Mesh Wrap based on Affine-Invariant Coordinates

Fernando de Goes
fernando@pixar.com
Pixar Animation Studios

Alonso Martinez
alonso@pixar.com
Pixar Animation Studios

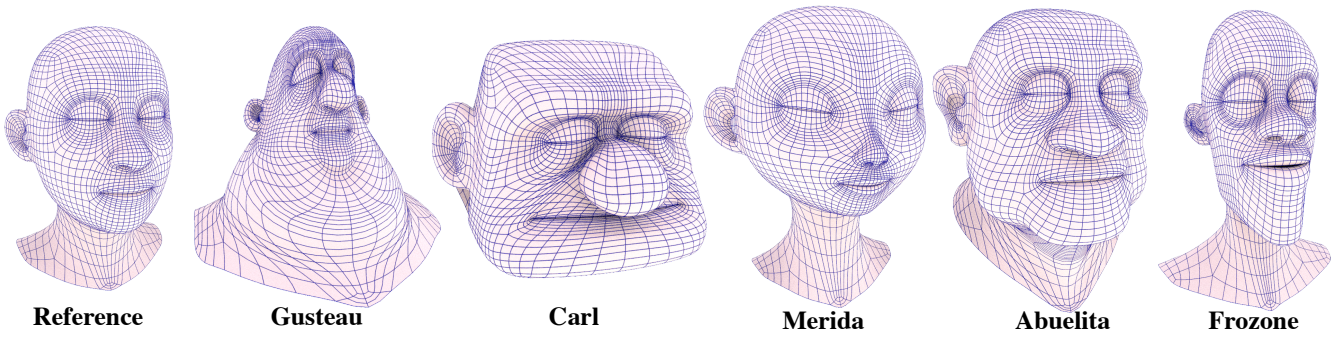

Figure 1: Our method is suited to wrap the mesh connectivity from a source model to target shapes. In this example, we used our wrap tool to share a reference mesh between characters from various feature films. ©Disney/Pixar.

ABSTRACT

We present a new technique to transfer the mesh connectivity between 3D models of different shapes. In contrast to prior work, our method is designed to wrap meshes under large, locally non-rigid deformations, which are commonly found in feature animations. To achieve this goal, we enrich the traditional iterative closest point scheme with mesh coordinates that parametrize the edge spans of a desired tessellation invariant to locally affine transformations. As a result, we produce surfaces that wrap a target geometry accurately, while resembling the patch layout of the source mesh. Our implementation also offers an interactive workflow to assist the authoring of curve correspondences. We employed this tool to wrap 600 humanoid assets to a reference mesh connectivity, spanning characters modeled over the last 15 years at Pixar.

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**.

KEYWORDS

mesh wrap, tessellation transfer, shape registration.

ACM Reference Format:
Fernando de Goes and Alonso Martinez. 2019. Mesh Wrap based on Affine-Invariant Coordinates. In *Proceedings of SIGGRAPH '19 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3306307.3328162

1 OUTLINE

We start by outlining our algorithm that wraps the mesh connectivity from a source model $\mathcal{S}$ to a target shape $\mathcal{T}$. Our method ouputs a mesh $\mathcal{M}$ that shares the same tessellation as $\mathcal{S}$ but fit to the geometry of $\mathcal{T}$ (Figure 3). We denote the vertex positions in $\mathcal{M}$ by $\{\mathbf{x}_i\}$ and stack them rowwise in a matrix $\mathbf{X}$. Similarly, we indicate the original mesh vertices in $\mathcal{S}$ by $\{\mathbf{y}_i\}$ and pack them in the matrix $\mathbf{Y}$. We also consider curve correspondences used as sparse hints to drive the deformation from $\mathcal{S}$ to $\mathcal{T}$ (§2). Equipped with this setup, we approach the task of wrapping a source mesh to a target shape as an iterative optimization that updates mesh vertices by alternating two steps (§5). The first step snaps $\mathbf{X}$ to $\mathcal{T}$ and computes a score to each projection (§3), while the second step relaxes $\mathbf{X}$ by minimizing the mesh distortion between $\mathcal{S}$ and $\mathcal{M}$ (§4), weighted by the projection scores and corresponding curves.

2 CURVE CORRESPONDENCES

To guide the wrap optimization, we support shape correspondences described by curves. We have found that curves are more descriptive than point-based features, since we can infer local stretching by comparing the arc-length parametrization between pairs of curves. Our implementation uses Houdini [Side Effects 2019] to draw curve strokes projected to the source and target shapes, and the correspondence is defined based on the stroke ordering. For the common case of humanoid models, we developed a custom user interface that displays suggestions for the curve placement (Figure 2). Our system tracks the selected suggestions and activates their correspondences following a predefined indexing (see supplemental video). We discretize corresponding curves with quadrature points sampled evenly on both source and target shapes. The target samples are represented by a matrix $\mathbf{Q}$ assigning each row to a sample location on $\mathcal{T}$. The source samples are encoded by a matrix $\mathbf{B}$ with rows set to the (generalized) barycentric coordinates that associate each sample with the vertices of $\mathcal{S}$. We can then reconstruct the sample positions on the wrap mesh $\mathcal{M}$ by minimizing $\|\mathbf{B}\,\mathbf{X} - \mathbf{Q}\|^2$.

3 FITTING

Our solver also accounts for the geometric discrepancy between the wrap mesh $\mathcal{M}$ and the target shape $\mathcal{T}$. To this end, we abstract $\mathcal{T}$ by

SIGGRAPH '19 Talks, July 28 - August 01, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-6317-4/19/07.
https://doi.org/10.1145/3306307.3328162

defining a projection operator Π that maps any mesh vertex $\mathbf{x}_i$ to its closest point $\mathbf{p}_i = \Pi(\mathbf{x}_i)$ on $\mathcal{T}$, which can be implemented efficiently using a bounding volume hierarchy. Similar to [Zhou et al. 2016], we relate each vertex projection to a score $m_i = 1/(1+\mu\|\mathbf{p}_i - \mathbf{x}_i\|^2)$, with values in the range $[0,1]$ based on the proximity to $\mathcal{T}$. The parameter μ is a scalar amount that stiffens the attachment between $\mathbf{x}_i$ and $\mathbf{p}_i$. By arranging the projection points rowwise in a matrix $\mathbf{P}$ and their respective scores in a diagonal matrix $\mathbf{M}$, we compute the fitting error between $\mathcal{M}$ and $\mathcal{T}$ via $\|\mathbf{M}(\mathbf{X}-\mathbf{P})\|^2$.

4 MESH DISTORTION

A key component in our method is the distortion term that quantifies the mesh deformation moving the points in $\mathcal{M}$ from $\mathbf{Y}$ to $\mathbf{X}$. Since Pixar characters are stylized and vary significantly between shows (Figure 1), we sought for a distortion model that enables locally non-rigid deformations. The distortion model should also promote the layout of the source tessellation so that the artistically-crafted edge flows can be resembled on the optimized mesh. To address these conflicting goals, we adopted the affine-invariant coordinates introduced by Budninskiy et al. [2017]. We compute these coordinates once by preprocessing the source mesh $\mathcal{S}$. For every vertex i in $\mathcal{S}$, we first collect a stencil of size n containing every vertex j that shares a face with i, and then assemble a matrix $\mathrm{d}\mathbf{Y}_i = \left[\cdots, \mathbf{y}_j - \mathbf{y}_i, \cdots\right]$ of size $3 \times n$. The local coordinates associated with the vertex i are set to a matrix $\mathbf{W}_i$ of size $n \times (n-3)$ that spans the nullspace of $\mathrm{d}\mathbf{Y}_i$, i.e., $\mathrm{d}\mathbf{Y}_i\,\mathbf{W}_i = 0$. The row-vectors in $\mathbf{W}_i$ define a $(n-3)$-dimensional embedding of the vertices within the stencil of i that captures its local structure agnostic to affine transformations. We compute $\mathbf{W}_i$ by extracting the right-singular vectors corresponding to zero singular values of the singular value decomposition (SVD) of $\mathrm{d}\mathbf{Y}_i$. We then construct our distortion objective as a least-squares function $\sum_i \|\mathrm{d}\mathbf{X}_i\mathbf{W}_i\|^2$ that evaluates how the source local coordinates $\{\mathbf{W}_i\}$ conform to the wrapped vertex stencils $\{\mathrm{d}\mathbf{X}_i\}$. One can further expand this expression into a quadratic form $\mathbf{X}^\top\mathbf{L}\,\mathbf{X}$, where $\mathbf{L}$ is a Laplacian-like sparse matrix containing the affine-invariant coordinates of $\mathcal{S}$. Compared to prior work, our formulation leads to a convex distortion minimization that supports a broader range of deformations, while preserving the relative spacing between nearby vertices.

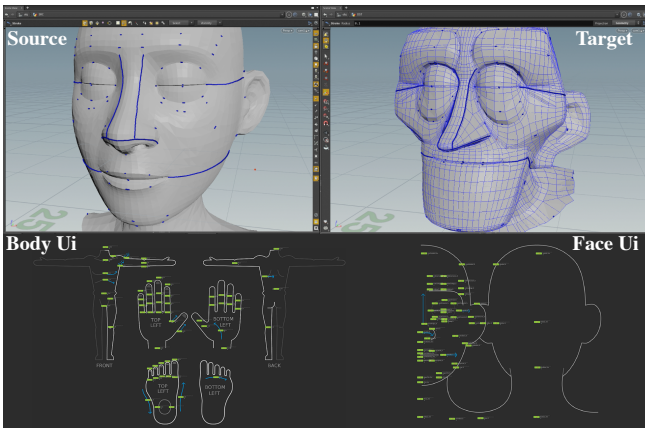

Figure 2: We developed a custom interface that provides suggestions to place curve correspondences on humanoid faces and bodies. ©Disney/Pixar.

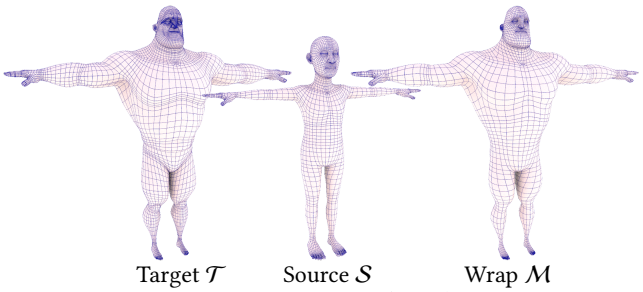

Figure 3: We produce a wrap model (right) by transferring a source mesh (center) to a target shape (left). ©Disney/Pixar.

5 NUMERICS

We initialize our algorithm by setting $\mathbf{X} = \mathbf{Y}$. In every iteration, we first compute $\mathbf{P}$ by projecting $\mathbf{X}$ to $\mathcal{T}$, then we estimate the vertex scores and set them to $\mathbf{M}$, finally we update $\mathbf{X}$ by optimizing a mix of mesh distortion, fitting error, and curve correspondences:

$$\min_{\mathbf{X}} \mathbf{X}^\top\mathbf{L}\,\mathbf{X} + \mu\|\mathbf{M}(\mathbf{X}-\mathbf{P})\|^2 + \kappa\|\mathbf{B}\,\mathbf{X} - \mathbf{Q}\|^2, \qquad (1)$$

where μ is the score parameter and κ is a stiffness amount (set to 0.1) that enforces the curve correspondences. Since this is a quadratic minimization, we compute $\mathbf{X}$ by solving the sparse linear system:

$$\left(\mathbf{L} + \mu\mathbf{M}^\top\mathbf{M} + \kappa\mathbf{B}^\top\mathbf{B}\right)\mathbf{X} = \mu\mathbf{M}^\top\mathbf{M}\,\mathbf{P} + \kappa\mathbf{B}^\top\mathbf{Q}. \qquad (2)$$

We implemented this linear solve using a Cholesky factorization followed by numerical updates at every iteration that incorporate the latest projection scores. We also structured our alternating steps in rounds in order to ramp the contribution of fitting term up as the optimization progresses. We start with a small stiffness amount ($\mu = 0.1$) and scale it up by an order of magnitude every 10 iterations. Our optimization completes when the largest projection residual is less than 10^{-4} or a maximum iteration count (set to 100) is reached.

6 RESULTS

Figure 1 presents a series of 3D faces collected from various Pixar shows sharing the same mesh connectivity computed by our method. Observe that our results reproduce a broad range of shapes, while retaining the underlying mesh structure. In the supplemental video, we include an animation that blends the face shapes from several Pixar characters wrapped by our solver. Figure 3 shows an example of a body shape wrapped by our algorithm. We have employed our tool to transfer show-specific assets to standard tessellations. In particular, we have successfully wrapped a reference mesh to every humanoid character from Pixar feature films starting from the original Incredibles (2004), in a total of 600 models. By sharing mesh connectivities, we have also assisted the generation of new background characters. Our optimization takes in average 5 rounds of 10 iterations to converge, reporting 3 seconds on meshes of 10k vertices, clocked on a 2.3 GHz Intel Xeon E5-2699.

REFERENCES

M. Budninskiy, B. Liu, Y. Tong, and M. Desbrun. 2017. Spectral Affine-Kernel Embeddings. *Comput. Graph. Forum (SGP)* 36, 5 (2017), 117–129.
Side Effects. 2019. Houdini Engine. (2019). http://www.sidefx.com
Q.-Y. Zhou, J. Park, and V. Koltun. 2016. Fast Global Registration. In *ECCV*. 766–782.