Fitting Parameters for Procedural Plant Generation

Albert Garifullin
Lomonosov Moscow State University
GSP-1, Leninskie Gory
119991, Moscow, Russia
albgar-14@yandex.ru

Alexandr Shcherbakov
Lomonosov Moscow State University
GSP-1, Leninskie Gory
119991, Moscow, Russia
alex.shcherbakov@graphics.cs.msu.ru

Frolov Vladimir
Lomonosov Moscow State University
Keldysh Institute of Applied Mathematics
GSP-1, Leninskie Gory
119991, Moscow, Russia
vfrolov@graphics.cs.msu.ru

ABSTRACT

We propose a novel method to obtain a 3D model of a tree based on a single input image by fitting parameters for some procedural plant generator. Unlike other methods, our approach can work with any plant generator, treating it as a black-box function. It is also possible to specify the desired characteristics of the plant, such as the geometric complexity of the model or its size. We propose a similarity function between the given image and generated model, that better catches the significant differences between tree shapes. To find the appropriate parameter set, we use a specific variant of a genetic algorithm designed for this purpose to maximize similarity function. This approach can greatly simplify the artist's work. We demonstrate the results of our algorithm with several procedural generators, from a very simple to a fairly advanced one.

Keywords
3D modeling, plants modeling, tree reconstruction, genetic algorithms

1. INTRODUCTION

Trees and other plants play a key role in shaping the landscapes around us, and therefore a realistic representation of vegetation is one of the important tasks of computer graphics. The use of tree models is necessary for many industries - from computer games and virtual reality systems to architecture and urban planning. Nowadays diversity and realism in the visualization of vegetation are needed.

It is possible to solve this problem with the help of 3D reconstruction, creating a model of a tree that exists in the real world. However, this requires a detailed representation of the tree structure like laser scanning data with a high level of detail, but such data is difficult and expensive to obtain. The reconstruction of the tree from the images probably won't be accurate, as many branches are hidden by the crown or strongly intertwined with each other. An alternative to reconstructing trees can be their creation using procedural generation. The main problem with this approach is that the tree is created from a set of some input parameters and their correct selection requires a lot of time from the artist. Various parameters significantly affect each other and it is often not obvious to the user how changing some value of the input parameter affects the final model.

We propose an algorithm that combines both of these approaches and performs image-based modeling of plants. It takes a single image and a procedural generator and finds such a set of parameters of this generator that results in a tree most similar to the image. The appropriate set of parameters is found as the maximum point of the “similarity” function of the model and the original image. To optimize this function, a special version of the genetic algorithm was implemented. The main advantage of our solution is its independence from the generator, as the algorithm treats it like a black box, while existing solutions rely heavily on their own generators, inevitably limited in their capabilities. Also, our approach allows users to specify some other desired properties of the model, such as its geometric complexity, which is useful for practical application.

2. RELATED WORK

Trees Modeling

The problem of vegetation modeling has been an area of scientific interest for many years. At the moment, all approaches to plant modeling can be divided into 3 groups - interactive, procedural, and reconstructive [Sme14]. Interactive methods are the most widely used in the industry. Projects like SpeedTree [St17] or Xfrog [Xf17] provide an artist with a tool that allows him to interactively create highly detailed tree models, but their use requires a certain level of skill, and creating a model takes a lot of time. Procedural generation methods can create a model of a plant without human involvement, relying only on a certain set of input parameters and, possibly, a description of the environment. There are many different methods of procedural modeling of plants. Early works used sets of rules to describe the structure of a tree [Hon71][Web95], L-systems [Pru86][Pru12], cellular automata [Gre89], and particle systems [Ree85]. All of them somehow come down to the recursive construction of a plant model, without taking into account the environment. Newer works concentrate on simulating the growth process and the influence of the environment on it [Lon12][Had17][Yi15][Yi18]. There are, although less commonly used in practice, methods for reconstructing tree models from multiple images [Neu07][Ch08], video [Li11], and laser-scanned 3D point clouds [Liv10][Du19], including using neural networks [Liu21].

Image-Based Modeling

The closest to our work are combined methods involving the use of a procedural generator for model creation. So in [St14], a 3D model of the tree is used, according to which the generator parameters are selected. The work [Tan08] focuses on modeling a tree based on a single image, in which the user needs to manually select the main branches and crown. According to this image, a 3D model of the tree is assembled from a given set of branches. The work [Li21] also uses only one image for creating a model, but fully automates this process by using three neural networks: the first is used to segment the image, the second to form an approximate representation of the tree in the form of radial bounding volumes, and the third to determine the type of plant. A plant species is defined as a set of parameters for a specific procedural generator, also described in the paper.

All these works are based on the use of their own procedural methods of plant modeling. This imposes serious restrictions on the application of their results since only those trees that can be described by the procedural model used in the work can be created. In addition, existing works do not allow the user to control how complex and detailed a 3D model will turn out, although this is necessary in many cases. Our method implements the same image-based approach but does not have these restrictions because it can work with different procedural generators.

Differentiable Rendering

In addition to specific reconstruction methods for trees, there are more general approaches. Currently, approaches based on differentiable rendering are popular for solving problems of accurate reconstruction [Has21][Mun21][Tak22][Hu22]. In these approaches, differentiable rendering can be used to propagate the error backward from the triangular mesh to the original model. So in [Mun21] the error spread from the mesh to the tetrahedron grid to optimize the topology, and in [Hu22] to the parameters of the procedural texture generator. Although this approach looks promising, there are several difficulties in applying it to the procedural generation of vegetation. Firstly, the procedural model must be differentiable, which is generally not satisfied. Secondly, small changes in the parameters of procedural generators do not always lead to small changes in the resulting 3D model. This means that gradient optimization methods will not have the expected efficiency that can be observed in well-known applications of differentiable rendering.

3. OVERVIEW

Our goal is to obtain a 3D model of a tree from its 2D image or sketch using some given procedural generator. In this case, the generator is considered as a black box, at the input of which is a certain set of numerical parameters, and at the output is a plant model represented by a graph of branches. In addition to the image, the user can specify their requirements for some properties of the resulting model that cannot be obtained directly from the image. For example, limit the number of nodes in the branch graph, i.e. the geometric complexity of the resulting model.

To solve this problem, a function is proposed that evaluates the degree of similarity of the model made by the generator with the original image, taking into account restrictions. Then the task of creating a tree similar to the image is reduced to finding the global maximum of this function on the entire set of acceptable parameters. It is important to note that, although the function has values from 0 to 1, its maximum value is unknown, because it is not guaranteed that the procedural generator can create a tree exactly of the required type. Moreover, it is obvious that this problem has not the only solution, since there may be different 3D models that are equally similar to the input image. To simplify working with the function, we transform each parameter's value into [0, 1] interval so the function is defined on the set $[0, 1]^n$, where n - is a number of generator's parameters.

To solve this optimization problem, a special version of the genetic algorithm was developed. Like any genetic algorithm, it does not guarantee the achievement of a global maximum, but on average it can find points good enough in terms of the function value. In its work, the genetic algorithm uses statistical information about the entire family of functions corresponding to different input images, but the same procedural generator.

The results of the algorithm with three different procedural generators and several different images are also demonstrated.

4. SIMILARITY FUNCTION

The value of similarity function is a multiplication of image similarity value and characteristic multipliers.
Image similarity value is obtained by comparings the source image with the impostors of the generated tree model. Semantic masks are used for comparison, where each pixel belongs to one of three categories: branch, foliage, background. To obtain such a mask from the source image, a neural network can be used similarly to [Li21], and in simple cases, we can get it based only on pixel color (green corresponds to leaves, brown or gray - to branches and trunk).

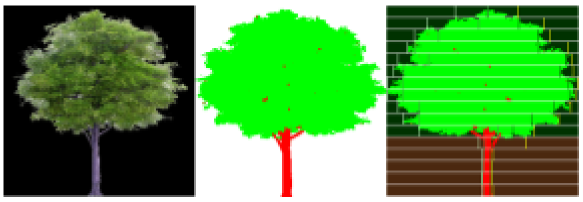
Figure 1. The original image, semantic mask, visualization of the division into stripes. Stripes of brown color correspond to the trunk, green color to the crown. Vertical lines inside each stripe - values ai, bi, ci, di respectively.

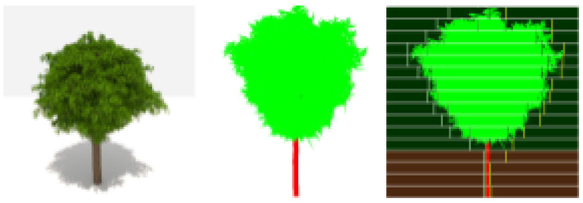
Figure 2. Generated tree, semantic mask (imposter), visualization of the division into stripes.

The image is divided into 20-30 narrow horizontal stripes, for each of which are determined:

- $[a_i, d_i]$ - crown borders
- $[b_i, c_i]$ - dense crown borders (>75% leaves pixels)
- B_i - branches pixels percentage
- L_i - leaves pixels percentage

According to the ratio B_i/L_i, each stripe refers either to the crown or to the trunk, see Figs. 1 and 2.
Comparing the parameters a_i, b_i, c_i, d_i and B_i/L_i ratio for every stripe of the original image and the image of the generated model, we calculate the value of image similarity $ImSim$.
Characteristic multiplier shows the difference in the characteristics of the model and given one. For characteristic C, the multiplier has the following formula:

$$C_{mul} = min(C_{model}, C_{reference}) / max(C_{model}, C_{reference})$$

The final function value:

$$Sim = ImSim * \prod_{c \in \mathbb{C}} C_{mul}$$

$\mathbb{C}$ - the set of all given characteristics. Among them may be:

- Number of vertices in the branch graph
- Height and width in some scale
- Average branches and leaves density
- Average leaf size

None of these characteristics is mandatory, but it is recommended to specify the number of vertices in the branch graph, otherwise the search for a solution will slow down due to the need to search for it among models with very high geometric complexity.

5. GENETIC ALGORITHM IMPLEMENTATION

The previously mentioned similarity function is used as an objective function for the genetic algorithm.
A proposed genetic algorithm consists of several elementary genetic algorithms, with a selection of the best results of each of them. Each elementary GA includes the initialization of a population and its evolution over a fixed number of generations. In the figure, each vertex of the tree is such an elementary GA. Algorithms on the leaves of the tree start with a randomly initialized population, and all the others form a population of the fittest “individuals” obtained in the child vertices.
All elementary GA work according to the same strategy ($f(x)$ - objective function)

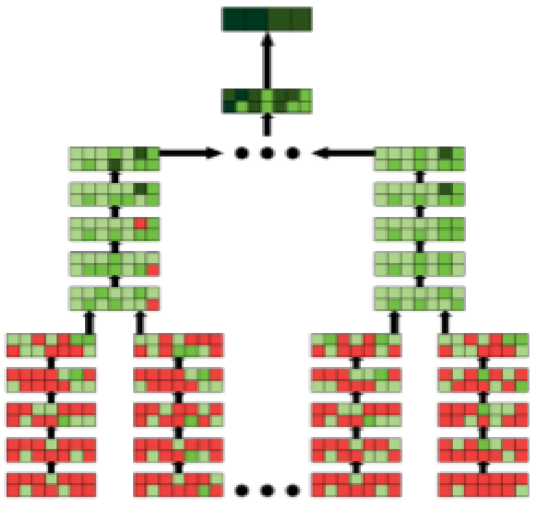

Figure 3. Tree-like population structure. Zero-level elementary GA are started with random genes, while GA from level i takes best species from two final populations of the previous level. The final result of a whole algorithm is a small set of best species on the very top level.

Selection

At the beginning of each iteration, half of the population with the worst fitness value is removed. The remaining individuals take part in the creation of a new generation. At each of the vacant places in the population, a new individual is created with one-dot crossover, its parents are selected from the remaining species of the previous generation. The fitness proportionate selection is used, which means that the probability of choosing an individual as a parent is proportional to the value of its fitness. For representatives of the new generation, the values of the objective function and the fitness function are calculated.

Mutation

Mutation chance M_{chance} and percentage of genes to change M_{genes} are constant.

Here is a proposed method for genome G mutation:

1) values of $n * M_{genes}$ randomly chosen genes are changed. The probability of mutation in the k gene is proportional to the *average rate of change* of this gene $V(k)$
2) For the result genome G' we estimate its *quality* $Q(G')$
3) Steps 1-2 are repeated several times (500 in the experiment) and the mutation results in a gene with a better quality score

Functions $V(k)$ and $Q(G')$ based on pre-collected information about the entire family of objective functions F = $\{f(x)\}$ (each function corresponds to its own input image and set of properties)

Gene value

$$V(k) = Average\left(\frac{f(\bar{x} + h*x_k) - f(\bar{x})}{h}\right)$$

Genome quality

$$P(i,j) = P(f(\bar{x}) > eps \,|\, (j-1)/k < x_i < j/k),$$
$$i \in \{1, ..., n\},\ j \in \{1, ..., k\}$$
$$P_0 = P(f(\bar{x}) > eps)$$
$$Q(G') = \sum_{i=1}^{n} Q(i, \lceil G'_i * k \rceil)$$
$$Q(i,j) = sgn(P(i,j) - P_0) *$$
$$max(P(i,j), P_0)/min(P(i,j), P_0)$$

Several hundred thousand calculations of functions were carried out for a fairly accurate assessment of $f(\bar{x})$ values for several dozen different images with each of the used procedural generators.

6. RESULTS

We implemented three different procedural generators to demonstrate the results of our method:

1) WeberPennGen - implementation of the algorithm described in [Web95]
2) GEGen - a generator simulating the process of tree growth, taking into account the environment, based on [Yi15][Yi18]
3) SimpleGen - a generator with a simple set of rules for the recursive description of the tree structure, used for testing purposes during development

In the experiments, images of trees were used as input data, the only additional requirement was the geometric complexity of the model. The result of the algorithm was a group of several best candidates, the selection of which was performed manually. To get the result, 40-60 thousand calls of the procedural generator were required, 10-50 minutes for calculation on a PC with AMD Ryzen 7 3700X, 16 GB RAM, Nvidia GeForce RTX 3070. Figures 4-8 show the ability of our algorithm to deal with different tree species. You could notice that the model takes from the image not only the approximate shape of the crown but also the structure of its edge. Images in Fig. 6 and 9 both have cone-shaped crowns, but the thuja in Fig. 6 has smooth edges, close to a real cone and the spruce tree in Fig. 9 has distinct branches with visible gaps between them. The density of the crown is also preserved, as you can see in Fig. 5 (both reference image and model have dense crows) and Fig. 8, where there are trees with a lot of gaps between branches. Fig. 7 shows the ability of our algorithm to use a very simple sketch as an input.

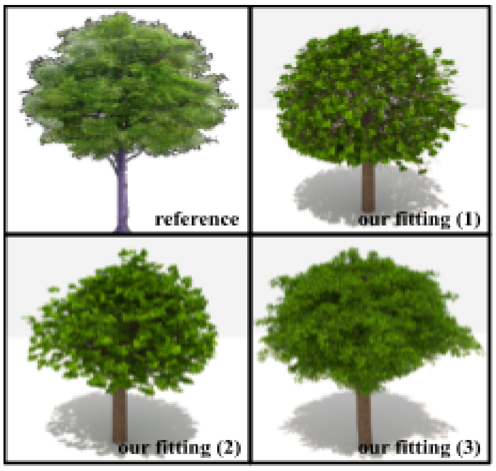

Figure 4. Input image and 3 best candidates created with WeberPennGen

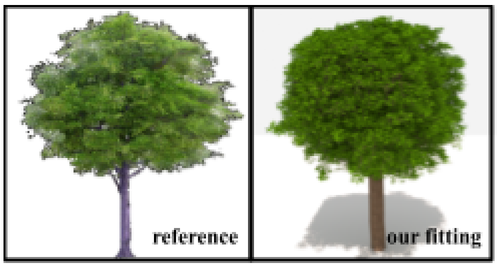

Figure 5. The same reference image as in Fig. 4, but GEGen generator is used

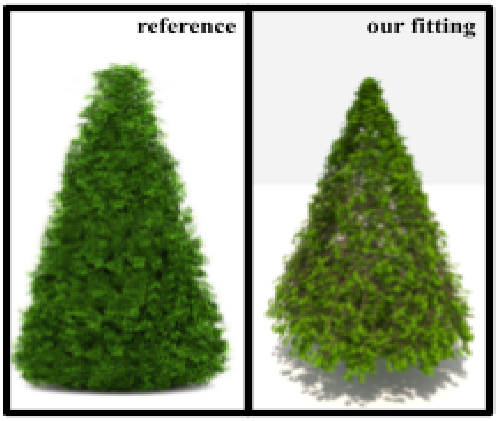

Figure 6. Decorative thuja tree

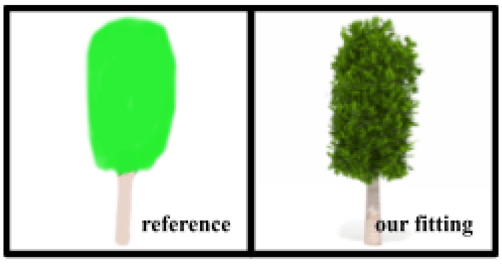

Figure 7. Tree based on simple sketch

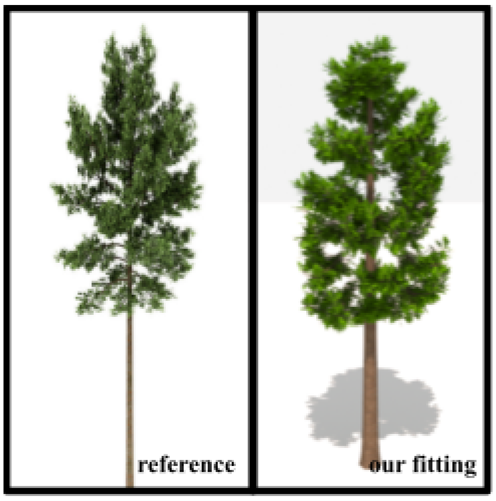

Figure 8. Small pine tree

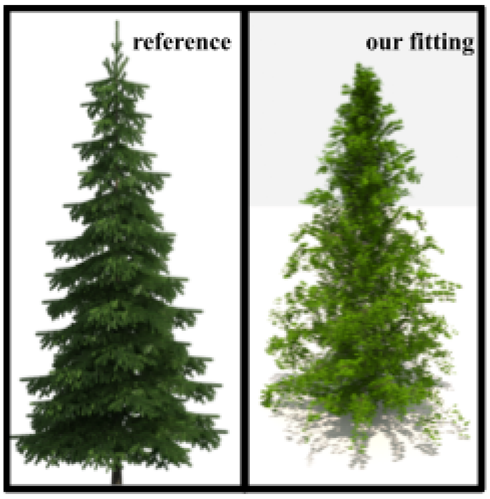

Figure 9. Spruce tree

7. CONCLUSION

A novel method of obtaining a 3D model of a tree from a single image was proposed. It is mostly autonomous: a user is needed at the very end to choose one of the created models. Unlike previous works, our method can work with an arbitrary procedural plant generator, which makes it easy to use modern solutions in this area without changing the algorithm itself. This, as well as the ability to specify the required complexity of the model, makes it more applicable for computer graphics applications.

The framework implemented in the process of working on the paper demonstrates the abilities of the method, but it can be improved in many different aspects. We consider the most promising refinement of the similarity function, as well as an additional assessment of the realism of the model in addition to its comparison with the sample. It is also promising to create a specialized tool for 3D modeling of plants using our algorithm or integrate it into existing ones.

8. ACKNOWLEDGMENTS

The work was sponsored by the non-profit Foundation for the Development of Science and Education "Intellect".

9. REFERENCES

[Sme14] Smelik, Ruben M., Tim Tutenel, Rafael Bidarra, and Bedrich Benes. "A survey on procedural modelling for virtual worlds." In Computer Graphics Forum, vol. 33, no. 6, pp. 31-50. 2014.

[St17] SpeedTree IDV Inc, 2017. URL: http://www.speedtree.com

[Xf17] Greenworks Organic Software. Xfrog procedural organic 3D modeler, 2017. URL: http://xfrog.com

[Pru86] Prusinkiewicz, Przemyslaw. "Graphical applications of L-systems." In Proceedings of graphics interface, vol. 86, no. 86, pp. 247-253. 1986.

[Yi15] Yi, Lei, Hongjun Li, Jianwei Guo, Oliver Deussen, and Xiaopeng Zhang. "Light-Guided Tree Modeling of Diverse Biomorphs." In PG (Short Papers), pp. 53-57. 2015.

[Yi18] Yi, Lei, Hongjun Li, Jianwei Guo, Oliver Deussen, and Xiaopeng Zhang. "Tree growth modelling constrained by growth equations." In Computer Graphics Forum, vol. 37, no. 1, pp. 239-253. 2018.

[St14] Stava, Ondrej, Sören Pirk, Julian Kratt, Baoquan Chen, Radomír Měch, Oliver Deussen, and Bedrich Benes. "Inverse procedural modelling of trees." In Computer Graphics Forum, vol. 33, no. 6, pp. 118-131. 2014.

[Li21] Li, Bosheng, Jacek Kałużny, Jonathan Klein, Dominik L. Michels, Wojtek Pałubicki, Bedrich Benes, and Sören Pirk. "Learning to reconstruct botanical trees from single images." ACM Transactions on Graphics (TOG) 40, no. 6 (2021): 1-15.

[Pru12] Prusinkiewicz, Przemyslaw, and Aristid Lindenmayer. The algorithmic beauty of plants. Springer Science & Business Media, 2012.

[Hon71] Honda, Hisao. "Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body." Journal of theoretical biology 31, no. 2 (1971): 331-338.

[Web95] Weber, Jason, and Joseph Penn. "Creation and rendering of realistic trees." In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 119-128. 1995.

[Gre89] Greene, Ned. "Voxel space automata: Modeling with stochastic growth processes in voxel space." In Proceedings of the 16th annual conference on Computer graphics and interactive techniques, pp. 175-184. 1989.

[Ree85] Reeves, William T., and Ricki Blau. "Approximate and probabilistic algorithms for shading and rendering structured particle systems." ACM siggraph computer graphics 19, no. 3 (1985): 313-322.

[Lon12] Longay, Steven, Adam Runions, Frédéric Boudon, and Przemyslaw Prusinkiewicz. "TreeSketch: Interactive Procedural Modeling of Trees on a Tablet." In SBIM@ Expressive, pp. 107-120. 2012.

[Had17] Hädrich, Torsten, Bedrich Benes, Oliver Deussen, and Sören Pirk. "Interactive modeling and authoring of climbing plants." In Computer Graphics Forum, vol. 36, no. 2, pp. 49-61. 2017.

[Neu07] Neubert, Boris, Thomas Franken, and Oliver Deussen. "Approximate image-based tree-modeling using particle flows." In ACM SIGGRAPH 2007 papers, pp. 88-es. 2007.

[Li11] Li, Chuan, Oliver Deussen, Yi-Zhe Song, Phil Willis, and Peter Hall. "Modeling and generating moving trees from video." ACM Transactions on Graphics (TOG) 30, no. 6 (2011): 1-12.

[Liv10] Livny, Yotam, Feilong Yan, Matt Olson, Baoquan Chen, Hao Zhang, and Jihad El-Sana. "Automatic reconstruction of tree skeletal structures from point clouds." In ACM SIGGRAPH Asia 2010 papers, pp. 1-8. 2010.

[Du19] Du, Shenglan, Roderik Lindenbergh, Hugo Ledoux, Jantien Stoter, and Liangliang Nan. "AdTree: accurate, detailed, and automatic modelling of laser-scanned trees." Remote Sensing 11, no. 18 (2019): 2074.

[Liu21] Liu, Yanchao, Jianwei Guo, Bedrich Benes, Oliver Deussen, Xiaopeng Zhang, and Hui Huang. "TreePartNet: neural decomposition of point clouds for 3D tree reconstruction." ACM Transactions on Graphics 40, no. 6 (2021).

[Tan08] Tan, Ping, Tian Fang, Jianxiong Xiao, Peng Zhao, and Long Quan. "Single image tree modeling." ACM Transactions on Graphics (TOG) 27, no. 5 (2008): 1-7.

[Ch08] Chen, Xuejin, Boris Neubert, Ying-Qing Xu, Oliver Deussen, and Sing Bing Kang. "Sketch-based tree modeling using markov random field." In ACM SIGGRAPH Asia 2008 papers, pp. 1-9. 2008.

[Has21] Hasselgren, Jon, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. "Appearance-Driven Automatic 3D Model Simplification." arXiv preprint arXiv:2104.03989 (2021).

[Mun21] Munkberg, Jacob, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. "Extracting Triangular 3D Models, Materials, and Lighting From Images." arXiv preprint arXiv:2111.12503 (2021).

[Tak22] Takimoto, Yusuke, Hiroyuki Sato, Hikari Takehara, Keishiro Uragaki, Takehiro Tawara, Xiao Liang, Kentaro Oku, Wataru Kishimoto, and Bo Zheng. "Dressi: A Hardware-Agnostic Differentiable Renderer with Reactive Shader Packing and Soft Rasterization." arXiv preprint arXiv:2204.01386 (2022).

[Hu22] Hu, Yiwei, Chengan He, Valentin Deschaintre, Julie Dorsey, and Holly Rushmeier. "An Inverse Procedural Modeling Pipeline for SVBRDF Maps." ACM Transactions on Graphics (TOG) 41, no. 2 (2022): 1-17.