Feature-Based Texture Stretch Compensation for 3D Meshes

Stéphane Grabli Kevin Sprout Yuting Ye
Industrial Light and Magic*

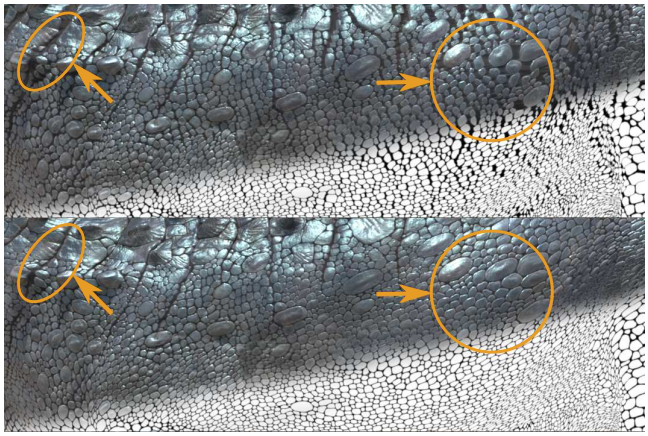

Figure 1: *Dinosaur skin under 3D deformation with (top) and without (bottom) texture stretch compensation.* ©*Universal Pictures.*

Abstract

One subtle artifact which still often reveals the synthetic nature of a digital creature on screen is the stretching of a supposedly rigid feature, such as a dinosaur scale or a callus, under deformation. We introduce a technique which attempts to preserve the shape of user-defined features when a 3D mesh deforms. Our approach makes no restriction on the type of features or their distribution—it can handle features of very high resolution and it is designed to fit in a texture-painting driven pipeline.

1 Overview

Creatures often have rigid patterns on their skins, such as scales or scars. When the 3D models are undergoing arbitrary deformations, these skin features need to move rigidly to retain their shapes. Enforcing this rigidity in the 3D deformation is impractical as this requires prohibitively high resolution meshes which impair solver stability, and limit artistic freedom. Procedural modeling techniques are effective for repetitive rigid features, such as scales on dinosaurs [Heckenberg et al. 2014]. Unfortunately, these techniques do not fit in our texture-painting driven pipeline. Based on our production requirements, we propose a texture reparameterization approach that deforms 2D textures to compensate for non-rigid deformation in the 3D mesh. This way, we allow for arbitrary 3D deformations, feature shapes and distributions, without affecting our modeling and texture-painting pipeline. Gal *et al.* [2006] solve a similar problem in 2D. Our approach takes inspiration from this work. However, our treatment of the problem using reparameterization, as described in section 2, is specific to 3D meshes. The flexibility afforded by solving the 3D-based mesh deformation constraints in 2D at texture resolution is a key advantage of our approach over previous ones.

*e-mail:{sgrabli,sprout,yye}@ilm.com

2 Reparameterization Algorithm

Given a parameterized mesh $\tilde{M} : (u, v) \rightarrow (x, y, z)$, and a texture mapping function $\tilde{T} : (u, v) \rightarrow (s, t)$, texture stretching at a given point (s, t) can be expressed as: $J_{\tilde{M}\circ\tilde{T}^{-1}}(s, t)$, where J denotes the Jacobian matrix. As $\tilde{M}$ animates into a deformed mesh M, the texture stretching changes inhomogeneously across the mesh. In order to preserve the shape of rigid features on M, we need to introduce a new parameterization $T : (u, v) \rightarrow (s, t)$ such that at any point where $\tilde{T}$ resolves to a rigid feature, the apparent stretching of the feature on the deformed mesh is the same as the rest mesh, up to a rotation. This constraint can be expressed as $J_{M\circ T^{-1}}(s, t) = RJ_{\tilde{M}\circ\tilde{T}^{-1}}(s, t)$, where R is a 3D rotation. Using the chain rule, the desired gradient on the unknown function T^{-1} is: $J_{T^{-1}}(s, t) = [(J_M(u, v)]^{-1}R[J_{\tilde{M}}(u, v)J_{\tilde{T}^{-1}}(s, t)]$.

In practice, we tessellate the texture space adaptively based on the grayscale feature map, and approximate the partial derivatives using finite differences on triangle edges. We end up with a similar Laplacian system as in Gal *et al.* [2006], where deformed 2D coordinates are solved based on gradient constraints, as specified above, and boundary constraints (we enforce $T = \tilde{T}$ at mesh boundaries). In addition, we introduce extra positional constraints at feature region centroids to maintain feature distribution and improve temporal coherence. The solution is a new set of (s, t) coordinates of each texture pixel used for texture look-up in rendering.

3 Implementation in Production and Results

Hero creatures are typically partitioned into around 100 tiles, each tile being mapped to a $4K \times 4K$ texture or higher. For each tile at each frame, we compute and store the reparameterization T as a high resolution texture. To minimize resource requirement, we carry out precomputations on the rest mesh before render time, and store the result with the mesh. As a result, the reparameterization can be computed more efficiently and stored only temporarily. The system was tested on a set of hero shots on *Jurassic World* and is giving good results. Many challenges remain before deploying it more broadly. Specifically, the current implementation can fail at preserving the macro-scale shape of complex features, which can be particularly objectionable for close-up shots. We hope to improve this situation by taking shape boundaries into consideration as well as better handle cases where features are simultaneously squashed and stretched.

References

GAL, R., SORKINE, O., AND COHEN-OR, D. 2006. Feature-aware texturing. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*.

HECKENBERG, D., HEGARTY, J., AND LEBLANC, J. P. 2014. Reptile: How to skin a dinosaur. In *ACM SIGGRAPH 2014 Talks*.