3D Model Cutting via Evolutionary Algorithms with Blender

Michele Tessari and Alessandro Zuech

***Abstract*—In order to automate some industrial processes that revolve around the carving of material to obtain a certain shape, the objective of this project is to build an autonomous program able to find the best combination of cuts that can slice an initial block of material into the target model.**

Different Evolutionary Algorithms were used to improve the optimization and find solutions with several cuts.

***Index Terms*—Genetic algorithm, AI, precision, sculpting.**

I. Introduction

CARVING is both a creative and a creation process, and has many applications in art and industry fields. Both fields use different tools in order to cut, carve or refine an initial block of material to obtain a final object, but most of the times these tools can't work on the whole initial block because they would wear out too much or they would be too slow. Indeed, one of the initial phases before a block gets carved, is the removal of a large amount of material using more powerful tools like:

- Chainsaws or circular saws for wood cutting
- Percussion hammers or diamond wire machines for very large marble cutting
- Laser, water jet, plasma cutting, electric discharge for cutting metals or other materials

All this is done to get a reasonable shape to work with, to then use more accurate tools for high precision milling or some other types of refinement. Usually the decision of the cuts is done manually by a human but there are some applications where making this process automated is necessary. Another reason to automate this operation is to minimize the number of cuts in order to save time, energy and extend the life of the cutting machinery.

We tried to solve this problem using different evolutionary algorithms without making any assumption on the 3D models and using 2 definitions of cut plus a third non-realistic one, because the use of EA makes possible also to use non standard cut definitions, and so to experiment artistic results.

II. Problem Definition

We have two 3D models defined as a mesh (list of vertices, edges and faces):

- *Target Mesh*: the reference model to approximate
- *Carving Mesh*: the initial model to cut in order to get an approximation of the target mesh (usually a cylinder)

M. Tessari and A. Zuech are with the DISI, University of Trento

The problem is defined as a constrained maximization problem, and the fitness to be maximized is the quantity of material removed from the carving mesh by one single cut minus a penalty factor in order to avoid to cut the target mesh. More in details the fitness function is defined as:

$$f(\mathbf{x}) = (V_{M_0^{carve}} - V_{cut(M_i^{carve},\mathbf{x})}) - g(M^{target}, \mathbf{x}).$$

Where:

- $\mathbf{x}$ is a cut representation
- $V_{M_i^{carve}}$ is the volume of the mesh to carve after i cuts.
- $V_{cut(M_i^{carve},\mathbf{x})}$ is the volume of M_i^{carve} cut with x
- $g(M^{target}, \mathbf{x})$ is the penalty function which is $V_{M_0^{carve}}$ if M^{target} intersects with $\mathbf{x}$ otherwise is 0

A. Cut Definition

We defined three problems distinguished by three types of cuts: planar, concave and spherical cuts. A planar cut is defined as an infinite plane cutting the mesh in 2 parts, a concave cut is defined as 2 planes that intersect each other where one of the four space sections, generated by the planes, is removed. The last is a spherical cut in order to explore a non-realistic setup.

The planar one is an easier problem to solve but it is limited and it can be bad for very concave models. The concave one can be applied to a larger set of situations and allows a potentially more precise approximation but with a significant increasing of the complexity. However, even if the concave cut can generate concave models, it cannot handle holes.

A set of cuts is then our population of individuals that are fed to the evolutionary algorithms, and represented by a sequence of floats. More in details a planar cut is defined as:

$$\mathbf{x}_{plane} = [o_x, o_y, o_z, n_x, n_y, n_z].$$

Where $\mathbf{o}$ is the position and $\mathbf{n}$ the normal vector of the plane in 3D coordinates. The concave cut is defined in the same way but adding an α parameter representing the angle between the intersection of the 2 planes:

$$\mathbf{x}_{wedge} = [o_x, o_y, o_z, n_x, n_y, n_z, \alpha].$$

Finally the sphere cut has only the origin and radius:

$$\mathbf{x}_{sphere} = [o_x, o_y, o_z, r].$$

The optimization is done one cut at the time where the best cut is selected from the final population after running the evolutionary computation and reaching the maximum number of generations. Finally, the carving mesh is sliced using the selected cut and then a new one is searched.

III. Methodology

A. Blender

Blender has been used as tool in order to handle 3D model files and compute cuts, volumes and intersections. It has been chosen because it can be used both as a python library without a view and as a debug/visualization tool. We choose to represent a cut as a boolean operator: a 3D operation between 2 meshes where the result is the difference, intersection or union of the two models where one model is the carving mesh and the other the representation of the cut. This representation makes the definition of the cuts more flexible, but it brings some problems related to speed performance and one bug (model disappearing with fast boolean).

B. Planar cut definition in Blender

The planar cut problem is the easiest between the two, and consequently has also been the first to be developed into a working program. The mesh representing the cut is defined as an half cube: the origin of the plane became the center of one of the faces of the cube, which is rotated in order to match the normal vector of the plane (e.g. Fig.1). The resulting cube is then used as element for the boolean operator.

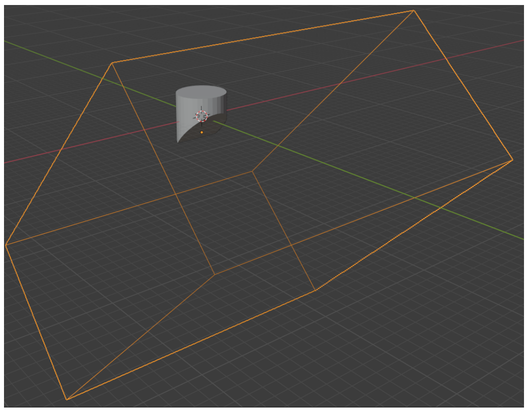

Fig. 1: Definition of the plane cut as an half cube, the origin is the orange point, the normal is $(0, 1, 1)$ (red line is X, green line is Y)

C. Wedge cut definition in Blender

Wedge cuts represent a more complex type of cut with respect to the planar type. In Blender, the cut is defined as a wedge used as element of the boolean operator, with the tip being considered as the intersection of the 2 planes. The origin is the center of the tip and the normal is the direction the wedge is pointing from the origin. The angle defines how large the tip is: if $\alpha = 180$, the tip is flat and the wedge behaves as a plane; if another angle is used instead, the wedge will behave as an intersection of planes (Fig.2).

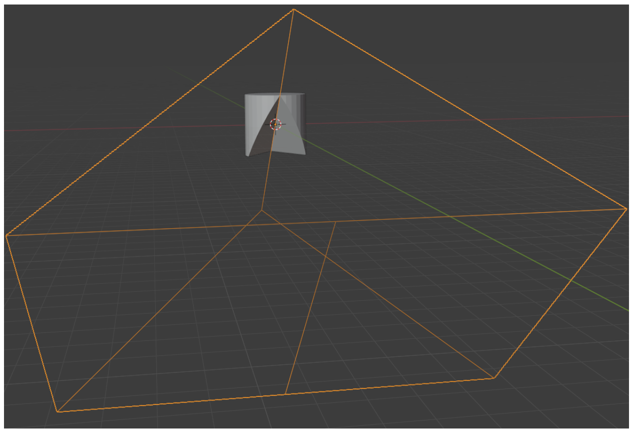

Fig. 2: Definition of the concave cut as a wedge, the origin is the orange point, the normal is $(0, 1, 1)$ (red line is X, green line is Y) and $\alpha = 90°$

D. Initial population generation

Evolutionary computation algorithms need mainly 2 functions: the generator that creates the initial population of the individuals and the evaluator providing the fitness of each individual.

The generator is not simply a random initialization of the planes (or wedges), but a random choice of an element from a list of planes (or wedges) oriented towards the center of the target mesh with the origin corresponding to a point belonging to the target mesh (the wedges will start with an angle of $180°$). In this way we get an already good initial exploration of the solution space with also initial good individuals and we use the algorithm to do mainly exploitation.

E. Evaluation of the cuts

The evaluator iterates on all the candidates and for each one of them it simulates a cut and computes the resulting volume that will be used to compute the fitness. The penalty checks if the simulated cut would intersects with the target mesh and in a positive case it subtract the fitness with a fixed value (static penalty) equal to the volume of the carving mesh without the new cuts applied.

Different functions to check intersections have been used in order to get better speed performances.

F. EC Algorithms

We tested the problems with 3 algorithms:

- **Genetic Algorithm:** A custom version of the classic genetic algorithm using generational replacement with weak elitism, tournament selection, classic crossover and gaussian mutation (except for the wedge and sphere cut problem where it has been tested also with a custom gaussian mutation)
- **Evolutionary Strategies:** Classic approach tested with static mutation strategy, global sigma, individual sigma and correlated multiple sigmas.
- **Particle Swarm Optimization:** Only tested with the default velocity formula but we didn't explore it in depth due to the bad results

G. Experimental Choices

For the plane cut problem a diamond was used as target mesh and a cylinder as carving mesh. The diamond has been chosen because it can be easily approximated with few cuts, and so good shapes can be seen instantly. Also, the diamond is a convex shape and so the problem can potentially carve the mesh perfectly (given enough cuts).

For the wedge cut problem we instead used a famous character model (Bulbasaur from Pokémon) because it is a convex model and good results should only be obtainable by cuts that are not possible with planar cuts or by removing more volume.

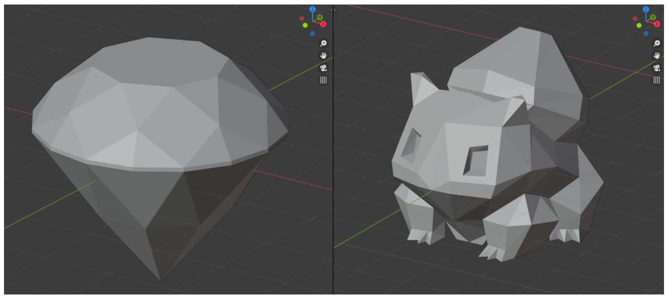
Fig. 3: Diamond mesh (left) and Bulbasaur mesh (right).

IV. Issues encountered

A. Speed performances

Blender modifiers are operators that affect an object's geometry or data in a non-destructive way. They can be added to an object and applied to make the changes permanent. They refer to the operations of difference, union or intersection between two meshes. In this project, the boolean modifiers were originally used for:

- creating the wedge.
- computing the volume for the fitness.
- applying the optimal cut.

It was found out that the presence of the modifiers, because they change the mesh and recalculate its properties (e.g. the normals), significantly slowed down the code and therefore three solutions were implemented to speed up the overall execution:

- substitute the ”EXACT” boolean modifier with its ”FAST” version (the precision depends on a user-defined threshold, but the speed boost is always more or less the same, even with low values of the threshold).
- calculate the volume without applying the modifier (before doing this we had to copy the carving mesh for every cut to evaluate).
- create the wedge shape without modifiers: The shape is created with a bottom-up approach defining its vertices, edges and faces from scratch instead of using a top-down approach (start from a geometry then use modifiers to get a wedge shape).

B. Boolean disappearing problem

Another issue that we called ”boolean disappearing problem” was discovered about ”FAST” boolean modifier: rarely, using the modifier in FAST mode, the carving mesh disappears completely when the cut is rotated by a certain angle. This is just a bug in Blender and it is less frequent with lower values of the threshold, but still relevant enough to compromise the optimization because the cut provides the maximum fitness possible (the disappeared mesh has zero volume) while bypassing every feasibility check. The solution implemented is to considered unfeasible the solutions that completely eliminate the carving mesh.

C. Shifting population problem

The last major issue that significantly compromised the optimization was the so-called ”shifting population problem”, which caused the population to move away from the target mesh as generations increased, reducing the exploration of the search space and compromising the quality of the final result. It was solved with the implementation of a population ”regeneration” strategy: new populations are randomly initialized after the best cut is applied to the carving mesh, instead of using the precedent population.

V. Results and discussion

We did many experiments with multiple runs (10) with different parameters using GA and ES but we did not investigated further for PSO because of his poor performances on these problems. Instead, GA and ES had good results in the plane cut problem, but on the wedge problem we could not get an improvement on the quantity of volume removed using the same carving mesh and the same concave target mesh ($\sim 10\%$ volume difference). We think that the main motivation of this is because good optima are hidden in very tight spaces that the algorithms can not reach.

A. Planar cut problem results

Best results have been reached with the custom version of genetic algorithms (Fig. 4) with the following parameters:

std dev:	0.1
mutation rate:	1
crossover rate:	0.5 (low influence)
elitism:	> 0
tournament size:	> 2

GA has seen to be constantly better than ES:

In Fig. 5 we can see that with a relatively small number of cuts (10) we obtained a good approximation of the diamond and the resulting mesh it reminds the target shape.

B. Wedge cut problem results

Besides the poorer performances the problem is also slower than the planar cut problem due to the more computation related to the check of the intersection between the target mesh and the wedge. To improve the fitness performances we tried

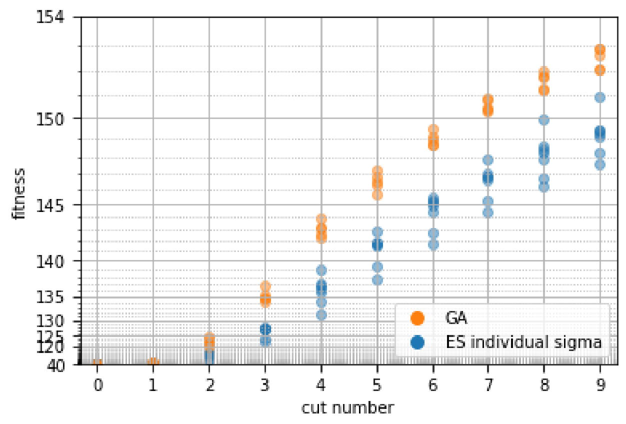

Fig. 4: Comparison between best GA and best ES (log scale)

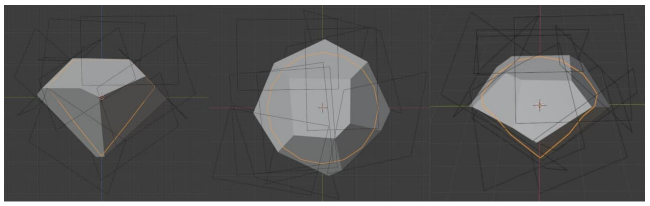

Fig. 5: resulting mesh after running GA with 10 planar cuts with the diamond as target mesh

to run the algorithm with a custom gaussian mutation that has 2 main differences with respect to the classical one:

- variable specific mutation rate: the mutation rate is no longer a single parameter but a vector of 7 mutation rates, one for every variable for the wedge cut representation
- variable specific standard deviation: the same thing is done for the σ parameter of the gaussian.

This technique brings a small improvement on the fitness, but still not enough to win against the plane cut problem.

Evolutionary Strategies do not bring improvements ($\sim$ 8-10% worse than GA) also testing with different types of the algorithm (static, global, individual, multi correlated).

C. Regeneration

The next figure illustrates the implementation of the "regeneration" strategy.

As the graph Fig. 8 shows, the added "regeneration" in the evolution enhanced both the convergence speed and the final result. The main reason of this improvement is that regeneration solves the shifting population problem. When the best individual was finally applied, the old population were too biased to explore well the new search space. Instead, with this new strategy, each cut became independent from its predecessor.

VI. Conclusion

The execution of the problem has been observed to be dependent on the target mesh we have to approximate: concave models can not be well approximated with planar cuts and is

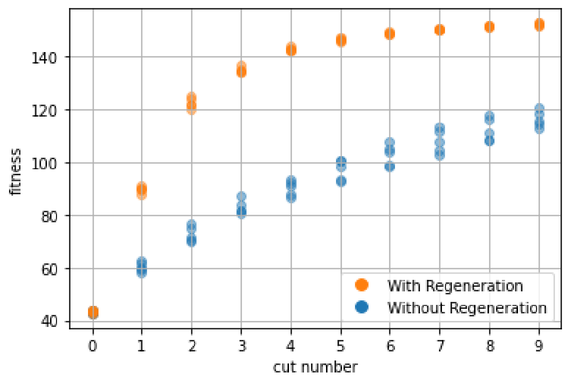

Fig. 6: Effects of implementation of regeneration strategy

very difficult to optimize using wedge cuts. On the other hand, convex meshes are very easily optimized with planar cuts.

The wedge problem is the one with the most potential because of its intrinsic ability to carve concave objects, but it is also the most complex to optimize. We did not obtain good results, but a more explorative algorithm with different mutation techniques (like mutating only the angles or the normals) or using an heuristic approach (for example forcing the origins to lie down to the surface of the target mesh) could be a possible solution to improve performances.

Even though an optimal deterministic algorithm already exists, our algorithm has been tested on the planar problem in order to get a proof-of-work. On the hand, the wedge problem does not have a deterministic solution yet and if the right EA technique is found the wedge problem will have a practical use. A potential extension of the project can be possible using different fitness definitions (entropy of the carving mesh vertices or alignment of the face normals) in order to experiment new artistic techniques.

VII. Contributions

The python scripts and this report were both written by Michele Tessari and Alessandro Zuech. More in details Michele Tessari did: main execution scripts, plane cut problem class, optimized version of the wedge cut problem class, majority of the experiments.
Alessandro Zuech: initial version of wedge cut problem class, part of the experiments.
Both equally: researches and tests to find the right tool to work with 3D models in python (very time consuming and in the end Blender was chosen).

Appendix A
Other Results

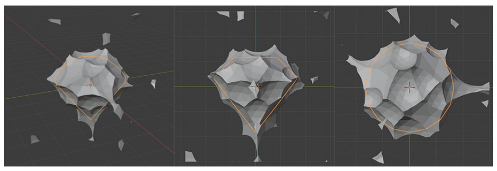

Fig. 7: Results using a sphere cut (40 cuts, GA with custom gaussian mutation, diamond as target)

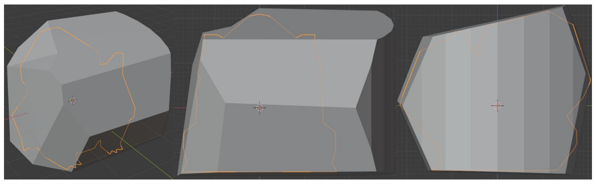

Fig. 8: Results using a wedge cut (10 cuts, GA with custom gaussian mutation, bulbasaur as target)

References

[1] Blender Python API Documentation. [Online] Available: https://docs.blender.org/api/current/index.html

[2] Inspyred Documentation. [Online] Available: https://pythonhosted.org/inspyred/

[3] Sunday, Dan (2001), Inclusion of a Point in a Polygon