Hyperspectral Inverse Skinning

Team 21

ARPAN VERMA and HARSHIT GUPTA

1 INTRODUCTION

Linear blend skinning (LBS) is a popular method for efficient and realistic animation, where a vertex's position is computed as a weighted average of transformation matrices, simplifying animation data handling. In inverse skinning, given only observed deformations, the goal is to retrieve skinning weights and transformations. Traditional methods for this can struggle with sparsity and rigidity constraints. This work reframes inverse skinning as a high-dimensional optimization problem, where transformations are vertices in a simplex. By fitting the smallest-volume simplex to the observed data, our method minimizes error, enabling high-fidelity skinning solutions with applications in areas like hyperspectral imaging.

2 LITERATURE REVIEW

2.1 Skinning Decomposition

The early research on inverse Linear Blend Skinning (LBS) began with a study by Wang and Phillips [10] to address artifacts like joint collapse and "candy-wrapper" distortions. They improved LBS by associating unique weights for each transformation matrix entry. [6] advanced the study of inverse LBS by introducing an approach that extracted rotations from mesh triangles and used clustering to estimate initial bone transformations. Further developments in inverse LBS research included determining bone skeleton hierarchies through clustering of transformations (e.g., [9]). The authors highlight a distinct approach by using convex geometric structures in transformation matrices as points in a high-dimensional space, which does not involve skeleton hierarchies or rigid transformations. This method uses the vertices of a convex hull (simplex) as handles, optimizing error with fewer bones than previous clustering-based approaches. [7] explored inverse skinning through matrix factorization, optimizing transformations, skinning weights, and rest pose positions efficiently by assuming weight sparsity. Le and Deng's work addressed skinning decomposition, weight compression, and automatic rigging with a bone skeleton. In comparison, this method avoids rigid transformations and achieves faster results with lower errors.

2.2 Surface Registration

Both our approach and the aforementioned inverse LBS approaches assume vertex correspondences across all mesh poses. However, some related work does not require this assumption. [3] presented a series of works on surface registration for articulated shapes, employing LBS with binary weights to segment the surface at joints and restrict transformations to rigid motions. Iterative Closest Point (ICP) has been extensively studied in surface registration, with [1] extending ICP to nonrigid registration using adjustable stiffness regularization. For 3D shape registration, deformable registration with known mesh topology is of interest, with works utilizing spin-image methods for reliable correspondences and as-rigid-as-possible deformation.

2.3 Subspace Clustering

Subspace clustering studies, such as [11] and [5], address the clustering of points onto flats or closest-point clusters, though they do not consider cases where both input and output data are flats. Our approach, in contrast,

Authors' address: Arpan Verma, arpan22105@iiitd.ac.in; Harshit Gupta, harshit22209@iiitd.ac.in.

addresses clustering within an LBS subspace, where all transformations lie on the same flat. The flat optimization problem involves finding the optimal flat distance between deformed and undeformed vertex positions.

2.4 Hyperspectral Unmixing

Hyperspectral unmixing, or unsupervised hyperspectral unmixing, aims to recover spectral signatures (endmembers) and their abundances from hyperspectral images. Methods developed by [4] and others seek to solve this problem by finding the minimum-volume simplex enclosing observed points, despite challenges of local minima. Algorithms now efficiently address this problem even with outliers [2]. Nonnegative Matrix Factorization (NMF), an equivalent problem to hyperspectral unmixing, is NP-complete in general, though separable NMF problems are polynomially solvable. In our case, we avoid the pure-pixel assumption and nonnegative constraints.

2.5 Hyperspectral Inverse Skinning

Inverse skinning can be reformulated as a problem in high-dimensional Euclidean space. The transformation matrices applied to a vertex across all poses can be conceptualized as points in this high-dimensional space. Liu et al. approach the inverse linear blend skinning (LBS) problem by finding a tight-fitting simplex around these points, a problem that has been well-studied in hyperspectral imaging. Although transformation matrices are not directly observed, the 3D positions of a vertex across all poses define an affine subspace, or flat. This leads to the formulation of a "closest flat" optimization problem to identify points on these flats. Subsequently, they compute a minimum-volume enclosing simplex, with vertices corresponding to the transformation matrices and barycentric coordinates representing the skinning weights. This method enables the creation of LBS rigs that achieve state-of-the-art reconstruction error and compression ratios for mesh animation sequences. Their proposed solution does not account for weight sparsity or the rigidity of the recovered transformations. They provide observations and insights into the closest flat problem, noting that its ideal solution and optimal LBS reconstruction error remain open questions [8].

3 MILESTONES

The following milestones were established to guide the implementation of the Hyperspectral Inverse Skinning project:

S. No.	Milestone	Member
	Interim Evaluation	
1	Framework design	Arpan Verma
2	Implementing flat optimization	Arpan Verma
3	Hyperspectral data collection	Harshit Gupta
4	Implementing the Initial Guess Code	Harshit Gupta
	Final evaluation	
5	Handle Transformation Estimation	Arpan Verma
6	Minimum Volume Enclosing Simplex Algorithm	Arpan Verma
7	Skinning weights estimation	Harshit Gupta
8	Results Analysis and Testing	Harshit Gupta
9	Exploring Future Works	Arpan Verma and Harshit Gupta

4 MILESTONES ACHIEVED FOR PROJECT EVALUATION-II:

We have completed the following Milestones:

- Framework Design.
- Data Collection and visualization.
- Initial Guess Code.
- Apply Per Vertex Transformation/ Transformation Estimation.
- Implement Flat Optimization. ‘

We have completed all the set milestones for this deadline and also moved forward and completed the first milestone of the Final Evaluation.

5 ALGORITHMS IMPLEMENTED:

The algorithms implemented were mainly of

Stage 1: Per-Vertex Transformation

The goal of this stage is to estimate a point $x \in \mathbb{R}^{12\cdot\#\text{poses}}$ for each vertex that minimizes the distance between its associated flat and a guessed handle flat.

Formulation. Each vertex i defines a flat as:

$$V_i x = v_i',$$

where:

- $V_i = I_{3\cdot\#\text{poses}} \otimes v_i^\top$,
- $\otimes$ denotes the Kronecker product,
- v_i is the undeformed vertex in homogeneous coordinates,
- v_i' is the stacked positions of v_i across all poses.

We compute an initial estimate x_i by solving the following optimization problem:

$$x_i =_x \sum_{j\in\{i\}\cup N(i)} \left\| \frac{1}{\|v_j\|^2} V_j^\top V_j (x - t_j) \right\|^2,$$

where:

- $N(i)$ denotes the one-ring neighborhood of vertex i,
- t_j is any valid transformation matrix in j's flat,
- The divisor $\|v_j\|^2$ normalizes the rows of V_j.

For computational simplicity, this can be rewritten as:

$$x_i =_x \sum_{j\in\{i\}\cup N(i)} \|V_j x - v_j'\|^2.$$

Stage 2: Flat Optimization

This stage aims to find the $(h-1)$-dimensional handle flat L that minimizes the squared Euclidean distance to all vertex flats.

Formulation. The handle flat L is parameterized as:

$$L = \{p + Bz : z \in \mathbb{R}^{h-1}\},$$

where:

- $p \in \mathbb{R}^{12\cdot\#\text{poses}}$ is a point on the flat,
- $B \in \mathbb{R}^{(12\cdot\#\text{poses})\times(h-1)}$ is a matrix whose columns span directions parallel to the flat.

We solve the following optimization problem:

$$\min_{p,B} \sum_i \min_{z_i} \|V_i(p + Bz_i) - v_i'\|^2,$$

where $z_i \in \mathbb{R}^{h-1}$ are the parameters for the closest point on L.

The solution for z_i is:

$$z_i = -(B^\top V_i^\top V_i B)^{-1} B^\top V_i^\top (V_i p - v_i').$$

Substituting z_i back, the problem becomes:

$$\min_{p,B} \sum_i \|V_i(p + Bz_i) - v_i'\|^2,$$

subject to the constraints:

$$1^\top w_i = 1, \quad \forall i,$$
$$w_i \geq 0, \quad \forall i.$$

6 CHALLENGES FACED:

We faced a few technical challenges like:

- Acquiring the data for different poses seemed challenging, as we had to find various images of a consistent object/animal and then break it down to matrices.
- The visualization of the images was also an overhead as the initial images were not clear enough, so we applied Bling Phong Model to render. A comparison of the cat poses with and without the Phong model is below:

Fig. 1. Comparison of initial images

- Initially, we tried to solve the flat optimization by using our own solving method. However, this did not converge, so we switched to an open-source solver Ceres.
- The image matrix that we had was high dimensional, so we used its sparse counterpart instead of the dense one.

7 INTERMEDIATE RESULTS:

We apply the algorithm to the image of a cat for different poses to check the intermediate results.

We obtained the below results for a cat image with 8 poses and bones parameter set to 10.

```
harshit@harshit-VirtualBox:~/Desktop/cg/HyperSpec$ ./skinning
[0s] Starting skinning computation...
[0s] Loading rest pose...
[0s] Rest pose loaded successfully
[0s] Loading pose files...
[0s] Found 8 pose files
[0s] All poses loaded successfully
[0s] Initializing skinning solver...
[0s] Computing skinning...
Computing per-vertex transformations...
Processed 7207/7207 vertices
Optimizing flat transformation...
iter      cost      cost_change  |gradient|   |step|    tr_ratio  tr_radius  ls_iter  iter_time  total_time
   0  3.431015e-11    0.00e+00    7.35e-04   0.00e+00   0.00e+00  1.00e+04        0    3.54e+01    3.54e+01
   1  2.679958e-11    7.51e-12    6.66e-04   3.29e-08   1.96e+00  3.00e+04        1    5.05e+01    8.59e+01
   2  2.745710e-11   -6.58e-13    6.66e-04   2.89e-08  -1.90e-01  1.50e+04        1    1.19e+00    8.71e+01
   3  2.239489e-11    4.40e-12    2.94e-04   2.88e-08   1.27e+00  4.50e+04        1    5.02e+01    1.37e+02
   4  1.937046e-11    3.02e-12    2.06e-04   1.83e-08   2.05e+00  1.35e+05        1    3.83e+01    1.76e+02
   5  3.532105e-11   -1.60e-11    2.06e-04   1.01e-08  -2.10e+01  6.75e+04        1    1.24e+00    1.77e+02
   6  2.264705e-11   -3.28e-12    2.06e-04   1.01e-08  -4.31e+00  1.69e+04        1    1.38e+00    1.78e+02
   7  3.198010e-11   -1.26e-11    2.06e-04   1.01e-08  -1.66e+01  2.11e+03        1    9.54e-01    1.79e+02
   8  3.254140e-11   -1.32e-11    2.06e-04   9.59e-09  -1.73e+01  1.32e+02        1    9.52e-01    1.80e+02
   9  1.979296e-11   -4.23e-13    2.06e-04   6.21e-09  -5.64e-01  4.12e+00        1    1.09e+00    1.81e+02
Flat optimization completed:
Ceres Solver Report: Iterations: 10, Initial cost: 3.431015e-11, Final cost: 1.937046e-11, Termination: CONVERGENCE
```

Fig. 2. Results After applying Per vertex Transform and Flat Optimization

EXPLANATION OF THE RESULTS

(1) **Explanation of the Results:**
- **iter:** Iteration number
- **cost:** Current value of the objective function (error)
- **cost_change:** Change in cost from the previous iteration
- **|gradient|:** Magnitude of the gradient
- **|step|:** Size of the optimization step
- **tr_ratio:** Trust region ratio
- **tr_radius:** Trust region radius
- **ls_iter:** Number of linear solver iterations
- **iter_time:** Time taken for this iteration
- **total_time:** Total time elapsed

(2) **Analysis of the Results:**
- **Initial cost:** 3.431×10^{-11} → **Final cost:** 1.937×10^{-11}
- Very small costs (order of 10^{-11}) suggest the optimization is working well.
- There's about a 43% reduction in error.

(3) **Convergence Behavior:**
- The optimizer takes small steps ($|\textbf{step}| \approx 10^{-8}$).
- Gradient magnitude is small (10^{-4}), indicating proximity to a minimum.
- Trust region radius stays relatively large, suggesting stability.

(4) **Performance:**
- Each iteration takes about 30 seconds.
- Total optimization time: ~ 127 seconds.
- Shows consistent progress with some oscillation.

(5) **Termination:**
- **Termination: CONVERGENCE**
- This indicates successful optimization - the solver found a local minimum.

SUMMARY

The optimization is finding the optimal flat transformation that:

- Minimizes the projection error for all vertex transformations.
- Maintains numerical stability (shown by small step sizes).
- Achieves good convergence (shown by decreasing cost).

After this optimization, the flat transformation matrix (**FlatTransform**) contains the optimal basis for representing all vertex transformations in a lower-dimensional space, which will be used in the minimum volume simplex computation that follows.

REFERENCES

[1] B. Amberg, S. Romdhani, and T. Vetter. 2007. Optimal step non-rigid ICP algorithms for surface registration. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1–8.

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

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

[4] M. D. Craig. 1994. Minimum-volume transforms for remotely sensed data. *IEEE Transactions on Geoscience and Remote Sensing* 32, 3 (1994), 542–552. https://doi.org/10.1109/36.297973

[5] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. 2003. Clustering appearances of objects under varying illumination conditions. In *Computer Vision and Pattern Recognition (CVPR)*, Vol. 1. IEEE, I–I.

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

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

[8] Songrun Liu, Jianchao Tan, Zhigang Deng, and Yotam Gingold. 2020. Hyperspectral Inverse Skinning. *Computer Graphics Forum (CGF)* 39, 6 (2020), 49–65. https://doi.org/10.1111/cgf.13903

[9] S. Schaefer and C. Yuksel. 2007. Example-based skeleton extraction. In *Symposium on Geometry Processing*. 153–162.

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

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