3D-Mesh Cutting Based On Fracture Photographs

Vincent Gaubert
ESGI
Paris, France
arckantox@gmail.com

Enki Londe
ESGI
Paris, France
enkilonde@gmail.com

Thibaut Poittevin
ESGI
Paris, France
thibaut.poittevin@gmail.com

Alain Lioret
ESGI / Université Paris 8
Paris, France
alainlioret@gmail.com

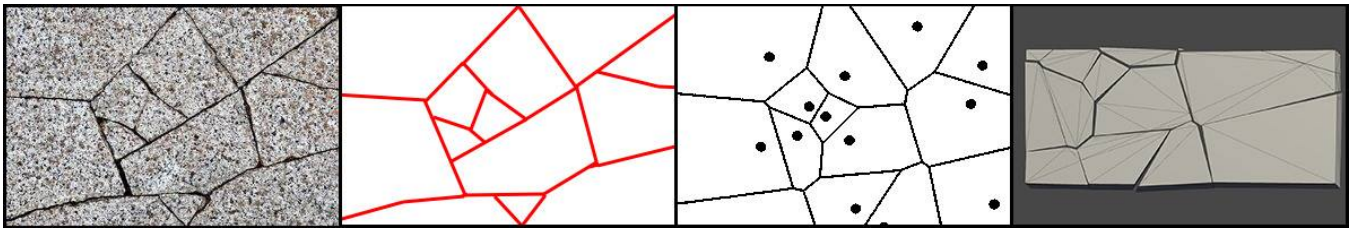

Figure 1: Example of analysis and reproduction of a real fracture. From left to right: (a) Original picture. (b) Analysis of the picture and information extraction. (c) One possible graph found by our tool after training. (d) Application of the result on a 3D mesh.

ABSTRACT

We propose a new approach to 3D mesh fracturing for the fields of animation and game production. Through the use of machine learning and computer vision to analyze real fractures we produced a solution capable of creating realistic fractures in real-time.

CCS CONCEPTS

• **Computing methodologies** → *Learning linear models*; *Mesh models*;

KEYWORDS

Real-time, Machine Learning, Geometry, Computer Vision

ACM Reference format:
Vincent Gaubert, Enki Londe, Thibaut Poittevin, and Alain Lioret. 2018. 3D-Mesh Cutting Based On Fracture Photographs. In *Proceedings of SIGGRAPH '18 Posters, Vancouver, BC, Canada, August 12-16, 2018,* 2 pages.
https://doi.org/10.1145/3230744.3230759

1 INTRODUCTION

Our objective is to create a tool able to cut meshes in a realistic way at run time.

For now, there are a lot of powerful mesh cutting algorithms, like Volumetric Approximate Convex Decompositions, but they are either not based on real images or not procedural.

We designed a solution capable of building a **profile** from a small data set of pictures and procedurally cutting 3D meshes at run time using these previously saved profiles.

That solution consist on a 3-parts tool including a Mesh Slicer, a Feature Extractor and a Profile Trainer. The **Mesh Slicer** generates a 2D graph from a profile, and can apply it to a 3D mesh to generate a fracture. The **Feature Extractor** use computer vision algorithms in order to extract usable data from an image. The **Profile Trainer** repetitively uses the two previous parts in order to build a profile corresponding to the desired result.

In order to test our solution, we've integrated it in Unity®software where the user can move in a 3D space and destroy virtual objects.

2 PROFILE

What we call "profile" is an array of data that describes which range of parameters for the Mesh Slicer allows the user to obtain a given result. A profile is built from two inputs: a small data set of fracture images, and a "maximum difference" defined by the user. The resulting profile, used by the Mesh Slicer, is giving fracture results that are similar to the input images.

We used that notion in order to allow the user to create different results depending on the material of the virtual object, and to allow him to use his own references to produce personalized results.

3 MESH SLICER

The Mesh Slicer is the only part that is used at run time. Its function is to create a graph (fig. 1c) and apply it on a mesh to subdivide it.

SIGGRAPH '18 Posters, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5817-0/18/08.
https://doi.org/10.1145/3230744.3230759

The graph is random but its shape is dependant of the parameters that are given to the Mesh Slicer (points count, distribution...). These parameters can be generated from a profile (fig. 2) to produce a specific fracture style.

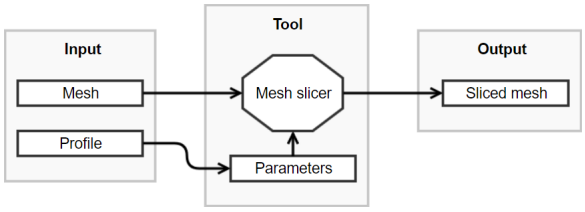

Figure 2: Run time phase: cutting a mesh using parameters stored in a profile

The graph creation is currently based on a Voronoï diagram which is used to partition a surface from a cloud of generated points. Then we project the diagram on the target mesh from an impact point and an impact direction and cut the mesh using the edges of the diagram (fig. 1d). Finally, we added the possibility of applying a force on all the resulting meshes to simulate the impact (fig. 4).

4 FEATURE EXTRACTOR

The Feature Extractor was necessary to extract pertinent data from images and therefore simplify the model of the Profile Trainer. It consist on an OpenCV application that applies an adaptive threshold in order to reveal a graph from the fracture, and then extracts the graph's points and edges that we feed the Profile Trainer with.

5 PROFILE TRAINER

The Profile Trainer uses a simple machine learning algorithm in order to find a profile corresponding to a set of fracture images. It begins by using the Feature Extractor to get graphs from input images, and creating a profile with random parameters. Then, it will repeatedly generate graphs with the Mesh Slicer from the profile's parameters and compare them with the input graphs. As long as the generated graphs differ from the input's, the profile's parameters are slightly adjusted until they allow the Mesh Slicer to generate a satisfying graph (fig. 3).

The profile is then saved and can be used at run time.

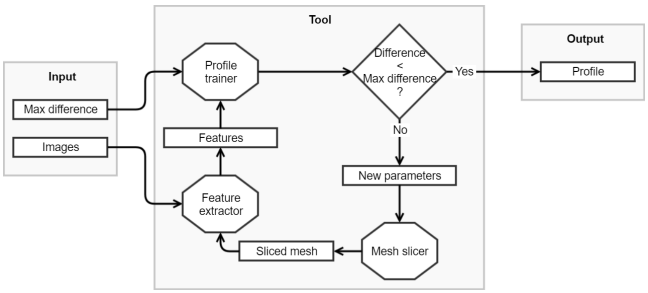

Figure 3: Training phase: building a profile from a small image data set

6 FUTURE WORK

This tool currently presents some limitations due to our algorithms choices: the choice of a Voronoï-based method prevents the possibility of creating concave cells and restricts the diversity of simulable materials. Our next objective is to enrich the Mesh Slicer by adding new graph styles and a larger amount of parameters, in order to enlarge the possible outputs and therefore be able to reproduce more types of material.

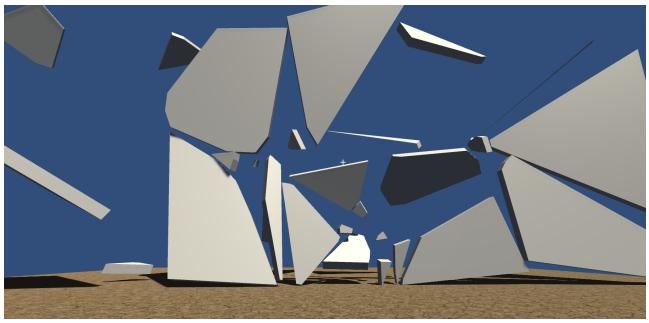

Figure 4: Unity3D integration of our solution

7 CONCLUSION

Currently to create a fracture model, the user have the choice between a fully procedural content or template assets. The first choice provides an infinite amount of results with a few data size, besides the result neither depend on the material nor is reality-inspired. The second one can provide a model created from a real fracture, but the resources size increase proportionally to the amount of different templates.

Despite its current limitations, this tool could provide a better balance between realism and resources size by providing large amount of reality-inspired templates within a smaller file format.

ACKNOWLEDGMENTS

We would like to thank Marc Bianchini and Nicolas Vidal for their support in this work.

REFERENCES

M. B. Dillencourt. 1990. Realizability of Delaunay Triangulations. *Inf. Process. Lett.* 33, 6 (Feb. 1990), 283–287. https://doi.org/10.1016/0020-0190(90)90210-O

Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2013. Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Trans. Graph.* 32 (2013), 115:1–115:10.

Ruohui Wang. 2016. *Edge Detection Using Convolutional Neural Network.* Springer International Publishing.