle transformations. The points' well-defined barycentric coordinates will serve as their skinning weights. This is the minimum volume simplex problem studied in the hyperspectral imaging community. We claim no particular contribution over the state-of-the-art [CCHM09, BD09, ACMC10, ALBDP14, LCWC16], though we are the first to recognize its relevance for inverse skinning. We provide a description for completeness.

be taken as a point on the Graff manifold and identified with a point on the higher-dimensional Grassmann manifold (Appendix A). We found the alternative of comparing both the principal angles of the parallel directions of F along with the Euclidean distance between flats to be numerically unstable.

Any simplex enclosing the set of points provides us with a set of handle transformations and skinning weights that satisfy the convexity constraints. All such simplices will have the same error in reproducing the poses; the error is entirely determined by the flat chosen in the previous section. Due to the linearity of LBS, this holds true even for arbitrary blends of the handle transformation matrices (e.g. during animation). Linearity tells us that we would see the same result as if we were to blend the observed posed mesh vertices directly.

Finding a minimum volume enclosing simplex is useful insofar as it leads to sparser skinning weights. The barycentric coordinates of a point are sparser, the closer a point is to a face of a simplex. (Lower dimensional faces are sparser than higher dimensional ones; e.g. points that lie on 3-simplex faces of a simplex have only four non-zeros.) The minimum-volume enclosing simplex ensures that there are points at least on all $(h-2)$-faces.

Therefore, although our approach does not make sparsity guarantees, this step increases sparsity and often leads to a large number of almost-sparse weights (<0.1). One could apply the weight-reducing post-process of [LS10]. (Intuitively, naive k-largest weight projection is akin to projecting points to the nearest $(k-1)$-face of the simplex. The discontinuities result from nearby points in $\mathbb{R}^{12\cdot\#\text{poses}}$ being on opposite sides of an angle bisector.)

Minimum volume simplex techniques work well in practice, even when all observed data are quite blended (the 'no pure-pixel' assumption), despite the seeming ill-posedness of the problem [HGPP13]. In our experiments, for LBS-created animations, given ground truth blended pose transformation matrices for mesh vertices, the minimum volume enclosing simplex exactly recovers the original LBS rig (handle transformations and skin weights) (Table 5).

The surprising success of these algorithms was recently explained by two (independent) proofs of sufficiency criteria [LML*15, FMHS15]. The criteria are fairly mild. They are summarized in

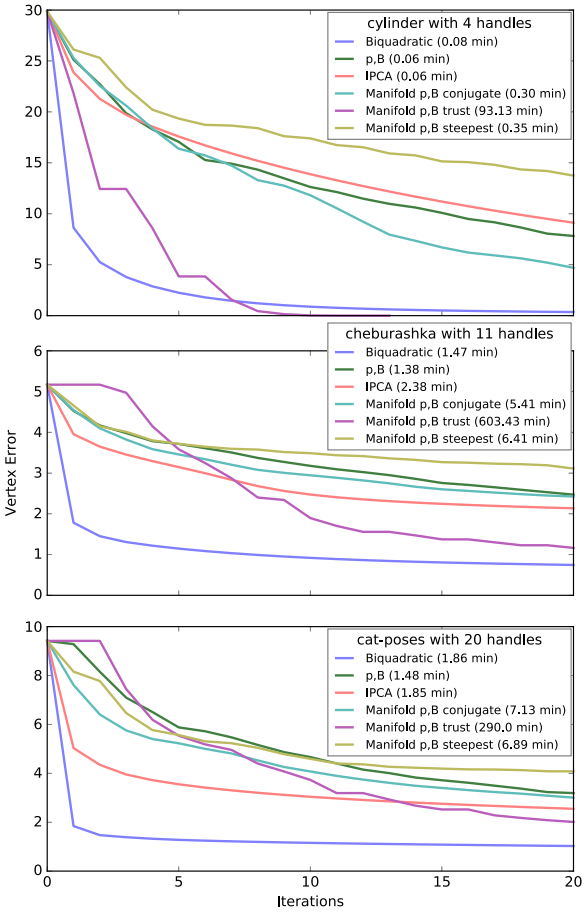

Figure 5: *Our alternating bi-quadratic optimization rapidly achieves very low error compared to other experimental optimization strategies. In one case (cylinder), Hessian-based optimization restricted to lie on the Grassmann manifold achieves ground truth and superlinear convergence after 93 min of computation. See Section 4.2 and Appendix C for details of these approaches.*

Figure 6. Interestingly, the criteria become easier to satisfy in higher dimensions as $\frac{1}{\sqrt{h-1}}$ decreases.

5.1. Finding a minimum volume enclosing simplex

Our first step is to apply PCA and project the $(12 \cdot \#\text{poses})$-dimensional points that lie on an $(h-1)$-dimensional flat down to $h-1$ dimensions.

Let C be a matrix whose h columns are the $(h-1)$-dimensional vertices of a simplex in homogeneous coordinates (a 1 appended). Then, C is a square $h \times h$ matrix whose bottom row is all 1's. The simplex volume is equivalent to the absolute value of the determinant $|C|$. Let D be an $h \times \#\text{points}$ matrix, where each column is one of the $(h-1)$-dimensional observed points in homogeneous coordinates. Our straightforward objective function can be expressed as:

$$\min_C |\det(C)| \tag{11}$$

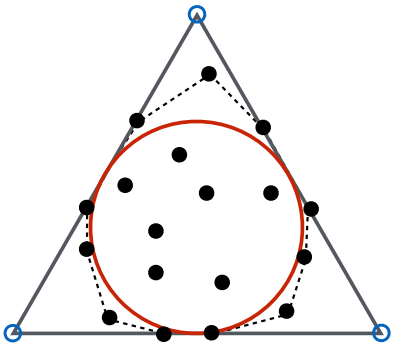

Figure 6: *A simplex is always regular in weight-space. If the convex hull (dashed line) of observed points (black dots) encloses a ball of radius $\frac{1}{\sqrt{h-1}}$, and the 'purest' observed point prevents any rotated regular simplex form also enclosing the points, then the minimum volume enclosing simplex is unique and exactly recovers the ground truth simplex vertices (blue dots).*

subject to:

$$C^{-1}D \geq 0 \tag{12}$$

$$C_{h,i} = 1, \quad \forall i \in [1,h]. \tag{13}$$

Equation (13) imposes the homogeneity of C, which in turn imposes the affinity of barycentric coordinates. In the above formulations, the constraint Equation (12) which computes the barycentric coordinates is non-linear, which makes Equation (11) difficult to solve. Since $\det(C^{-1}) = \frac{1}{\det(C)}$, $\min(|\det(C)|) \Leftrightarrow \max(\det(C^{-1}))$. Let $X = C^{-1}$. Because the volume of a simplex increases exponentially with its dimension, the gradients of this objective function quickly become vanishingly small. Instead, one can optimize the logarithm of the determinant of X as a better-behaved objective function. This monotonic transformation does not affect the solution. Equation (11) can be reformulated as:

$$\min(-\log\det(X)) \tag{14}$$

subject to:

$$XD \geq 0 \tag{15}$$

$$X\mathbf{1}_h = [0,0,0,\ldots,1]^T, \tag{16}$$

where $\mathbf{1}$ is a column vector of 1's. Equation (16) imposes the homogeneity of X^{-1}. Equation (14) is a non-linear minimization problem with linear constraints. Constrained gradient-based optimization, e.g. SLSQP [Kra88], works successfully albeit too slowly for our large problems.

Following the work of Agathos *et al.* [ALBDP14], we optimize the objective function using sequential quadratic programming (SQP) with a local majorizer for the quadratic and linear terms:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top G\mathbf{x} + \mathbf{c}^\top\mathbf{x} + \text{constant}, \tag{17}$$

Table 2: *A comparison of reconstructed vertex error and computation time between [KSO10] and our flat optimization (Section 4).*

				Approx. error E_{RMS}		Execution time (min)	
Dataset	# vertices	# poses	# bones	Kavan *et al.*	Ours	Kavan *et al.*	Ours
crane	10 002	175	40	1.4	0.73	0.36	2.66
elasticCow	2904	204	18	3.6	3.23	0.08	1.16
elephant	42 321	48	25	1.4	0.46	0.37	3.49
horse	8431	48	30	1.3	0.35	0.07	0.67
samba	9971	175	30	1.5	0.86	0.26	2.1

Table 3: *A comparison of reconstructed vertex error and computation time between SSDR [LD12] and our flat optimization (Section 4).*

				Approx. error E_{RMS}		Execution time (min)	
Dataset	# vertices	# poses	# bones	SSDR	Ours	SSDR	Ours
cat-poses	7027	9	10	6	5.51	0.5	0.34
			15	4.2	3.02	0.8	0.50
			20	3.3	1.32	1.3	0.57
			25	2.5	0.47	1.9	0.61
chickenCrossing	3030	400	20	7.0	3.79	11.8	6.47
			28	4.2	1.46	24.6	7.48
elephant-gallop	42 321	48	10	3.9	5.95	12.6	1.52
			20	2.2	1.65	31.7	2.20
			27	1.6	0.69	50.5	3.48
elephant-poses	42 321	10	10	6.6	15.982	5	1.47
			21	2.8	2.52	17	1.92
face-poses	29 299	9	27	3.2	0.59	13.9	1.61
horse-collapse	8431	53	10	4.6	4.19	2.1	0.33
			20	3.1	0.92	4.7	0.52
horse-gallop	8431	48	10	4.8	3.89	1.9	0.33
			20	2.3	0.91	6	0.49
			33	1.6	0.26	10.4	0.75
horse-poses	8431	10	10	5.5	4.89	0.6	0.24
			20	2.6	1.11	1.6	0.32
lion-poses	5000	9	10	5.8	5.79	0.3	0.12
			21	3.0	1.03	0.9	0.17
pcow	2904	204	10	7.3	5.98	2.5	0.97
			24	3.3	2.25	7.4	1.25
pdance	7061	201	10	4.2	3.40	6	2.17
			24	1.4	0.27	22.8	2.12
pjump	15 830	222	20	4.2	3.19	36.1	4.26
			40	2.6	1.43	94.2	5.09

where $\mathbf{c} = \mathbf{g} - H\mathbf{x}$ and $G = \mathrm{diag}(H)$. $\mathbf{g}$ is the gradient of $-\log\det(X)$ at our current (vectorized) guess for X, and H is the Hessian. The gradient, Hessian and inverse Hessian have simple closed-form expressions. $\mathbf{g} = \mathrm{vec}(-X^{-T})$,$H = X^{-T} \otimes X^{-1}$ and $H^{-1} = X^T \otimes X$.

We solved Equation (17) subject to the constraint Equations (15) and (16) with MOSEK [ApS18] via CVXOPT's qp interface [ADV18]. If the returned solution increased the simplex volume (the majorizer is only local), we iteratively bisect the line segment between the returned solution and our previous solution until the simplex volume decreases or convergence. Unlike [ALBDP14], we terminate after a maximum of 10 rather than four SQP iterations; further iterations make little progress.

Given a solution X, handle transformations in all poses are simply the columns of X^{-1} rotated and translated back to $\mathbb{R}^{12\cdot\#\mathrm{poses}}$. Skinning weights are trivially computed as XD.

5.1.1. *Initial guess*

We start optimization with a valid initial guess. We translate the data points so that all entries become positive (the minimum corner of the axis-aligned bounding box **b**). The translated points are now bounded on all-but-one side by the axis-aligned planes through the origin. We then find the offset of the plane with normal **1** so that all translated points lie on the origin-side of the plane. The smallest such offset d for this plane is the maximum L_1

Table 4: *A comparison of reconstructed vertex error and bits per vertex per frame (bpfv) time between [LDJ*19] and our flat optimization (Section 4). Better (lower) errors are indicated in bold.*

				Approx. error E_{RMS}		bpfv	
Dataset	# vertices	# poses	# bones	Luo *et al.*	Ours	Luo *et al.*	Ours
chickenCrossing	3030	400	20	12.34	**3.79**	4.02	4.13
			28	6.90	**1.46**	5.62	5.79
elephant-gallop	42 321	48	10	29.48	**5.95**	6.57	6.76
			20	7.86	**1.65**	13.50	13.51
			27	4.11	**0.69**	18.49	18.24
horse-collapse	8431	53	10	**1.83**	4.19	6.67	6.49
			20	**0.31**	0.92	12.55	12.99
horse-gallop	8431	48	10	33.28	**3.89**	7.01	7.12
			20	5.59	**0.91**	14.85	14.24
			33	1.73	**0.26**	23.82	23.50
pcow	2904	204	10	61.57	**5.98**	2.84	2.89
			24	11.28	**2.25**	6.92	6.94
pdance	7061	201	10	40.30	**3.40**	2.79	2.14
			24	8.11	**0.27**	5.17	5.13
pjump	15 830	222	20	11.28	**3.19**	3.33	3.37
			40	2.07	**1.43**	6.89	6.74
crane	10 002	175	40	3.37	**0.73**	8.76	8.85
elasticCow	2904	204	18	20.45	**3.23**	5.97	5.20
elephant	42 321	48	25	4.64	**0.46**	17.07	16.89
horse	8431	48	30	2.45	**0.35**	20.94	21.37
samba	9971	175	30	3.58	**0.86**	6.60	6.64

norm of any translated point. The vertices of a valid initial simplex are the intersection points of the plane with each coordinate axis: $(d, 0, 0, \ldots)$, $(0, d, 0, \ldots)$, ... along with the origin. (These vertices must be translated by $-\mathbf{b}$.)

6. Results and Evaluation

We evaluated our algorithms on a 2015 13" MacBook Pro with a 2.9 GHz Intel Core i5-5257U processor and 16 GB of RAM. We tested our methods on various models with different sets of poses given as 3D triangle meshes with vertex correspondence. The dataset includes new examples with known ground truth bone transformations and per-vertex weights computed via bounded biharmonic weights [JBPS11]. Tables B.2 and 5 and Figure 8 show reconstruction errors for per-vertex transformations and per-bone transformations, respectively. In this validation experiment, we run our simplex optimization until convergence.

We also evaluated our algorithm on examples from two state-of-the-art approaches, [KSO10] and [LD12]. These examples were not created by LBS and do not have zero-energy solutions. For all examples, we run our bi-quadratic flat optimization with our initial guess (unconstrained, one-ring neighbourhood keeping transformations for vertices with the 50th percentile lowest error). The running time for the initial guess is small, less than 10 s for most models and approximately 30 s for the elephant, our most complex example.

We compared our bi-quadratic flat optimization approach with two state-of-the-art inverse skinning models, [KSO10] and smooth skinning decomposition with rigid bones (SSDR) [LD12]. Le and Deng [LD12] discussed and showed the superiority of their SSDR method over previous approaches like skinning mesh animations [JT05] and learning skeletons for shape and pose [HTRS10]. For the consistency of comparison, we use the same error metric used by SSDR and proposed in [KSO10], which is $E_{RMS} = 1000\sqrt{\frac{\sum_i \|\hat{\mathbf{v}}_i - \mathbf{v}'_i\|^2}{3|V||P|}}$, where $\hat{\mathbf{v}}$ and $\mathbf{v}'$ are the recovered and ground truth vertex positions, $|V|$ is the number of vertices and $|P|$ is the number of poses. Our bi-quadratic flat optimization reduces error quickly in most cases (Figure 5). Therefore, for the sake of efficiency, we terminate optimization after four iterations. Table 2 shows that our bi-quadratic flat optimization achieves a smaller vertex error than Kavan *et al.*'s approach, though at the cost of slightly longer computation time. We speculate that Kavan *et al.*'s better performance is due to their assumption of four-sparsity for the weights. By fixing the sparsity pattern, they have 4 degrees of freedom per vertex rather than h. They also operate in reduced coordinates by eliminating vertices with linearly dependent trajectories. Table 3 shows that our approach not only generally outperforms SSDR in error measurement, but also has a big win over SSDR on performance. Table 3 contains the error and running time as reported in [LD12]; we also ran their code without the rigidity constraint with virtually no change to the reported error and running time. Figure 7 and the supplemental videos show that our output closely matches the ground truth, whereas [LD12] produces visual artefacts. (See the Supplemental Materials for full recovered sequences, including side-by-side comparisons with previous work when available.) Unlike these approaches, we do not enforce four-sparsity for the recovered weights. Popular automatic skin weight approaches do not generate sparse weights [JBPS11, BP07].

Inverse skinning can also be used as a form of mesh animation compression. Mesh animation compression algorithms take a

Table 5: *Information about per-bone transformation estimation (Section 5) on various models.*

Model	# vertices	# faces	# bones	# iterations to converge	Transformation error per bone	Weight error	Optimization time (min)
cylinder	420	800	4	4	6.1e-5	2.1e-5	<0.005
cube	1538	3072	4	3	2.9e-5	2.4e-4	<0.005
cheburashka	6669	13 334	11	9	2.9e-6	9.7e-7	0.96
wolf	5075	10 018	21	13	7.3e-5	4.8e-6	5.12
cow	11 666	23 328	15	10	8.8e-5	7.1e-7	16.19

Each model's examples are generated manually from skeletal deformation. Hyperspectral unmixing approach can recover ground truth within minutes.

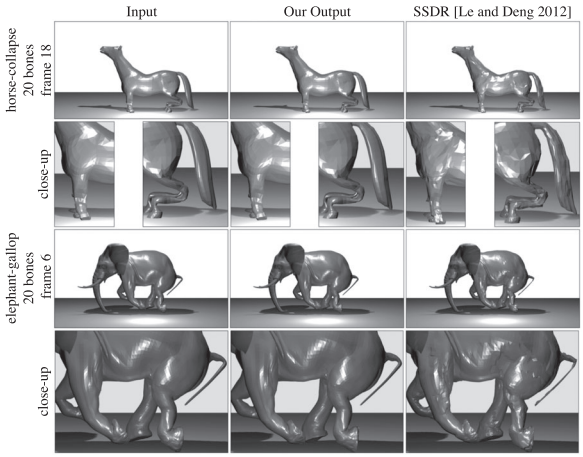

Figure 7: *Our output closely matches the input mesh sequences. The output from SSDR [LD12], even without rigidity constraints, displays clear visual artefacts. (We render with flat shading to accurately portray surface quality.) See the Supplemental Materials for entire mesh sequences.*

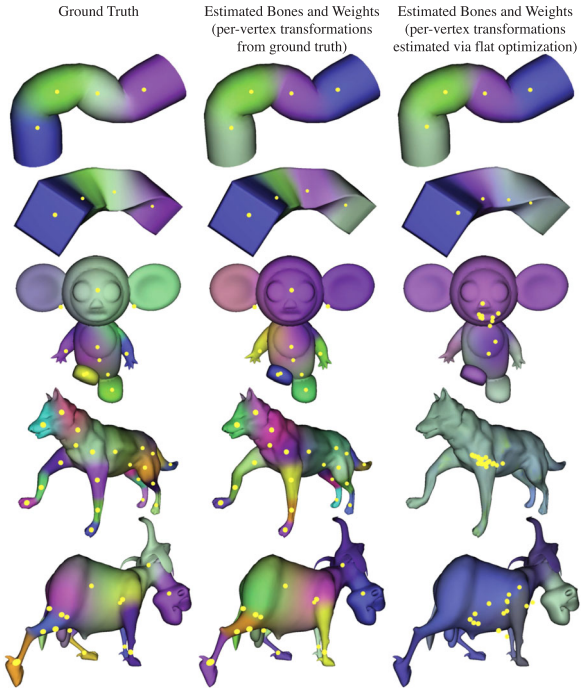

Figure 8: *Given ground truth per-vertex transformations (left), the recovered weights and bone transformations match almost exactly (middle). However, recovered per-vertex tranformations enclose too big a volume in transformation space, resulting in incorrect bone transformations and low sparsity (right). In all cases, vertex positions are recovered almost exactly. Each handle is assigned a colour and visualized per-vertex via colour mixing and with a yellow dot at its centre-of-mass.*

temporally coherent sequence of frames as input and produce an encoding that aims to minimize the reconstruction error for a target bits per vertex per frame (bpfv). The bpfv for an uncompressed 3D vertex is $3 \cdot 32$ bits using single-precision floating-point numbers. In inverse skinning, the handle weights are a fixed cost per vertex ($h \cdot 32$ bits using single-precision floats), regardless of the animation length. Each animation frame requires an affine transformation matrix per handle to be shared by all vertices ($\frac{12 \cdot h}{\#vertices} \cdot 32$ bits). The total bpfv for a given mesh animation compressed using inverse skinning is obtained by adding the per-frame bits to the fixed cost amortized across all frames. In Table 4, we compare our approach to a state-of-the-art mesh animation compression method [LDJ*19]. (We omit examples consisting of temporally incoherent poses, for which the technique of Luo *et al.* [LDJ*19] is not applicable.) In all but one animation, our approach obtains a much lower reconstruction error for the same total bpfv. Across all examples, our approach's reconstruction error is a factor of 4.6× lower (geometric average). In the one example for which our approach is inferior, a horse collapses as if made of cloth. We conjecture that any animations in which large portions of a shape move coherently (if not rigidly) will be better compressed using inverse skinning.

As shown in Table 5 and Figure 8, our hyperspectral unmixing approach recovers nearly exact ground-truth per-bone transformations and skinning weights when given perfect (ground-truth) per-vertex transformations. To measure transformation error, we use the mean absolute error across all $12 \cdot$ #poses entries in the transformation matrices. To further verify the recovery's correctness, we visualize each bone's position as the weighted centre of mass of all vertices and each bone's influence as a distinct colour. Figure 8 shows that our hyperspectral unmixing approach is sensitive to errors from per-vertex transformations. For this table and figure, we ran flat optimization and simplex fitting each for 10 iterations. We

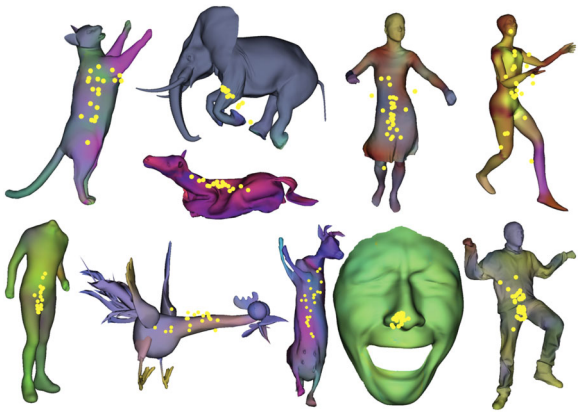

Figure 9: *The results of our hyperspectral inverse skinning pipeline on examples not created with linear-blend skinning. These examples do not have zero-energy solutions or ground-truth per-vertex transformations. Human motion sequences are performance captured. Our approach recovers the vertex positions with state-of-the-art error. However, recovered per-vertex tranformations often enclose large volumes in transformation space, resulting in low sparsity. Each handle is assigned a colour and visualized per-vertex via colour mixing and with a yellow dot at its centre-of-mass.*

experimented with random sampling to drop outliers, but found that it did not help. When comparing the visual distribution of per-vertex transformation matrices between our recovered results and the ground truth, there is sometimes a systematic error to our local minima (rotation in $\mathbb{R}^n$) rather than noise (Figure 4). Figure 9 visualizes our estimated bones' positions and weights on examples not created with LBS and so without a zero-energy solution or ground-truth per-vertex transformations. The human motion sequences were obtained via performance capture. Run-time performance ranges from a few seconds to minutes for larger models and is dominated by the number of handles rather than the number of vertices. In the future, we would like to experiment with other simplex fitting algorithms that claim superior performance [BD09, LCWC16].

7. Conclusion

We have shown that a re-formulation of inverse skinning as a problem in high-dimensional Euclidean space leads to a very fast and novel decomposition of the problem. Our first stage poses the problem of finding per-vertex transformations as flat/flat distance minimization. This problem has a simple expression. We experimented with many approaches to solve this problem, resulting in a fast and efficient solver. However, we consider this to still be an open problem. No approach is yet capable of recovering known ground truth solutions. Our second stage connects skinning decomposition to *hyperspectral image unmixing*, which is well studied in the field of remote sensing. These algorithms make mild assumptions and are capable of simultaneously recovering ground truth bone transformations and per-vertex skinning weights given correct per-vertex transformations. Similar ideas involving the convex hull were successfully used to find palette colours in RGB colour space [TLG16]. The idea can be naturally transplanted to decomposing transformation data, where the number of independent bones equals the dimensionality of all per-vertex transformations (e.g. reduced by PCA).

7.1. Limitation and future work

Our approach has several limitations. We do not consider weight sparsity. In our scenario, four-sparsity can be interpreted as finding a set of tetrahedra sharing vertices. It may be worth exploring an optimization for the vertices of a tetrahedral mesh or in which a set of intersecting 3D flats are optimized. Second, we do not constrain transformations to be rigid. This may or may not be appropriate depending on the application. This may aid in recovering ground truth per-vertex transformations, which our flat optimization is currently unable to do. One could constrain the vertices of the simplex to be rigid transformations, but this may require an adjustment to the handle flat itself. Alternatively, one could skip flat optimization and formulate a minimum-volume simplex optimization problem where the simplex vertices directly lie in $\mathbb{R}^{12\cdot\#\text{poses}}$. Finally, we do not consider artistic controls, such as allowing the user to specify or suggest a handle's position.

Acknowledgements

We are grateful to Harry Gingold, Alec Jacobson, Shahar Kovalsky, Arthur Dupre, Jie Gao and Leonard Schulman for informative discussions. We are grateful to Kyle Falicov for help with rendering. We thank Guoliang Luo for running his algorithm on our data. We thank the anonymous reviewers for their many helpful suggestions. Zhigang Deng was supported in part by the U.S. National Science Foundation (IIS-1524782). Songrun Liu, Jianchao Tan and Yotam Gingold were supported in part by the U.S. National Science Foundation (IIS-1453018), a Google research award and a gift from Adobe Systems Inc.

Appendix A: Flats

A *flat* is an affine subspace of $\mathbb{R}^n$. It generalizes the concept of a line or plane (a linear subspace offset from the origin) to higher dimensions. A flat $\mathcal{L}$ can be defined implicitly via $\mathcal{L} = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a}\}$, where the row-space of the matrix $A \in \mathbb{R}^{k\times n}$ is spanned by the directions orthogonal to the flat and a is the product of A and a point on the flat. The rows of A are often assumed to be orthonormal. We, too, assume this, and so our flat is $(n-k)$-dimensional. Note that when $k = 1$, this implicit form becomes the familiar expression for a (hyper)plane with normal $\mathbf{n}$: $A = \mathbf{n}^\top$ and so $\mathcal{L} = \{x \mid \mathbf{n}^\top x = \mathbf{n}^\top \mathbf{p}\}$, where $\mathbf{p} \in \mathbb{R}^n$ is a point on the flat (hyperplane). A hyperplane is a flat whose dimension is one less than the ambient space. It always has one direction of perpendicularity, which is easy to visualize in 3D. A line in 3D has a 2D space of normals (and two rows in its A): the plane perpendicular to it (Figure A.1).

A flat can also be defined explicitly as $\mathcal{L} = \{\mathbf{p} + B\mathbf{z}\}$, where the columns of the matrix B span directions parallel to the flat, $\mathbf{z}$ is the vector of parameters and $\mathbf{p}$ is again a point on the flat. The columns of B span the nullspace of A. If we assume that the columns of B are orthonormal, then $B \in \mathbb{R}^{n\times(n-k)}$ and $\mathbf{z} \in \mathbb{R}^{n-k}$. A flat can also be written as $F\mathbf{w}$, where the columns of the matrix $F \in \mathbb{R}^{n\times(1+n-k)}$

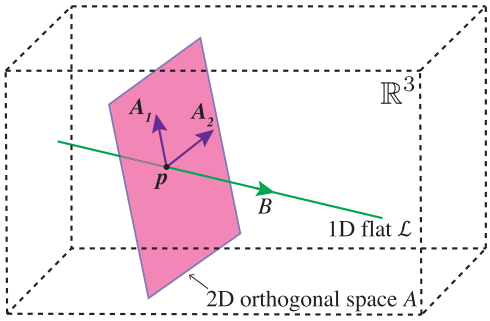

Figure A.1: *A one-dimensional flat $\mathcal{L} = \mathbf{p} + B\mathbf{z}$ and its orthogonal directions A in $\mathbb{R}^3$.*

are affinely independent points in the flat and the parameters $\mathbf{w} \in \mathbb{R}^{1+n-k}$ must sum to 1.

It is also useful to consider projections onto flats. Assuming A has orthonormal rows, $P_{row} = A^\top A$ is a projection matrix mapping a point in $\mathbb{R}^n$ to the closest point in A's row-space. (If the rows of A are linearly independent but not orthonormal, then the projection matrix is $A^\top(AA^\top)^{-1}A$.) The matrix $I - P_{row}$ is a projection matrix onto A's nullspace. Given B with orthonormal columns, $P_{null} = BB^\top$ and $I - P_{null}$ are projection matrices onto the nullspace and row-space, respectively. (If B's columns are linearly independent but not orthonormal, $P_{null} = B(B^\top B)^{-1}B^\top$.) Depending on n, k and the data at hand, desired projection matrices may be more readily computed in one or the other manner. Given a projection matrix P, $P^\top = P = PP$. The eigenvalues of P are either 0 or 1.

The reader is referred to DuPré and Kass [DK92] for an in-depth discussion on distance and degrees of parallelism between flats.

The space of all k-dimensional linear subspaces of $\mathbb{R}^n$ is known as the Grassmann manifold $\mathbf{Gr}(k, \mathbb{R}^n)$. The space of all k-dimensional affine subspaces of a vector space is the less well-known Graff manifold $\mathbf{Graff}(k, \mathbb{R}^n)$. The Graff manifold can be identified with $\mathbf{Gr}(k+1, \mathbb{R}^{n+1})$ by intersecting a linear subspace with the plane $x_{n+1} = 1$ (the hyperplane perpendicular to the last or $n+1$-th coordinate axis). This intersection results in $k+1$ points in $\mathbb{R}^n$ (the $n+1$-th coordinate for all points is 1). These $k+1$ points span the affine subspace. The canonical or principal angles between k and l-dimensional linear subspaces (defined by a $k \times n$ matrix B_1 and $l \times n$ matrix B_2, each with orthonormal columns) can be computed as the arccos of the singular values of $B_1^\top B_2$; when $k \neq l$, there are additional $|k-l|$ principal angles of $\frac{\pi}{2}$.

Appendix B: Per-Vertex Transformation Experiments

We experimented with Equations (7) and (8) (Section 4.1) with and without the constraint that $\mathbf{x}_i$ exactly reproduces the observed deformation (e.g. $\bar{V}_i\mathbf{x}_i = \mathbf{v}'_i$) and with three notions of vertex neighbourhoods $\mathcal{N}(i)$: the one-ring and randomly sampling vertices from within a Euclidean or geodesic distance of $\mathbf{v}_i$.

Table B.1 shows 3D positional and transformation errors resulting from Equation (8) on a set of models with known ground-truth transformations. In all tables, position errors in $\mathbb{R}^3$ are computed via E_{RMS} from [KSO10]. Transformation errors in $\mathbb{R}^{12\cdot\#\text{poses}}$ are element-wise average absolute deviation. For the unconstrained approaches, the one-ring neighbourhood generated the lowest 3D position error. The exact reproduction constraint increased the transformation error with negligible visual benefit to the position error. The Euclidean neighbourhood leads to slightly lower transformation error but substantially increased position errors. The geodesic neighbourhood produces worse results than the Euclidean. We experimented with various Euclidean and geodesic random sampling strategies. We found that the best result among 10 random subsets of 48 vertices taken from the 120 nearest neighbours generally produced the lowest error, so we used this random sampling strategy in our tables. The tables show the result of minimizing 3D error (8), which produced superior results to (7).

Table B.2 compares the downstream performance of these strategies followed by PCA as the initial guess for flat optimization (Section 4.2). In this experiment, we perform PCA on the 50% of per-vertex initial guesses with lowest position error. The unconstrained one-ring neighbourhood outperformed the other strategies. As a result of our experiments, and owing to its simplicity and run-time performance, we use the one-ring neighbourhood with unconstrained 3D error (8) for our results.

Appendix C: How Not to Minimize Flat/Flat Distances

We seek to minimize the sum of squared flat/flat distances (Equation 6) given an initial guess $\mathcal{L}_{guess}$. This minimization can

Table B1: *The error resulting from various initial guess schemes compared with ground truth.*

	Transformation errors/vertex errors E_{RMS}					
	Unconstrained approaches			Constrained approaches		
Model	One-ring	Euclidean	Geodesic	One-ring	Euclidean	Geodesic
cylinder	0.19/7.65	0.09/38.3	0.34/167.55	0.24/0.0	0.08/0.0	0.32/0.0
cube	0.12/4.69	0.04/12.96	0.18/11.09	0.12/0.0	0.04/0.0	0.18/0.0
cheburashka	0.04/0.22	0.03/4.0	0.05/6.20	0.04/0.0	0.03/0.0	0.05/0.0
wolf	0.04/0.40	0.03/3.43	0.18/14.47	0.04/0.0	0.03/0.0	0.19/0.0
cow	0.25/0.13	0.18/1.21	2.18/29.85	0.25/0.0	0.18/0.0	2.12/0.0

Position errors in $\mathbb{R}^3$ are computed via E_{RMS} from [KSO10]. Transformation errors in $\mathbb{R}^{12\cdot\#\text{poses}}$ are element-wise average absolute deviation.

Table B2: *The error resulting from various initial guess schemes followed by 10 iterations of our bi-quadratic flat optimization (Section 4.2) compared with ground truth.*

	Transformation errors/vertex errors E_{RMS}						
	Unconstrained approaches			Constrained approaches			
Model	One-ring	Euclidian	Geodesic	One-ring	Euclidian	Geodesic	without initial guess
cylinder	0.01/0.88	0.21/13.52	0.52/18.42	0.01/1.48	0.2/12.91	0.58/20.1	0.45/31.29
cube	0.10/6.12	0.11/6.59	0.28/10.76	0.11/5.97	0.09/7.96	0.34/9.52	0.2/15.51
cheburashka	0.02/0.92	0.04/0.83	1.02/1.59	0.02/0.83	0.03/0.91	0.99/2.15	0.1/1.22
wolf	0.2/1e-8	1.12/6.7e-8	2.19/1.3e-4	0.2/6.7e-9	2.11/1.1e-6	0.58/2.7e-5	0.32/5e-10
cow	0.42/0.2	5.53/0.27	10.92/0.98	0.27/0.22	1.46/0.31	18.57/1.76	1.63/0.74

In this experiment, we keep the 50% of per-vertex initial guesses with lowest position error.

be expressed in numerous ways. See Figure 5 for a comparison where relevant.

Direct optimization (p, B). We directly optimize Equation (6) using the BFGS algorithm [NW06]. This never achieves the low error of our proposed bi-quadratic approach. We also experimented with a combination of these two approaches, where we improve the bi-quadratic solution with direct optimization or switch approaches every 10 iterations. These combinations were inferior to simply running the bi-quadratic approach for additional iterations.

Optimization on an appropriate manifold (p, B manifold). We optimize Equation (6) with various algorithms (gradient descent, conjugate gradient and trust region) on the space of $\mathbb{R}^n \times \mathbf{Gr}(h-1, \mathbb{R}^n)$ [EAS98, TKW16]. The gradient descent and conjugate gradient algorithms are slower to compute and achieve higher error per iteration than our proposed bi-quadratic approach. The Hessian-based trust region algorithm is much slower to compute, taking hours to execute 20 iterations. However, on our simplest example, a cylinder with four bones, the trust region algorithm achieves superlinear convergence and the known ground truth solution (Figure 5).

Global optimization. We employed basin hopping [WD97], which is a stochastic global minimization algorithm in which random modifications of the current state are optimized via continuous optimization. We used our proposed approach (Section 4.2) for the continuous optimization. Basin hopping failed to improve upon the error of our proposed approach alone. The random modifications did not find basins with lower error. This approach is not plotted in Figure 5, because the curve would cover that of our proposed bi-quadratic approach.

Karcher mean. We experimented with computing the Riemannian centre of mass or Karcher mean of the given flats. The Karcher mean was proposed in the literature [CHV17, MRBD*14] as an effective technique for finding the centroid to a set of points on a Riemannian manifold. We experimented with representing flats as points on (1) the product manifold $\mathbb{R}^n \times \mathbf{Gr}(h-1, \mathbb{R}^n)$ or (2) the Graff manifold identified with points on the higher dimensional Grassmann manifold (Appendix A). In our setting, the unknown flat has different dimension than the given flats; in this case, the additional principal angles needed for the geodesic distance computation are taken as $\frac{\pi}{2}$. Unfortunately, this approach does not find a flat with small distance to other flats. We believe that this is due to the distortion of distances on the product or Graff manifolds.

Iterative PCA. We optimize Equation (6) with a different alternating decomposition than our proposed bi-quadratic approach. Instead, we alternate between (1) solving for the closest point on each vertex's flat to the handle flat $\mathcal{L}$ and then (2) solving for the flat that minimizes the squared distance to these closest points. Step (1) can be solved via

$$\underset{\mathbf{x}}{\operatorname{argmin}} \ (\mathbf{x}-\mathbf{p})^\top P_{null}(\mathbf{x}-\mathbf{p}) \tag{C.1}$$

subject to:

$$\bar{V}_i\mathbf{x} = \mathbf{v}_i', \tag{C.2}$$

where $P_{null} = I_{3\cdot\#\text{poses}} - B(B^\top B)^{-1}B^\top = I_{3\cdot\#\text{poses}} - BB^\dagger$ is the orthogonal projector onto the null-space of the handle flat (Appendix A) and $B^\dagger$ is the Moore–Penrose pseudo-inverse of B. This requires solving a different $(3\cdot\#\text{poses})\times(3\cdot\#\text{poses})$ system of equations for each vertex, with the constraint implemented either via Lagrange multipliers or as a least squares soft constraint. Step (2) can be solved by PCA, taking the first $h-1$ principal components as the parallel directions for the handle flat and the centre as the point through which the handle flat passes.

This iterative PCA (IPCA) approach produces better results than all other techniques except for our bi-quadratic approach (and the very expensive Hessian-based trust region approach). Our bi-quadratic approach alternates between (1) solving for the closest point on the handle flat $\mathcal{L}$ to each vertex's flat (in terms of handle flat parameters $\mathbf{w}_i$) and (2) solving for a new handle basis matrix F that minimizes the distance to the vertex flats using the $\mathbf{w}_i$ parameters. Our bi-quadratic approach is faster to compute, as it only requires the solution to a single, smaller $4h\times 4h$ system of equations.

Iterative Laplacian re-weighting. Any point on a d-dimensional flat can be represented as the weighted average of $d+1$ or more

affine independent points. In our setting, this implies that the following energy for per-vertex transformation matrices $\mathbf{t}_i \in \mathbb{R}^{12\cdot\#\text{poses}}$ should be zero:

$$E_{\text{local}} = \sum_i \left\| \mathbf{t}_i - \sum_{j\in\mathcal{N}(i)} w_{ij}\mathbf{t}_j \right\|^2, \tag{C.3}$$

where $\mathcal{N}(i)$ are the neighbours of vertex i and w_{ij} are scalar weights that sum to one. E_{local} can be expressed as $E_{\text{local}} = \sum_i \|L\bar{\mathbf{t}}\|^2$, where L is a $12\cdot\#\text{pose}\cdot\#\text{vertices}$ laplacian matrix and $\bar{\mathbf{t}}$ is a column vector containing all vertices' transformation matrices across all poses. We experimented with two definitions of vertex neighbours: the one-ring; and a fixed, random set of $2h$ vertices. To reproduce the observed poses, we wish to minimize:

$$E_{\text{data}} = \sum_i \|\bar{V}_i\mathbf{t}_i - \mathbf{v}'_i\|^2. \tag{C.4}$$

We optimize the sum of the two terms. The expression $E_{\text{local}} + E_{\text{data}}$ is quadratic in either w_{ij} or $\mathbf{t}_i$, so we alternate between solving for one while fixing the other. When solving for w_{ij}, E_{data} is constant and can be ignored, resulting in a small, typically underdetermined, local system per-vertex that can be solved in a least-square sense. Solving for $\mathbf{t}_i$, however, amounts to solving a very large, sparse system of equations. Finally, we take the first $h-1$ principal components of the final $\mathbf{t}_i$ to be the handle flat.

Because of the very large system of equations, this approach executes much more slowly than our proposed bi-quadratic approach and produces solutions with more error per iteration.

Orthogonal projector. The minimal distance between flats (Equation 5) can be written as $\|C(x_0 - y_0)\|$, where x_0 is any point on one flat and y_0 is any point on the other flat and C is the projection matrix onto the intersection of the two flats' orthogonal spaces [DK92]. For our problem, this results in the expression:

$$\sum_i \|C_i(\mathbf{p} - \mathbf{t}_i)\|^2 = \mathbf{p}^\top \left(\sum_i C_i\right)\mathbf{p} + \left(\sum_i \mathbf{t}_i^\top C_i\mathbf{t}_i\right) - 2\mathbf{p}^\top\left(\sum_i C_i\mathbf{t}_i\right), \tag{C.5}$$

where the $\mathbf{t}_i$ are any valid transformation matrix in vertex i's flat (Equation 7). The projection matrix C_i can be written (via the Anderson–Duffin formula) as $C_i = 2P_{\bar{V}_i}(P_{\bar{V}_i} + P_B)^\dagger P_B$, where P_B and $P_{\bar{V}_i}$ are orthogonal projectors onto the column-space of B and the row-space of $\bar{V}_i$, respectively. This approach is unstable and tends to increase error from a good initial guess.

Equation (C.5) is minimized (by setting the derivative with respect to $\mathbf{p}$ to 0) when $\mathbf{p} = (\sum_i C_i)^{-1}(\sum_i C_i\mathbf{t}_i)$. Substituting this expression for $\mathbf{p}$ results in:

$$\min_B \left(\sum_i \mathbf{t}_i^\top C_i\mathbf{t}_i\right) - \left(\sum_i C_i\mathbf{t}_i\right)^\top \left(\sum_i C_i\right)^{-1}\left(\sum_i C_i\mathbf{t}_i\right). \tag{C.6}$$

This expression is numerically unstable, because C_i is rank deficient. This rank deficiency corresponds to the fact that p can be any point on a flat. Even with a pseudoinverse, the expression is unstable.

References

[ACMC10] Ambikapathi A., Chan T.-H., Ma W.-K., Chi C.-Y.: A robust minimum volume enclosing simplex algorithm for hyperspectral unmixing. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)* (2010), IEEE, pp. 1202–1205.

[ADV18] Andersen M. S., Dahl J., Vandenberghe L.: CVXOPT: A python package for convex optimization (2018). http://cvxopt.org. Last accessed on 8 October 2018.

[AGKM12] Arora S., Ge R., Kannan R., Moitra A.: Computing a nonnegative matrix factorization–Provably. In *Proceedings of the 44 Annual ACM Symposium on Theory of Computing* (2012), ACM, pp. 145–162.

[ALBDP14] Agathos A., Li J., Bioucas-Dias J. M., Plaza A.: Robust minimum volume simplex analysis for hyperspectral unmixing. In *2014 Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)* (2014), IEEE, pp. 1582–1586.

[ApS18] ApS M.: MOSEK version 8.1 (2018). http://mosek.com/. Last accessed on 8 October 2018.

[ARV07] Amberg B., Romdhani S., Vetter T.: Optimal step nonrigid ICP algorithms for surface registration. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07* (2007), IEEE, pp. 1–8.

[BD09] Bioucas-Dias J. M.: A variable splitting augmented Lagrangian approach to linear spectral unmixing. In *1st Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, 2009. WHISPERS'09* (2009), IEEE, pp. 1–4.

[BP07] Baran I., Popović J.: Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics 26*, 3 (2007), 72:1–72:8.

[CCHM09] Chan T.-H., Chi C.-Y., Huang Y.-M., Ma W.-K.: A convex analysis-based minimum-volume enclosing simplex algorithm for hyperspectral unmixing. *IEEE Transactions on Signal Processing 57*, 11 (2009), 4418–4432.

[CHV17] Chakraborty R., Hauberg S., Vemuri B. C.: Intrinsic Grassmann averages for online linear and robust subspace learning. In *Computer Vision and Pattern Recognition (CVPR)* (2017), Vol. 1, IEEE, pp. I–I.

[Cra94] Craig M. D.: Minimum-volume transforms for remotely sensed data. *IEEE Transactions on Geoscience and Remote Sensing 32*, 3 (May 1994), 542–552.

[CTA*14] Cao V.-T., Tran T.-T., Ali S., Laurendeau D.: Nonrigid registration for deformable objects. In *2014 International Conference on Computer Graphics Theory and Applications (GRAPP)* (2014), IEEE, pp. 1–10.

[CZ08] Chang W., Zwicker M.: Automatic registration for articulated shapes. *Computer Graphics Forum 27* (2008), 1459–1468.

[CZ09] Chang W., Zwicker M.: Range scan registration using reduced deformable models. *Computer Graphics Forum 28* (2009), 447–456.

[CZ11] Chang W., Zwicker M.: Global registration of dynamic range scans for articulated model reconstruction. *ACM Transactions on Graphics 30*, 3 (2011), 26.

[DATTS08] De Aguiar E., Theobalt C., Thrun S., Seidel H. -P.: Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum 27* (2008), 389–397.

[DK92] DuPré A. M., Kass S.: Distance and parallelism between flats in r^n. *Linear Algebra and its Applications 171* (1992), 99–107.

[EAS98] Edelman A., Arias T. A., Smith S. T.: The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications 20*, 2 (1998), 303–353.

[FMHS15] Fu X., Ma W.-K., Huang K., Sidiropoulos N. D.: Blind separation of quasi-stationary sources: Exploiting convex geometry in covariance domain. *IEEE Transactions on Signal Processing 63*, 9 (2015), 2306–2320.

[GB08] Gain J., Bechmann D.: A survey of spatial deformation from a user-centered perspective. *ACM Transactions on Graphics 27*, 4 (2008), 107.

[GLS10] Gao J., Langberg M., Schulman L. J.: Clustering lines in high-dimensional space: Classification of incomplete data. *ACM Transactions on Algorithms 7*, 1 (2010), 8.

[HGPP13] Hendrix E. M., García I., Plaza J., Plaza A.: On the minimum volume simplex enclosure problem for estimating a linear mixing model. *Journal of Global Optimization 56*, 3 (2013), 957–970.

[HSK15] Holden D., Saito J., Komura T.: Learning an inverse rig mapping for character animation. In *Symposium on Computer Animation* (New York, NY, USA, 2015), ACM, pp. 165–173.

[HTRS10] Hasler N., Thormählen T., Rosenhahn B., Seidel H.-P.: Learning skeletons for shape and pose. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2010), ACM, pp. 23–30.

[HYL*03] Ho J., Yang M.-H., Lim J., Lee K.-C., Kriegman D.: Clustering appearances of objects under varying illumination conditions. In *Computer Vision and Pattern Recognition (CVPR)* (2003), Vol. 1, IEEE, pp. I–I. https://ieeexplore.ieee.org/abstract/document/1211332.

[JBK*12] Jacobson A., Baran I., Kavan L., Popović J., Sorkine O.: Fast automatic skinning transformations. *ACM Transactions on Graphics* (2012).

[JBPS11] Jacobson A., Baran I., Popović J., Sorkine O.: Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics 30*, 4 (2011), 78:1–78:8.

[JDKL14] Jacobson A., Deng Z., Kavan L., Lewis J.: Skinning: Real-time shape deformation. In *ACM SIGGRAPH* (2014), Vol. 22.

[JT05] James D. L., Twigg C. D.: Skinning mesh animations. *ACM Transactions on Graphics 24* (2005), 399–407.

[Kra88] Kraft D.: *A Software Package for Sequential Quadratic Programming*. Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt, 1988.

[KSO10] Kavan L., Sloan P.-P., O'Sullivan C.: Fast and efficient skinning of animated meshes. *Computer Graphics Forum 29*, 2 (2010), 327–336.

[LCWC16] Lin C.-H., Chi C.-Y., Wang Y.-H., Chan T.-H.: A fast hyperplane-based minimum-volume enclosing simplex algorithm for blind hyperspectral unmixing. *IEEE Transactions on Signal Processing 64*, 8 (2016), 1946–1961.

[LCY*18] Lu X., Chen H., Yeung S.-K., Deng Z., Chen W.: Unsupervised articulated skeleton extraction from point set sequences captured by a single depth camera. In *Proceedings of AAAI 2018* (2018).

[LD12] Le B. H., Deng Z.: Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics 31*, 6 (2012), 199.

[LD13] Le B. H., Deng Z.: Two-layer sparse compression of dense-weight blend skinning. *ACM Transactions on Graphics 32*, 4 (2013), 124.

[LD14] Le B. H., Deng Z.: Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics 33*, 4 (2014), 84.

[LDJ*19] Luo G., Deng Z., Jin X., Zhao X., Zeng W., Xie W., Seo H.: 3D mesh animation compression based on adaptive spatio-temporal segmentation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2019), I3D '19, ACM, pp. 10:1–10:10.

[LML*15] Lin C.-H., Ma W.-K., Li W.-C., Chi C.-Y., Ambikapathi A.: Identifiability of the simplex volume minimization criterion for blind hyperspectral unmixing: The no-pure-pixel case. *IEEE Transactions on Geoscience and Remote Sensing 53*, 10 (2015), 5530–5546.

[LMR*15] Loper M., Mahmood N., Romero J., Pons-Moll G., Black M. J.: SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 34*, 6 (Oct. 2015), 248:1–248:16.

[LS10] Landreneau E., Schaefer S.: Poisson-based weight reduction of animated meshes. *Computer Graphics Forum 29*, 6 (2010), 1945–1954.

[LS13] Lee E., Schulman L. J.: Clustering affine subspaces: Hardness and algorithms. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms* (2013), Society for Industrial and Applied Mathematics, pp. 810–827.

[MBDC*14] Ma W.-K., Bioucas-Dias J. M., Chan T.-H., Gillis N., Gader P., Plaza A. J., Ambikapathi A., Chi C.-Y.: A signal processing perspective on hyperspectral unmixing: Insights from remote sensing. *IEEE Signal Processing Magazine 31*, 1 (2014), 67–81.

[MK16] Mukai T., Kuriyama S.: Efficient dynamic skinning with low-rank helper bone controllers. *ACM Transactions on Graphics 35*, 4 (2016), 36.

[MRBD*14] Marrinan T., Ross Beveridge J., Draper B., Kirby M., Peterson C.: Finding the subspace mean or median to fit your need. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1082–1089.

[NW06] Nocedal J., Wright S. J.: *Numerical Optimization* (2nd edition). Springer, New York, 2006.

[PB11] Papazov C., Burschka D.: Deformable 3D shape registration based on local similarity transforms. *Computer Graphics Forum 30* (2011), 1493–1502.

[SE09] Schneider D. C., Eisert P.: Fast nonrigid mesh registration with a data-driven deformation prior. In *2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)* (2009), IEEE, pp. 304–311.

[SY07] Schaefer S., Yuksel C.: Example-based skeleton extraction. In *Symposium on Geometry Processing* (2007), pp. 153–162.

[TE18] Thiery J.-M., Eisemann E.: ARAPLBS: Robust and efficient elasticity-based optimization of weights and skeleton joints for linear blend skinning with parametrized bones. *Computer Graphics Forum 37*, 1 (2018), 32–44.

[TKW16] Townsend J., Koep N., Weichwald S.: Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research 17*, 137 (2016), 1–5.

[TLG16] Tan J., Lien J.-M., Gingold Y.: Decomposing images into layers via RGB-space geometry. *ACM Transactions on Graphics 36*, 1 (2016), 7.

[Vav09] Vavasis S. A.: On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization 20*, 3 (2009), 1364–1377.

[WD97] Wales D. J., Doye J. P.: Global optimization by basin-hopping and the lowest energy structures of Lennard–Jones clusters containing up to 110 atoms. *Journal of Physical Chemistry A 101*, 28 (1997), 5111–5116.

[WP02] Wang X. C., Phillips C.: Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2002), ACM, pp. 129–138.

[YMYK14] Yoshiyasu Y., Ma W.-C., Yoshida E., Kanehiro F.: As-conformal-as-possible surface registration. *Computer Graphics Forum 33* (2014), 257–267.

[ZSWL12] Zhang T., Szlam A., Wang Y., Lerman G.: Hybrid linear modeling via local best-fit flats. *International Journal of Computer Vision 100*, 3 (2012), 217–240.