Computers & Graphics 34 (2010) 430–440

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Real-time path-based surface detail

Carles Bosch *, Gustavo Patow

Grup de Geometria i Gràfics, Universitat de Girona, E-17071 Girona, Spain

ARTICLE INFO

Article history:
Received 4 December 2009
Received in revised form
3 March 2010
Accepted 15 April 2010

Keywords:
Surface detail
Real-time rendering
Vector graphics

ABSTRACT

We present a GPU algorithm to render path-based 3D surface detail in real-time. Our method models these features using a vector representation that is efficiently stored in two textures. First texture is used to specify the position of the features, while the second texture contains their paths, profiles and material information. A fragment shader is then proposed to evaluate this data on the GPU by performing an accurate and fast rendering of the details, including visibility computations and antialiasing. Some of our main contributions include a CSG approach to efficiently deal with intersections and similar cases, and an efficient antialiasing method for the GPU. This technique allows application of path-based features such as grooves and similar details just like traditional textures, thus can be used onto general surfaces.

1. Introduction

Up to now, real-time visualization of surface detail has been limited to the generation and usage of geometry or sampled data structures as in bump mapping [1], displacement mapping [2], or relief mapping [3,4], which show aliasing problems for close views and do not provide a correct solution for filtering in far-views. On the other hand, vector textures are gaining popularity [5,6], but are limited to flat 2D representations without encoding other 3D information besides normal map perturbation techniques [7]. Visibility and occlusion issues in these representations have never been treated in the context of vector-based representations.

This paper presents a feature-based per-pixel displacement mapping technique. It builds upon previous vector texture representations and per-pixel displacement mapping techniques, and makes a step forward to achieve a robust and flexible real-time vector based displacement mapping algorithm (see Fig. 1). The presented method is capable to visualize geometry details like scratches, cracks, grooves and extremely sharp edged features like bricks or edges on manufactured objects. Also, our method allows accurate visualization of these path-based features in a single pass algorithm by performing a single write per pixel.

Approach: Our real-time method computes 3D geometric detail in texture-space by using a continuous representation that is stored in two textures, without relying on additional geometry (albeit with an increase in computational cost). See Fig. 2. We use techniques derived from the usage of vector textures in the GPU to store the geometry and properties of the features, and evaluate them in real-time. The first texture is a grid that specifies the positions of the features, providing pointers to the second texture which contains the feature paths, profiles and material information. A fragment shader at the GPU evaluates this data, and generates an accurate and fast rendering by using a constructive solid geometry (CSG) analogy.

Contributions: The new method presented here is the first real-time approach to present 3D vector-based surface detail other than flat textures. In particular, it allows accurate visualization of path-based features, although this could be extended to other features as well. We also introduce a CSG analogy that is both flexible and powerful. The visualization is done in a single pass and by performing only one write per pixel. As a consequence, we have a low-bandwidth coherent memory access, which is advantageous for many-core architectures. Also, it has efficient approximate antialiasing which allows the rendering of the features from close to distant views. We use two main approximate filtering techniques, called region-sampling and supersampling. Both techniques are used in combination to solve both visibility and shadowing antialiasing issues.

Limitations: Our path-based feature representation shares a few limitations with other vector-based representations [6]. For example, it assumes a static layout of features, as a dynamic situation would require re-encoding features at each time step, which is very fast but is not capable of real-time results. Also, a feature segment can be replicated in many texels it overlaps, but in our experience there is almost no storage overhead. Also, each texel may have a different number of features, thus requiring an indirection scheme to avoid data sparseness. Finally, we require the features and the object surface where they are applied to have low curvature, in order to obtain correct visibility computations.

* Corresponding author. Tel.: +34 972 418832; fax: +34 972 418792.
E-mail addresses: cbosch@ima.udg.edu (C. Bosch), dagush@ima.udg.edu (G. Patow).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.04.001

Fig. 1. The Knight Champion rendered from different viewpoints and distances using our approach (left images, 59–78 fps) and relief mapping (right images, 57–81 fps) to simulate the armor engravings. Observe how our antialiasing strategy correctly reproduces the grooves visibility while relief mapping produces noticeable artifacts (insets).

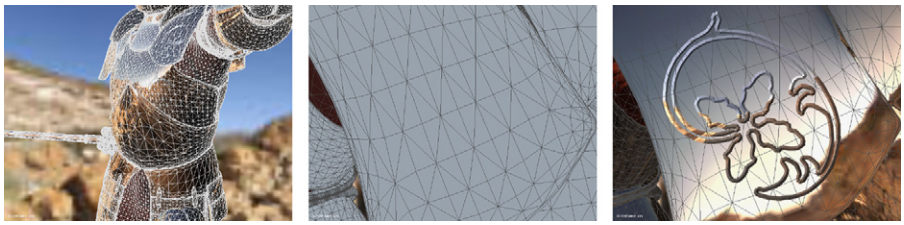

Fig. 2. The presented method is capable to visualize geometry details like scratches, cracks, grooves and extremely sharp edged features without the amount of geometry needed to get the details. This clearly shows the advantages of texture-based methods. Left: the full model. Middle: the base geometry. Right: the generated detail.

2. Previous work

The method we present in this paper is closely related to surface detail techniques, real-time vector texture representations, and scratches and grooves modeling and rendering.

In general, macro-geometric models use general techniques that allow the simulation of different kinds of surface details, such as bump mapping [1], displacement mapping [2], relief mapping [3] or parallax mapping [8,4], among others. For an in-depth survey on displacement mapping techniques on the GPU, refer to [9]. These macro-geometry models suffer from resolution problems and are not able to correctly simulate high frequency or very close details. Compared to these approaches, the method presented here addresses these issues in an efficient and natural way, as can be seen in Fig. 1.

Our method is also related to real-time vector graphics, which always have had a great appealing because of their seamless scaling capabilities. In [10], they require a heavy preprocessing that includes segmenting the contour and embedding each segment in a triangle. Other schemes present limitations in the number of primitives allowed for each texel: a few line segments [11–13], an implicit bilinear curve [14], two quadratic segments [7], or a fixed number of corner features [5]. Also, Parilov et al. [7] presented a method for rendering normal maps with discontinuities, which was restricted to path patterns with no "T" junctions, no occlusion computations and with a unique profile for all features. All these methods share a drawback of limiting the number of allowed primitives, which is bad for areas which require high detail. One solution would be to use a finer lattice, but this would greatly increase storage needs. We use a variable-length texel representation that allows for patterns of arbitrary complexity, having none of the above-mentioned restrictions. Our approach stores feature paths in a way similar to [5,6], but here it is used to store a 3D structure, not a 2D one as in the mentioned methods.

Scratch models simulate small isolated grooves that are visible but where their geometry is imperceptible. These models combine a 2D texture with an anisotropic BRDF model. The texture specifies the position of each scratch, while the BRDF is used to compute their reflection. Examples are the works by Merillou et al. [15] and Bosch et al. [16]. Recently, an extension has been proposed to deal with more general path-based features, which removes the limitations on the size of the features or their geometric cases (e.g. intersections) [17]. Our method is based on a similar idea, but our rendering techniques are GPU-friendly, which results in real-time frame rates for a similar rendering quality.

Porumbescu et al. [18] introduced shell maps, which allow to add arbitrary small-scale surface detail to a triangulated object, but not at interactive rates. Later, [19] introduced techniques that allowed the obtention of interactive frame-rates. The technique presented here is not as general as these, but allows real-time frame-rates to be obtained.

Naturally, details like grooves can also be included in the geometry model of the objects. Such approach is usually taken for interactive sculpting or editing. Clearly, our method avoids the fine discretization required by those methods by transferring those evaluations to the pixel shader, and thus lowering bandwidth needs without performing scattered updates to the framebuffer, as would happen with geometry-based approaches.

3. Overview

In this work we represent 3D geometric detail with a continuous representation based on paths and cross-sections in texture space. This information is stored in two textures: the first one is a grid overlaid on the surface features, where each cell provides the positions of the features themselves and references the second texture. This second texture contains the geometry and properties of the feature path profiles, and material information. As mentioned, the proposed method does not need additional geometry to represent these surface details.

At runtime, our algorithm performs a search in texture space for the intersection between the viewing ray (transformed to tangent coordinates) and the features, sequentially evaluating the contents of each texel along the projected ray. In order to perform an accurate and fast rendering of the features, we use a constructive solid geometry (CSG) analogy to compute the intersection between each viewing ray and the features in each