DOI: 10.1111/cgf.13432

Eurographics Conference on Visualization (EuroVis) 2018
J. Heer, H. Leitte, and T. Ropinski
(Guest Editors)

Volume 37 (2018), Number 3

Landscaper: A Modeling System for 3D Printing Scale Models of Landscapes

K. Allahverdi[1] H. Djavaherpour[1] A. Mahdavi-Amiri[2] F. Samavati[1]

[1]University of Calgary, Canada
[2]Simon Fraser University, Canada

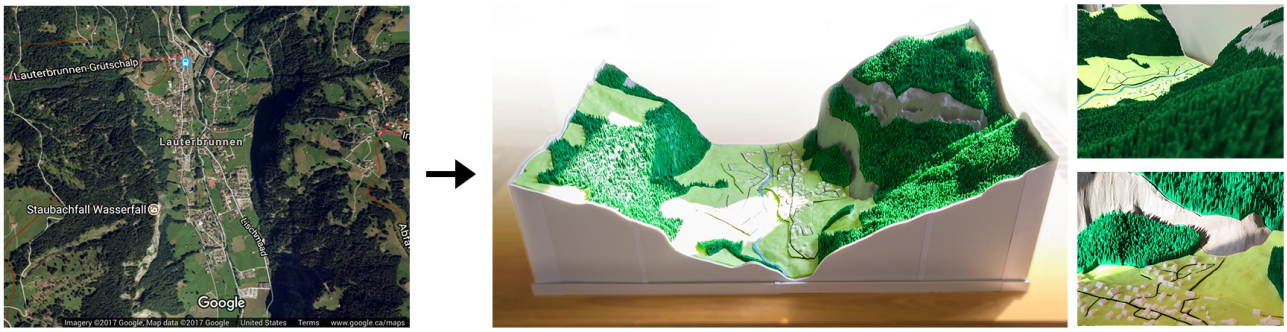


Figure 1: *Left: A satellite image from Google Maps. Right: Our 3D printed model of the same area.*

Abstract

Landscape models of geospatial regions provide an intuitive mechanism for exploring complex geospatial information. However, the methods currently used to create these scale models require a large amount of resources, which restricts the availability of these models to a limited number of popular public places, such as museums and airports. In this paper, we have proposed a system for creating these physical models using an affordable 3D printer in order to make the creation of these models more widely accessible. Our system retrieves GIS relevant to creating a physical model of a geospatial region and then addresses the two major limitations of affordable 3D printers, namely the limited number of materials and available printing volume. This is accomplished by separating features into distinct extruded layers and splitting large models into smaller pieces, allowing us to employ different methods for the visualization of different geospatial features, like vegetation and residential areas, in a 3D printing context. We confirm the functionality of our system by printing two large physical models of relatively complex landscape regions.

CCS Concepts

•***Computing methodologies*** → ***Computer graphics;***

1. Introduction

Visualizing geospatial features is a challenging task. One contributing factor to this challenge is the variety and size of geospatial data sets. Another lies in how to assign the data to an appropriate representation of the Earth for a better understanding of regions and their geospatial features. For example, how to visualize hiking trails in a national park or ski trails in mountains such that people can better imagine the place and its geospatial features. Physical visualization is beneficial as a 3D physical model enables tactile exploration and easy circumnavigation [HW17]. These attributes of 3D physical models (i.e. scale models) help to better explore complex GIS data, which makes this type of physical visualization useful for both educational and scientific purposes. Moreover, urban designers, urban planners and architects can benefit from these scale models when visualizing their 3D designs to non-experts [HW17].

Currently, popular public places such as national parks, airports and ski resorts benefit from these models as they help visitors develop a familiarity with the area quickly. Also, this type of physical visualization can be employed to raise awareness about environmental issues, such as global warming, by highlighting impacts to endangered regions of interest over time.

The most common method used to create these scale models is to use subtractive machining with a CNC (Computer Numerical Control) machine, where the required model is machined out of a sheet or volumetric material. Rapid Prototyping (RP) is another method that uses Additive Manufacturing (AM), where models are created by adding layers of materials. Stereolithography (SLA) and Fused Deposition Modeling (FDM) are two RP-based approaches that can operate on affordable desktop 3D printers. SLA uses an ultraviolet laser to solidify a rather costly liquid material, such as resin, while FDM process operates by melting and extruding a plastic material.

Every 3D printing process impose several difficulties on creating scale models. CNC is limited to use only one material for each session and requires costly machines. Methods using RP are considered generally inappropriate for building large landscape models due to the size limitation of machines that use this method [KGL01]. Specifically, FDM process suffers from a lower accuracy, problem with overhang parts, warpage and a limited number of materials to use for each session.

In this paper, we employ an affordable 3D printer which uses FDM process to design a system that makes creation of scale models accessible and reproducible. While SLA is a reasonable method to use with our system, since it is currently more expensive than FDM, we choose FDM due to their affordability and availability. Using our system we make 3D models that both respect the GIS data and the FDM process limitations. We propose methods to 3D print large physical models using such 3D printers, while providing methods for physically visualizing rather complex geospatial features such as vegetation without creation of overhangs and handling delicate pieces such as narrow road models. Furthermore, we consider the accuracy limitation of these 3D printers for both designing 3D models and their assembly.

Since geospatial areas are usually complex with large fine features (e.g. valleys, trails, roads and vegetation), creating a 3D model of these regions that is suitable for 3D printing is not an easy task. Although one may assume the use of highly detailed data sets (e.g. LiDAR), these kinds of data sets are only available for small regions, and capturing them requires a substantial amount of resources. Furthermore, utilizing the highly detailed information of a region of interest is neither necessary nor sufficient for 3D printing. For example, a perfect tree model has many delicate features and is not tailored for small scale physical models of landscapes and urban areas. On the other hand, vast numbers of 2D and 2.5D geographical data sets (i.e. digital elevation models (DEM), satellite images and GIS vector data) are freely available. These data sets have been captured for most areas of the globe, through various providers and governmental initiatives (e.g. OpenStreetMap, CDSM [CDS14]). Therefore, we aim at developing a 3D modeling system that creates appropriate 3D models (with respect to the 3D printing constraints) for regions of interest described by real geospatial data sets (see Figure 1). The inputs to our system are DEM and 2D GIS vector data. From these data sets our system generates a series of 3D meshes, each of which represents a unique geospatial feature from the input data. These feature meshes are further processed to create a collection of 3D elements, where each element is suitable for printing. After printing these 3D elements, they are assembled using a manual process to create the final physical model (Figure 1, middle image).

To create 3D feature meshes, we employ Constrained Delaunay Triangulation (CDT) to generate a 2D triangulation that respects all of the 2D input features. Using the input DEM, this 2D triangulation is turned into a 3D mesh. A region growing algorithm is applied to this 3D mesh to extract separate feature submeshes for each input feature (see Figure 2). This algorithm also computes hierarchical information for each face, which is used to filter and combine features (see Section 6.3.1). As we use affordable 3D printers, there is a printable size limitation. A grid optimization is used to break large feature submeshes into printable 3D elements.

Printing a realistic physical model is another challenge, as FDM process limits the range of physical models that can be created. A simple approach would be to use a different color for each feature. In this way, a lake printed in blue can be recognized in the physical model. However, a forest region and a grass field are hard to differentiate without additional information, as they will have similar green colors. To address this issue, while considering the printable models for FDM process, we create appropriate designs for different features to assist with feature recognition (e.g. our system creates a simulated displacement map in order to print massive vegetation areas).

To improve the manual assembly process, our system generates 3D models with a proper coding based on the feature number and grid location, resulting from the segmentation algorithm. Our system also avoids creating very small pieces that are harder to assemble, by using a cost function during segmentation. In regions whose features have complex interactions (e.g. residential areas), our system creates stencil layers that ease the attachment of features. To help fitting pieces, our system has a configurable offset amount to be applied to each piece based on the specific 3D printer inaccuracy. To further enhance this manual process, our system packs smaller pieces together to reduce the number of printing sessions required. We also help reduce printing time by creating pillars to support the surfaces of the physical models.

Our main contribution is to introduce and develop a system (see Figure 2) that, given a geographical region and its associated GIS data, creates 3D elements of a scale model that are printable by affordable 3D printers, e.g. FDM based printers. We have proposed methods to physically visualize geospatial features that are generally difficult to 3D print for affordable 3D printers. By employing displacement maps for vegetation regions and stencil layers for residential areas, our system speeds up the assembly process for massive amounts of models, while respecting the input geospatial data sets. By decomposing a landscape region to its features, we provide a way to 3D print a scale model using 3D printers with limited number of materials. Our system provides a local grid segmentation algorithm to create smaller pieces that fit inside the printable volume of 3D printers. This segmentation operates on every feature mesh locally, which makes the algorithm more flexible than a global grid

for the whole physical model. Furthermore, since the virtual model of a scale model is rather highly detailed and complex, separating features and segmenting them helps to create smaller 3D elements that are more manageable for the 3D printing software to process. Lastly, the 3D elements created by our system can be assembled easily after printing to create large landscapes at scales between 1:1000 and 1:2500.

2. Related Work

In this Section, we note the works most related to our system. Since our work combines geospatial data sets and 3D printing into an integrated system, we present works related to both of these subjects in the following.

2.1. Geospatial Data Sets

Geospatial data may refer to a variety of data sets, including elevation, satellite images, feature vectors that represent roads, rivers or political boundaries, in addition to 3D data sets that can be captured as point clouds to represent 3D models such as buildings, bridges, or statues [VTP15, MAAS15]. Such geospatial data sets are captured through different techniques (e.g. satellites, LiDAR, surveying, or drones) [CW11]. Much effort have been devoted to clean-up, post process, and visualize these data sets [MAAS15, MWA*13, SOC*13, CR11, CB13, DMAS17].

Although a vast variety of available geospatial data exists that can be used to represent a location, much work has gone into synthesizing geospatial data in order to obtain a better visualization, correcting the available data sets, or building a virtual environment. For instance, a comprehensive survey on how to construct 3D buildings using point clouds is presented in [MWA*13]. Road systems can be also generated using sketch-based systems [ALD12, McC08]. Terrains can be synthesized using Digital Elevation Models (DEM) [ZSTR07] or multiresolution approaches [BSS08]. A variety of different data sets are also integrated in [SR16, KRS15] in an interactive system that corrects the mismatches between the elevation data sets and imagery. In [SR16, KRS15], in addition to employing geospatial data to determine the location of the geospatial features, some features are also synthesized to produce better representations for roads, trees, and water bodies.

2.2. 3D Printing

Many applications have been proposed for 3D prints. For instance, they have been used to recreate cultural heritage [SCP*14], to make 3D puzzles [XLF*11, LFL09], or produce objects that can stand or spin [PWLSH13, BWBSH14]. In [DMAS17], a scale model of the world was created to help visualize geospatial data sets that span large regions of the globe. Our system considers a new application for 3D prints (i.e. landscape creation), which, unlike [DMAS17], can show detailed information for much smaller regions, like cities or mountains.

Although 3D prints have many applications in fabrication, they still have some limitations. For instance, much research has been done to strengthen 3D prints by fortifying high-stress parts of the print [TJ11, SVB*12, ZPZ13]. Since the result of the 3D printing is usually stiff and static, some related work is devoted to providing elasticity and movement to the final 3D printed object [PZM*15, SBR*15, PTC*15]. To avoid wasting material, new methods of packing a segmented 3D object with less gaps and therefore less need for supporting material have been proposed in [VGB*14, YCL*15].

Since we wish to use affordable and small 3D printers, we must grapple with two significant challenges from 3D printing: limitations on the number of colors and materials as well as limitations on the size of the printing volume. In order to produce a better looking 3D print, in [MAWS15] a system was proposed to segment a model into nearly developable patches that can be used as a guide for pasting desired materials on top of a 3D print. Some systems have also been designed to print a texture on an intermediate domain such as water [ZYZZ15, PDP*15] or plastic [SPG*16] and then paste it on the 3D print using a simulation of the intermediate domain. To overcome the problem of the size of 3D printers, Chopper was proposed to segment a 3D print into printable pieces that can be later attached to each other using some male and female pins [LBRM12]. Other systems include CofiFab [SDW*16], in which a laser-cut base is constructed by coarsening the mesh. The mesh is then segmented and attached to the base. As a result, the 3D printing time and material are reduced and larger objects can be printed due to the segmentation of the initial mesh.

In this paper, we are concerned with segmenting a large surface representing a landscape region and, unlike Chopper, we do not need to account for arbitrary meshes. As we have different geospatial features in a landscape region, we can segment each of their corresponding meshes separately, which results in a more appropriate segmentation. We use a grid segmentation for each of these features while respecting a cost function that minimizes both the total number of pieces and the number of small pieces. We also solve the limitations on color by creating separate extruded layers for different geospatial features.

3. System Overview

To print a physical model of geospatial regions, the region of interest on the globe is taken as input (see Figure 2). Our system then retrieves the required geospatial elevation data and features corresponding to the region based on the user's needs from a preselected set of sources, including OpenStreetMap (OSM) and Canadian Digital Surface Model (CDSM). Some examples of these features include vegetation area, roads, lakes and buildings. Elevation data is stored as a regular grid of data points, while features are in vector format, representing a path (e.g. roads) or boundaries (e.g. lakes). Our system creates an initial 3D mesh that respects the elevation data and the features. This 3D mesh is then converted to a set of submeshes for each feature. These submeshes are extruded as layers for 3D printing, and if a layer does not fit inside the 3D printer, it is segmented into smaller printable pieces. In the final phase, our system creates a 3D model file after adding extra details to each piece to assist with feature recognition. Assembling all of the 3D printed models creates the final physical model of the desired region. Each step of this process, as illustrated in Figure 2, is described in following sections.

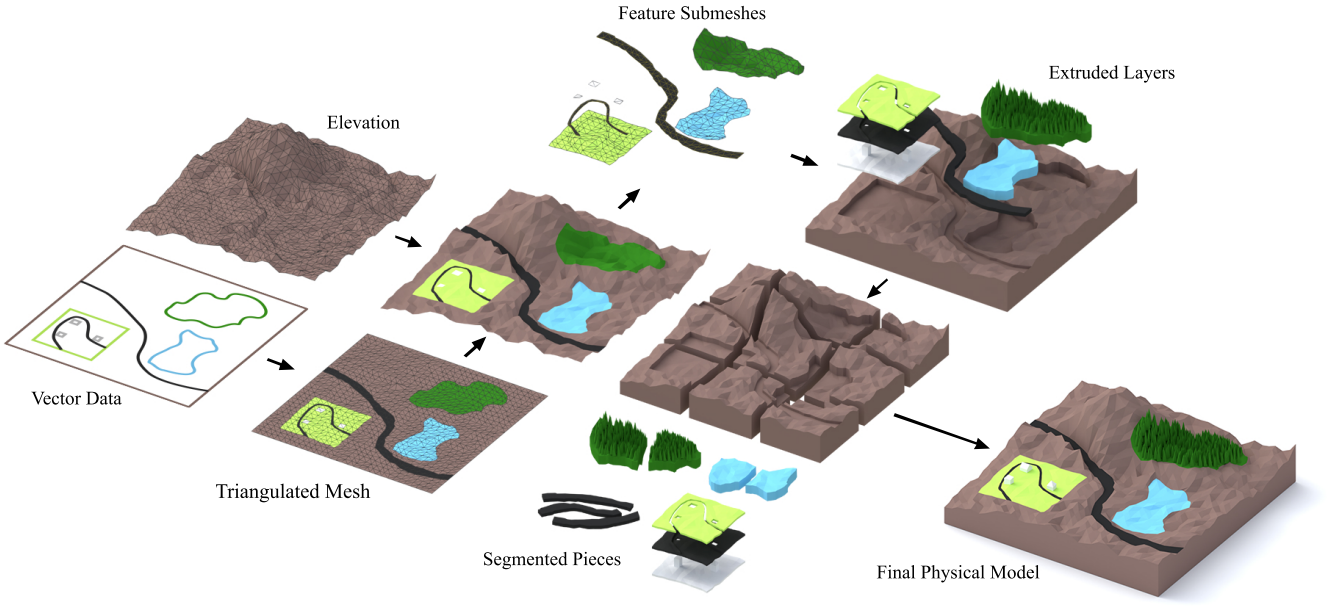


Figure 2: *We use several 2D GIS sources to retrieve the required features and elevation data for the region of interest. A Constrained Delaunay Triangulation is employed to create an initial 3D model. Using a technique based on winding numbers, feature submeshes are extracted from this model. These submeshes are used to create extruded layers. Each of these layers is designed based on the characteristics of its corresponding feature to create a more realistic model. For extruded layers that do not fit inside the 3D printer, a grid segmentation method is used to create appropriate smaller pieces. Eventually, printing all the pieces and assembling them creates the final physical model.*

4. Mesh Generation

In order to 3D print an object, a 3D model has to be created based on the GIS data. Although creating a regular 3D mesh from the elevation data is straightforward, constraining the model to respect feature data requires more consideration. To create this 3D model, we first generate a 2D mesh by employing Constrained Delaunay Triangulation (CDT) [Che87], with vector geospatial features as constraints (see Figure 2). Adding elevation data to this triangulation produces our initial 3D model as a mesh. To respect the resolution of the elevation data, we restrict edge lengths to an appropriate limit (Figure 3).

Figure 3: *For any number of features (a), we use CDT to create an initial mesh (b). A region growing algorithm is applied later to determine what feature each face belongs to (c).*

4.1. Creating Separate Feature Layers

Our initial 3D elevation mesh can be used to identify the peaks and valleys of the selected region. However, geospatial features are not necessarily visible in this model. Those features describe real-world entities, such as roads, buildings or vegetation areas, and are important for understanding a region of interest. To help identifying them, we can use a separate and appropriate material with a natural and familiar color (e.g. translucent blue for lakes or body of water) for printing the feature region. Alternatively, rather than using a separate material, we can extrude those regions to make them visible on the mesh. However, this solution requires a guide to describe each feature and may result in confusion for detecting features.

To use a different material for each feature, we create a set of *feature submeshes* and then extrude each of these submeshes to create a set of *extruded layers* (Figure 4). We extrude the submeshes vertically to avoid overhangs on the edges of printed layers. This will increase the accuracy of layer edges and helps fitting them in the final physical model. A detailed discussion is provided in Section 7. These extruded layers can be later attached by indenting the underlying mesh. We can also add details to each of these extruded layers based on the properties of the feature associated with that feature layer, as described in Section 6. Eventually, these extruded layers should fit together nicely into the indented mesh to represent the final model (Figure 1).

4.1.1. Extracting Feature Boundaries

Geospatial features are represented as a set of points, pathways or areas tagged with contextual information to identify their corre-

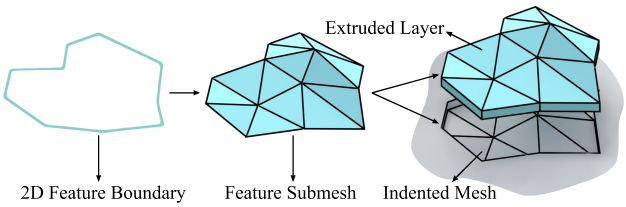


Figure 4: *In order to assign different materials to each feature, we create separate layers for each feature.*

sponding entities. To create extruded layers for each feature, our system begins by extracting submeshes for all of our input geospatial features. For each of these features, our system obtains a vector of 2D points defining the boundary of an area that a feature occupies in 2D. In the case of pathway features, as described in Section 6.1, we transform them to area features by applying a slight offsetting. This is followed by extracting boundary loops as closed paths from the vector of points, and determining whether each boundary loop is an outer boundary or an inner boundary. We assign a counter-clockwise direction for outer boundaries and a clockwise direction for inner boundaries. These directions help us define the inside region of each feature. We input all of the directed boundary paths for each feature to CDT, and retain this direction information for each edge that corresponds to a feature boundary in the resulting triangulation.

4.1.2. Feature Interior Marking

To create feature submeshes, it is necessary to know the interior triangles encompassed by the feature vectors. These feature vectors are a collection of line segments (Figure 3b), and their interior triangles are not readily available; that is, extracting features submeshes out of these vectors is not a trivial task. Feature submeshes may overlap or contain other feature submeshes (e.g. buildings inside a vegetation area). In the case of feature submeshes contained in other feature submeshes, we need to extract the *containment hierarchy* of the feature submeshes. This hierarchy is beneficial in several ways. For example, we can exclude roads that are outside a residential region from being printed, or we can use it to create a stencil layer, as described in Section 6.3.1, to print all the feature submeshes contained in a residential area. Hence, a robust method is needed to identify the feature hierarchies for the feature submeshes. To achieve this, we perform a region growing algorithm that extracts the containment hierarchy and accounts for overlapping feature submeshes.

We denote the list of all the triangles in the mesh as M, and the i_{th} feature as f_i. The feature submesh M_i represents the submesh of M which is inside f_i. We say f_i is contained in f_j if and only if all of the triangles in M_i are also present in M_j. Based on this definition, we create a tree T that represents the hierarchy of feature submeshes, where each node of this tree is a feature boundary, and the feature boundary f_i is a child of f_j if and only if f_i is contained in f_j. For a given triangle t, we let f_t denote the smallest feature that contains t (Figure 5a). We can then find the hierarchy of feature containment by traversing all the ancestors of the node corresponding to f_t in T.

To determine feature containment and also the smallest feature for every triangle in the mesh, we use a technique based on winding numbers [HA01]. Our algorithm begins by traversing all triangles in M. For each triangle t, we cast an infinite ray from the center of the triangle. If we cross an edge on the boundary of feature f_t, similar to winding number method, we determine if the ray is exiting the feature based on the edge direction. In this case, we check if we have a node for f_t in our tree T. If so, we return f_t as the smallest feature that the triangle t is contained in. If there is no node for f_t, we continue traversing the infinite ray to find all the edges intersected by the ray that belong to a feature boundary (see Figure 5b). For each of these edges, we assign a winding number w_{f_i}, based on the direction of the corresponding vector in the feature boundary f_i. We create a sequence of these winding numbers, starting from the closest crossing. An example sequence for the feature boundaries in Figure 5b can be seen below:

$$(+1_{f_i}, -1_{f_i}, +1_{f_j}, +1_{f_i}).$$

We reduce this sequence by cancelling each -1 with the *next* $+1$ for the same feature boundary. For instance, the example sequence above reduces to:

$$(+1_{f_i}, +1_{f_j}).$$

This sequence shows the hierarchy of feature submeshes surrounding this triangle (f_i, f_j). We apply this process for all triangles to both create the hierarchy tree T and find the smallest feature submeshes that contain each triangle. As explained in Section 4.1, these feature submeshes are used to create extruded layers (Figure 3c).

Handling Partial Covering: Geospatial features can naturally overlap (see Figure 5b). In our physical models, only the surface of a region is visible and we need to choose one feature, out of all the overlapping features, for each triangle to belong to. This is not a trivial task, as there is no information in our input data about the feature type precedence. To address this issue, we offer two possibilities. First, we use a priority list defined by the user, and in case of overlaps these priorities are used to decide which feature submesh should be chosen. If the priority is not explicitly defined, we utilize a statistical procedure to determine which feature has top priority by checking the corresponding feature boundaries, summing the edge lengths for those edges along the boundary of the overlap, and choosing the one that has the longest total length.

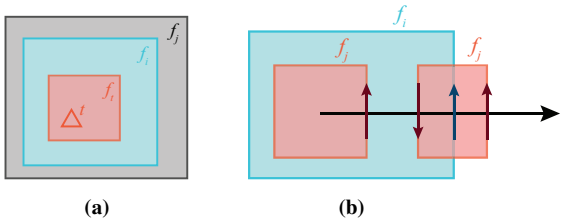


Figure 5: *In (a), the red feature submesh is inside two other feature submeshes. To detect this situation, as shown in (b), we cast an infinite ray to determine the hierarchy of surrounding feature submeshes. Each arrow here shows a crossing for a feature boundary.*

From this, we choose the dominant feature submesh and ignore small overlapping feature submeshes.

5. Segmentation

After generating each feature submesh, we need to create an extruded layer that can be printed with a different material. However, if a feature submesh is larger than the printable volume of our 3D printer, we need to break it into smaller pieces first, and then we can extrude them to create extruded pieces for printing. We propose a method to create these pieces such that they fill the printable volume as much as possible. Although our method is extendable to volumes, in our application, we mainly encounter feature submeshes that only exceed the 2D plane limitation. Furthermore, in those rare cases that a feature submesh extends beyond the height limitation, segmentation is trivial and does not require a sophisticated method. Hence, in our method we ignore the height limitation and project feature submeshes into the 2D plane for segmentation. In the subsections, we explain our method in detail.

5.1. Local Grid Segmentation

The largest printable area takes different shapes and sizes in different 3D printers. These shapes primarily include squares, rectangles and circles. To simplify the segmentation, for all of these cases we consider the largest square that fits within the printable area of the 3D printer to be the maximum size of a printable mesh. We later describe how to employ the unused areas to optimize the printing process in Section 5.2.

Our problem is similar to the polygon interior covering (PIC) problem, where the aim is to cover a target polygon with convex polygons. This problem is known to be NP-hard [CR94]. In PIC, the only criterion is the number of convex polygons used for covering. In our case, we are using squares to cover each feature submesh. Unlike PIC, after covering the whole feature submesh, we need to evaluate each resulting piece to see how appropriate they are for 3D printing (Figure 6).

To cover each feature submesh, we create a regular grid with each cell having the largest square size that is printable. By placing each mesh over this grid, we can break it into smaller pieces that fit inside the 3D printer. Since we use a local grid for each feature submesh, we can independently translate and rotate the mesh to find different ways to slice it. We call each translation/rotation pair a *configuration* in our algorithm. Based on the specific requirements of 3D printing, we define a cost function to evaluate each configuration and choose the optimal one (Figure 7). In the following section, we discuss how we perform this task and discuss the cost function.

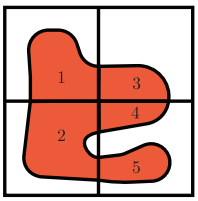


Figure 6: *Each piece is a connected component of a feature submesh after segmentation. We need to evaluate each piece based on how appropriate they are for 3D printing.*

5.1.1. Cost Function

There are many different metrics to consider when segmenting a feature submesh into a set of smaller pieces. We are interested in creating pieces that are not too small, as they are harder to fit into the final printed model and also more fragile. To prevent creating small pieces, we define the cost function as,

$$C(r,t) = \sum_i \frac{1}{a_i},$$

where $C(r,t)$ is the cost of using the configuration with rotation r, and translation t and a_i is the area of each piece (Figure 6). Note that a_i is determined after uniformly scaling the feature submesh and the grid to make the sides of each grid cell unit length. The upper bound of our cost function is infinity, as there is no limit to how small the pieces may become. To find the lower bound, we need to solve the following minimization,

$$\min_{(r,t)} \sum_i^N \frac{1}{a_i},$$

where we assume we have $N \in \mathbb{N}$ pieces. Solving this minimization leads to the answer $a_i = \frac{A}{N}$, if we denote the area of the mesh as A. Therefore, for a fixed number of pieces, we have,

$$\min \sum_i^N \frac{1}{a_i} = N \times \frac{N}{A} = \frac{N^2}{A}.$$

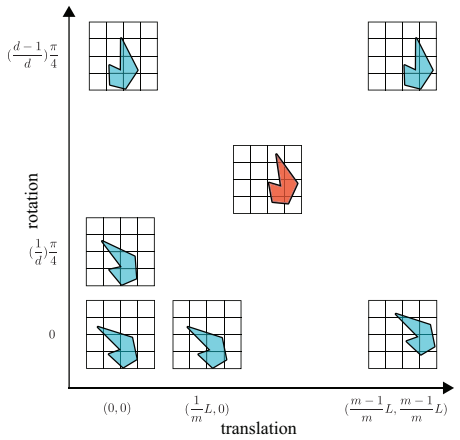


Figure 7: *For each orientation of a feature submesh on a regular grid, where L is the grid cell size, we search different possible translations, shown as a 2D vector in the figure. To speed up this search, each translation axis is uniformly discretized to m values and orientation is uniformly discretized to d values. We choose the optimal configuration for breaking the submesh into smaller pieces, shown as red.*

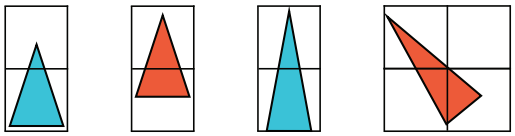


Figure 8: *The shape in (a), can be repositioned on the grid to create equal area pieces (b). In (c), there is no way to create 2 equal area pieces with two grid cells, and achieving the lower bound cost is not possible. If we allow for the creation of more pieces, we can create 3 equal area pieces (d).*

The cost function has hyperbolic growth as we decrease the areas of the pieces, and prevents creating small pieces. Moreover, if we increase N, the minimum cost will increase quadratically, which illustrates that the cost function also minimizes the number of pieces. As we know the minimum value for N (with unit length grid cell dimensions) is $\lceil A \rceil$, we can determine the global minimum:

$$\min \sum_i \frac{1}{a_i} = \frac{\lceil A \rceil^2}{A}.$$

This lower bound cannot be achieved for all shapes of a mesh. For example, in Figure 8a, the shape of the mesh allows one to create equal area pieces (Figure 8b). In Figure 8c, we see an example where creating equal area pieces with the minimum number of pieces is not possible. In some cases, it may be possible to create equal area pieces if we first break the mesh into more pieces (Figure 8d). However, in general creating more pieces is not desirable, and we should consider adding more pieces only if it results in a smaller cost for the whole mesh.

In order to see under which conditions the cost function will increase the number of pieces, we can perform an analysis of the cost function. We start by checking when it is possible to ensure that increasing the current number of pieces would not result in a smaller cost value. If, for the current configuration, $a_i \geq \frac{A}{r}$, where $r \leq 1$, and $M \geq N$ is the new number of cells, we have,

$$\sum_i^N \frac{1}{a_i} \leq \frac{Nr}{A} \leq \frac{M^2}{A} \Rightarrow$$
$$\frac{1}{r} \geq \frac{N}{M^2}.$$

Therefore, we can be sure that no better answer will be found by increasing the number of cells to M if $a_i \geq \frac{N}{M^2}A$. This suggests that, if there is a case in which increasing the number of pieces can prevent creating a very small piece, the cost function would allow for an increase in the number of pieces. As we discuss in Section 5.1.2, we attempt to search the discretized space of configurations as much as possible, and if a better configuration with more pieces is found, we would consider that as well.

5.1.2. Finding the Best Configuration

To find the best configuration, we can rotate and translate each feature submesh on a regular grid with cells that are as large as the biggest square that fits in the printable area of the 3D printer.

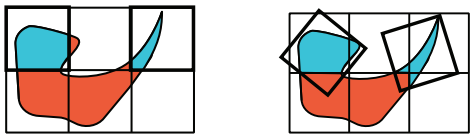


Figure 9: *The blue pieces in (a) after post-processing can include neighbor pieces as well, as in (b).*

The range of possible translations, as we are searching on a regular square grid, is:

$$0 \leq t_x, t_y < 1, \quad (1)$$

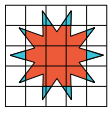
where t_x and t_y are translations along the x and y axes respectively. For the rotation, thanks to the symmetry of the square grid cells, we need to only rotate up to 45 degrees:

$$0 \leq r \leq \frac{\pi}{4}. \quad (2)$$

All possible combinations of r and $t = (t_x, t_y)$ define all the possible configurations.

To solve this non-linear optimization, we discretize the ranges of both r and t to ten uniform steps and perform a brute-force search. Since the segmentation time of feature submeshes is quite negligible compared to the 3D printing time, we decided on using this exhaustive search to avoid any probable local minima. Apart from creating more undesirable pieces, a local minimum can potentially result in more 3D printing sessions. This leads to an increased overall 3D printing time due to the required preparation time for each session. Furthermore, the discretization resolution can be changed in our system if a higher accuracy is required.

5.2. Post Processing



A local grid can help to segment a feature submesh separately from the rest of feature submeshes. However, as we limit the segmentation to use a single square grid for the whole feature submesh, creating some undesirable small pieces might be unavoidable, as shown in the figure on the right. One way of fixing this issue is to merge some of the resulting pieces locally after segmentation (Figure 9).

We start by creating a connectivity graph from the results of the local grid segmentation. Each node in this graph represents a piece, and we connect two nodes if the corresponding pieces are adjacent (Figure 10). In order of highest cost, we traverse the nodes of the graph and merge all adjacent pieces until the resulting piece completely fills the full printable area (not the largest fitting square, which could be a circle or a rectangle), and then repeat the process for all the remaining nodes in the graph.

6. Designing Features

For each feature, we need to create a feature submesh and extrude it. Without additional information, the only way to identify a feature, besides its shape, would be material and color. However,

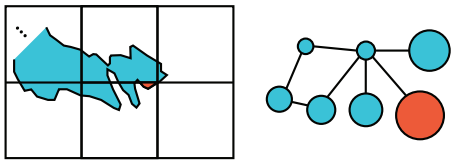


Figure 10: *Here a part of a feature submesh and its corresponding connectivity graph is shown. The size of the nodes in the graph is proportional to the cost of their corresponding pieces. The node with the highest cost is shown in red.*

changing the material and color is not sufficient for visualizing certain features, such as vegetation regions. Therefore, we need to add special characteristics to feature submeshes and modify them. In order to do this, we should consider two important goals: a) how real it looks, and b) how easy it is to identify. We address these two goals by using aerial photos of the region that we are printing as a reference to evaluate whether the resulting feature submeshes look real, and also by using general-purpose maps to check if the feature types can be identified at least as easily as on those maps.

6.1. Pathways

The first type of feature we consider consists of roads, trails, waterways and any similar feature that defines a pathway on the land. The raw information for this type is a sequence of points (i.e. feature vector). To create and print an extruded layer for this feature type, we need to create a polygon that resembles a pathway, which we accomplish by applying an offsetting to the input feature vector to create its feature submesh. If we use the exact width of a pathway for offsetting, it can become too thin in the scale of landscapes 3D prints. To help make this feature type more identifiable and create visible pathways, we assign a minimum width of 0.5mm to these features (Figure 11). This limit is based on the general precision of 3D printers. Furthermore, we adjust the relative width of different types of pathways based on their type (e.g. a highway road is made wider than a service road). There are cases in which pathways have complex interactions, for example in residential areas, that cause assembly of the individual pathways to be difficult. In these cases, we join all of the offset pathways into a single feature submesh and attach them to an underlying layer, as described in Section 6.3.1, and avoid producing many fragile pathway models.

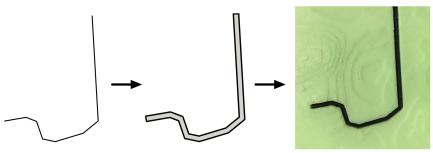


Figure 11: *We connect offset polylines as a simple and fast approach to create an offset polygon for pathways.*

6.2. Vegetation

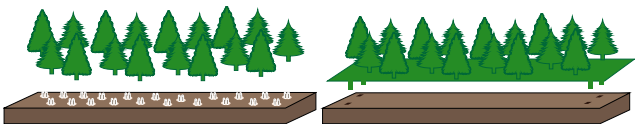
3D printing vegetation is a challenging task. There are several established methods for simulating virtual vegetation regions. However, printing a huge number of delicate and highly detailed tree models is not possible due to the printer limitations. Also, the miniature scale is fragile and hard to print and assemble. One approach to overcome this issue is to use a small number of large trees to symbolically represent the vegetation area. With this approach, the final print is more stable. However, this method drastically simplifies the vegetation region by ignoring the density and size of the trees.

To improve the resemblance between the printed region and the real region, we populate the vegetation area with tree models, while creating variations in the size, shape and distribution of trees by using a simulation technique. To find a natural distribution for a vegetation area, we use an L-system grammar inspired by the work of Lane and Prusinkiewicz [LP02]. The two main production steps are: a) self-thinning, where larger trees dominate smaller ones and b) senescence, where some of the old trees die and are removed. The final result of this simulation will give a set of tree positions and each's respective size.

With a natural distribution for trees, we can create the vegetation area by placing tree models at each of the distributed tree positions and sizing them based on the simulated tree size for that position. However, at the scale of our physical models, the vegetation area may cover thousands of trees. For example, our simulation in the Lauterbrunnen region of Switzerland produced more than 5000 tree positions. It is not possible to print all of these tree models individually. One way to address this challenge is to blend all of the trees on the extruded layer (Figure 12).

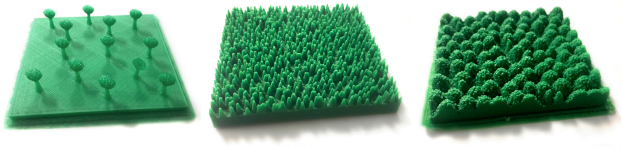
Another issue lies in the complex nature of detailed tree models. The real diameter of a tree in our physical model is too small, and no useful details can be seen at this scale. For example, at the 1:2000 scale of our Lauterbrunnen model, the diameter of each tree on average is about 2 millimeters (based on the average crown width of the Scots pine, which is about 3.8 meters [SBVV17]).

At this scale, we use simple primitives to represent individual trees. Each primitive is simple for printing, and at the same time resembles a real tree seen from far away (e.g. pine trees can be modeled with cone primitives). These primitives can be sized according to the simulation results and placed on a base extruded layer. In regions that are densely populated with trees, these simple primitives can have many intersections, which are extremely hard to print. For these regions, our system applies a displacement map [MS16] to the feature submesh, which blends the simple primitives with the



(a) *Individual tree models.* **(b)** *Combined tree models.*

Figure 12: *Combining all the tree models into a single layer is beneficial for 3D printing.*



(a) (b) (c)

Figure 13: *Adding tree primitives to the extruded layer. (b) and (c) use displacement mapping to blend the tree primitives in densely populated regions.*

underlying mesh (Figure 13). Each vertex in the feature submesh is translated by a vector based on this displacement map. As the resolution of this map is much higher than the feature submesh, our system increases the mesh resolution before applying it.

6.3. Urban Structures

Creation of urban structures faces two main challenges. One challenge is assembling a large number of buildings in such areas will be hard and time consuming. Another challenge is creating appropriate models for buildings. For many regions in the world, available GIS data lack detailed information for individual buildings, and creation of appropriate models becomes difficult.

6.3.1. Stencil Layers

To ease assembly of a large number of buildings, we combine all of their models into a single feature submesh, which results in another 3D element (Figure 14). The space between buildings may contain other feature types (e.g. roads and grass fields). To include these feature types, our system can generate extra layers to overlay atop the buildings layer (see Figure 15). Some feature types like roads can be very delicate and complex to print and overlay. In order to better handle these feature types, our system can combine their extruded layers into a single layer called a *stencil layer*. As shown in Figure 15, these stencil layers are created by excluding from the feature submesh those areas through which one should be able to view the underlying extruded layers. In this way, we produce a final result which is still easy to print and assemble, and is consistent with the real world (Figure 15c).

Our system is given a configuration file that describes all the

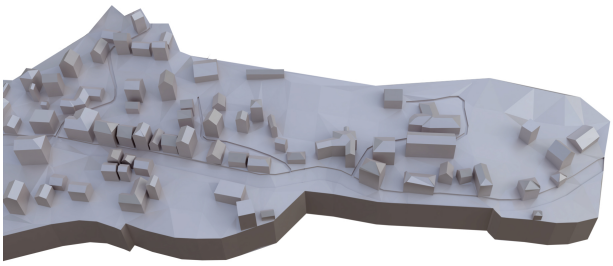
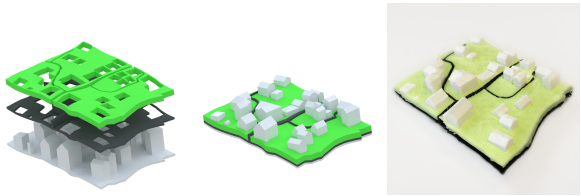


Figure 14: *3D render of a set of buildings combined into a single extruded layer.*



(a) (b) (c)

Figure 15: *We overlay stencil layers, on top of buildings to add missing and otherwise fragile feature types.*

(a) *Gable roof.* **(b)** *Hip roof.* **(c)** *Flat roof.*

Figure 16: *Different types of generated roofs.*

features that should be displayed using stencil layers atop a parent layer. In the first step, this parent layer is created by blending a main feature (e.g. buildings) on a base feature layer (e.g. residential area). This main feature is given extra extrusion before blending to compensate for the area that stencil layers will cover. Afterward, each stencil layer is created by creating holes in the base feature layer for the underlying features and blending the corresponding feature to the stencil layer. Due to the 3D printer's lack of precision, these pieces will not fit together if we print them directly afterwards. Our system applies an offset to the boundary of each stencil layer to help with the fitting.

6.3.2. Building Models

Based on the scale of a physical model, buildings can be modeled as simple blocks, individual boxes or as detailed as a house with windows, doors and other features. At our desired scale, roofs play a tangible role in making building models more realistic. The 2D GIS data retrieved by our system only provides the outlines of the buildings and no information about the precise design of each building. We use a statistical distribution approach to randomly design each building's roof based on what can be observed from aerial photos of the region. For example, we may observe that around 90% of the buildings have gable style roofs, 5% of them are hip style and the rest are flat roofs. We then create roofs using these findings and design each roof accordingly (Figure 16). To create gable and hip style roofs we extract straight skeletons.

7. Results and Discussion

We implemented our system in C++ using the CGAL and Boost libraries. The input to our system is a configuration file in JSON format that describes the extents of the desired region to be printed, the required scale of the model, the desired features and their properties and the 3D printer maximum printable volume. We used

a MakerGear M2 as our 3D printer, with a printable volume of $20cm \times 25cm \times 20cm$. Our printer had only one extruder, limiting us to only one material per printing session. For slicing and printing, we used Simplify3D with medium quality settings, and for printing material we used PLA.

The expected output of our system is a physical model that respects the input GIS data. We have tested our system on two regions: Lake Louise in Alberta, Canada and Lauterbrunnen valley in Switzerland. For each physical model that we 3D print, we evaluate the result based on how accurately the resulting 3D printed pieces respect the input GIS data. We also discuss our system design decisions and how they impacted the printing and assembly of our physical models.

We checked the accuracy of our physical model by first comparing the 3D model files' dimensions with their respective input GIS data. The error in the 3D model files was much smaller than the 3D printer precision, and were the result of rounding-errors in the floating-point calculations (e.g. intersection and union operations). Our system uses arbitrary-precision arithmetic to vastly reduce these errors, reducing them to a maximum error of around $10^{-13}m$. Practically speaking, the final error for each 3D printed piece is dominated by imprecisions introduced during 3D printing, which makes the assembly difficult. One source of imprecisions is the 3D printer and the 3D printing settings used. When using FDM process, support structures need to be printed to hold the parts of a model hanging in the air (i.e. overhangs). These parts of the model suffer more inaccuracy due to both lack of a solid base in those regions and remnants of support structures on the model. Another source of error while using FDM process is warpage. As measured and analyzed in [ABC18], the maximum horizontal dimension of a model can increase this warpage. Therefore, we may conclude segmenting larger models to smaller pieces could also decrease the imprecisions due to warpage.

By employing vertical extrusion, we prevented overhang imprecisions on the boundary of our pieces. For our first physical model, the Lake Louise model, we managed to fit most of the pieces without any post-processing. However, fitting some parts proved to be difficult and required some sanding. To overcome this issue, after doing several tests, we realized an offset of 0.4 millimeters for the boundary of each piece would help fitting them with no difficulties. Assembling our second physical model, confirmed this offsetting solves the fitting problem.

Based on our experience, the maximum horizontal error for our pieces was about 0.2 millimeters. However, the average vertical error was about 0.5 millimeters, reaching to a maximum of 1 millimeter. The main reason for this increased error, as discussed above, is the need to print support structures for holding the model, while another reason would be warpage of larger pieces. The following sections discuss each of our results in more detail.

7.1. Lake Louise

In our first attempt, we used our algorithm to print the area around Lake Louise. The final physical model is $80 \times 66cm^2$, and consists of 25 pieces for the base model, 23 pieces for the vegetation area and 3 pieces for the lake. Other feature types required no segmentation. The whole model contains 61 pieces that were printed during 55 printing sessions (we achieved 10% reduction in the number of sessions due to packing). A translucent blue was used to print the lake pieces. We used an appropriate color for each of other feature types, for example a green material for the vegetation area. To enhance the aesthetics of the base material, which was printed separately before assembling the model, we applied a textured spray. These separate layers allow us to perform different kinds of post-processing for each piece without affecting the other pieces. The final result can be seen in Figure 17.

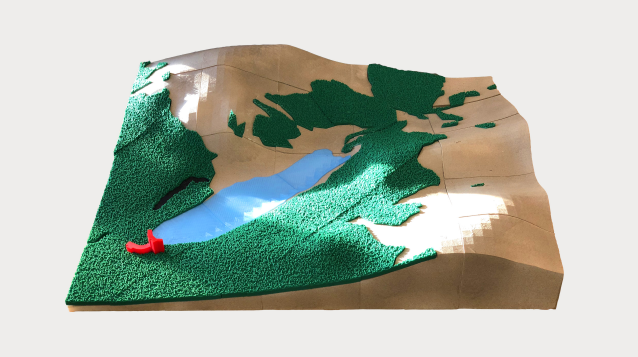
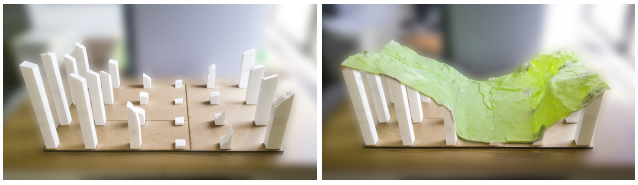


Figure 17: *The resulting physical model of Lake Louise area. For more images, please refer to supplementary material.*

7.2. Lauterbrunnen

After evaluating the Lake Louise physical model, we realized the available data sets were not diverse enough, as it mostly contained vegetation areas. Therefore, we decided to test our algorithm with datasets representing more complex features. Lauterbrunnen valley contains many different feature types with a very interesting elevation model that makes it one of the deepest valleys in the Alpine chain, where the mountains rise to more than 700 meters on either side. We used a scale of 1:2000 for the Lauterbrunnen model, resulting in a dimension of $96 \times 57cm^2$ for this region.

Lauterbrunnen has a very high frequency of large elevation changes (reaching about $38cm$ of elevation change in our model), making it wasteful to print the entire base naively. Such an approach would take about about 20 days and more than 10 Kg of filament for the base pieces, even when using a very low amount of printing infill (like 5%). Some parts also have a height beyond the printable

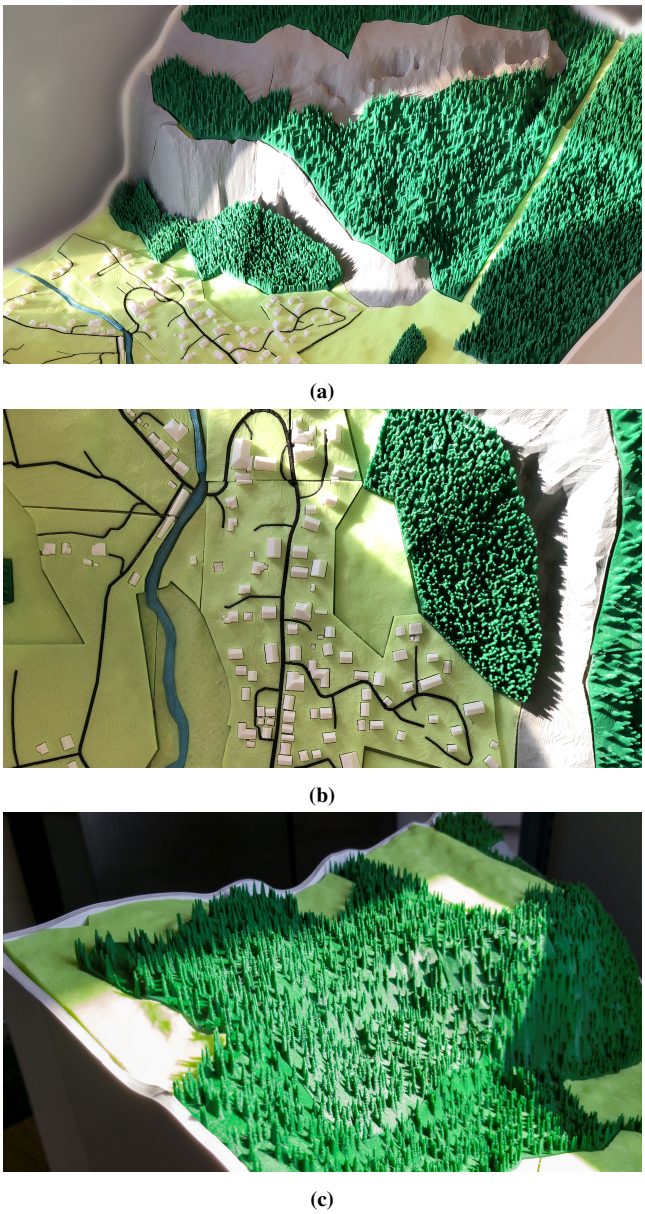


(a) *Pillars.* **(b)** *Pillars with surface attached.*

Figure 18: *We use pillars to reduce printing time and material use.*

volume of our 3D printer. In an attempt to solve these issues, we used small pillars to hold the surface of the model (Figure 18). The visible part of the base is created as a thin layer (i.e. 0.5 millimeters thick) and attached to the pillars. Printing time for these pillars is far more reasonable, and several pillars can be printed in one printing session.

The total number of separate pieces for this model was about 125 pieces, which were printed over 88 printing sessions with a total printing duration of 29.5 days (we achieved a 30% reduction in the number of sessions due to packing). About 9% of these sessions failed due to the following issues: 1) filament getting stuck in the 3D printer nozzle, 2) running out of filament in the middle of a printing session, 3) imprecise leveling of the 3D printer's bed leading to weak adhesion of model and failing the print later, and 4) printing over USB and failure of the connection. As we faced these issues, we tried to reduce them by keeping a precise record of material usage, leveling the 3D printer's bed frequently, and printing directly on the 3D printer using SD card. Of the aforementioned printing duration, approximately 14 days were spent on printing the base surface layers and pillars, which shaved a week off of the naive approach and prevented the increased failure rate that accompanies longer printing sessions. Overall, the final physical model used 10.9 Kg of filaments, costing about 327 CAD. The final physical model can be seen in Figure 19. One issue we discovered after assembling the final model was a slight vertical misalignment of several adjacent sections (see Figure 19b). This is due to stacking several stencil layers. Small imprecision of each stencil layer accumulates and makes some perceivable misalignments. This issue can be easily addressed in our system by increasing the depth of indentation in the offsetting stage. Figure 20 shows a reprint of a small region in the physical model of Lauterbrunnen valley after adjusting this parameter in our system.



(a) **(b)** **(c)**

Figure 19: *The resulting physical model of Lauterbrunnen valley.*

8. Conclusions and Future Work

We presented a system that can print physical models of landscapes using an affordable 3D printer, and provided solutions for two main challenges that must be faced when printing with these printers. The size limitation of 3D printing is addressed by using a grid segmentation algorithm that is applied locally to each feature submesh. Using separate models for different feature types and creating stencil layers both provides a way to use any number of materials and to incorporate complex feature interactions without producing very delicate feature layers.

Using a square grid for segmenting each feature submesh makes our method efficient and easy to implement. However, it may result in some undesirable pieces. We apply a simple post-processing step that will merge some of these undesirable pieces. Using non-square grids or more sophisticated post-processing can help to generate more appropriate pieces.

We have provided a method to print vegetation areas using a displacement map. This method can be extended to create different types of vegetation areas with different types of trees and more realistic distributions based on density information from GIS sources. These designs can be also extended to create appropriate textures for other types of features, like mountains, deserts and water bodies.

By creating stencil layers, we provided a way to print several feature types in a region with complex interactions. Even though we used very thin underlying base layers for blending features, the material usage can be further reduced by carefully removing unnecessary unseen parts of these layers.

9. Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. We wish to thank Troy Alderson for helping with reviewing the paper.

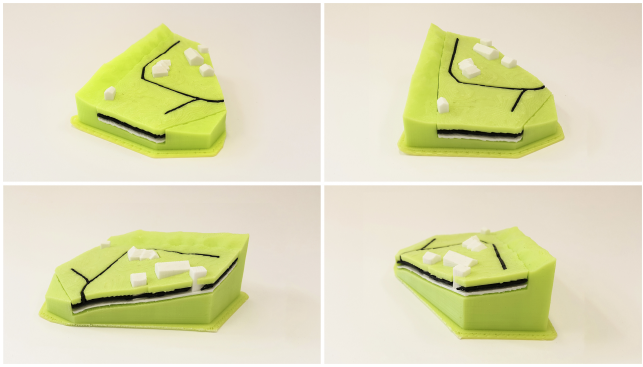


Figure 20: *Using more indentation, our system can account for accumulated imprecision of stencil layers.*

References

[ABC18] ARMILLOTTA A., BELLOTTI M., CAVALLARO M.: Warpage of fdm parts: Experimental tests and analytic model. *Robotics and Computer-Integrated Manufacturing 50* (2018), 140–152. 10

[ALD12] APPLEGATE C. S., LAYCOCK S. D., DAY A. M.: A sketch-based system for highway design with user-specified regions of influence. *Computers & Graphics 36*, 6 (2012), 685–695. 2011 Joint Symposium on Computational Aesthetics (CAe), Non-Photorealistic Animation and Rendering (NPAR), and Sketch-Based Interfaces and Modeling (SBIM). 3

[BSS08] BROSZ J., SAMAVATI F. F., SOUSA M. C.: Terrain synthesis by-example. *Communications in Computer and Information Science: Advances in Computer Graphics and Computer Vision 4* (2008), 58–77. 3

[BWBSH14] BÄCHER M., WHITING E., BICKEL B., SORKINE-HORNUNG O.: Spin-It: Optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 33*, 4 (2014), 96:1–96:10. 3

[CB13] COZZI P., BAGNELL D.: A WebGL globe rendering pipeline. In *GPU Pro 4: Advanced Rendering Techniques*. A. K. Peters/CRC Press, 2013. 3

[CDS14] Geobase canadian digital surface model (cdsm), 2014. URL: http://publications.gc.ca/pub?id=9.676773&sl=0. 2

[Che87] CHEW L. P.: Constrained delaunay triangulations. In *Proceedings of the Third Annual Symposium on Computational Geometry* (New York, NY, USA, 1987), SCG '87, ACM, pp. 215–222. 4

[CR94] CULBERSON J., RECKHOW R.: Covering polygons is hard. *Journal of Algorithms 17*, 1 (1994), 2 – 44. 6

[CR11] COZZI P., RING K.: *3D Engine Design for Virtual Globes*, 1st ed. CRC Press, 2011. 3

[CW11] CAMPBELL J. B., WYNNE R. H.: *Introduction to Remote Sensing*, 5th ed. Guilford Publications, 2011. 3

[DMAS17] DJAVAHERPOUR H., MAHDAVI-AMIRI A., SAMAVATI F. F.: Physical visualization of geospatial datasets. *IEEE Computer Graphics and Applications 38*, 3 (May 2017), 61–69. 3

[HA01] HORMANN K., AGATHOS A.: The point in polygon problem for arbitrary polygons. *Computational Geometry 20*, 3 (2001), 131–144. 5

[HW17] HULL C., WILLETT W.: Building with data: Architectural models as inspiration for data physicalization. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017), ACM, pp. 1217–1264. 1

[KGL01] KVAN T., GIBSON I., LING W.: Rapid prototyping for architectural models. 2

[KRS15] KETABCHI K., RUNIONS A., SAMAVATI F. F.: 3d maquetter: Sketch-based 3d content modeling for digital earth. In *2015 International Conference on Cyberworlds (CW)* (Oct 2015), pp. 98–106. 3

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: partitioning models into 3d-printable parts. 3

[LFL09] LO K.-Y., FU C.-W., LI H.: 3d polyomino puzzle. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 157:1–157:8. 3

[LP02] LANE B., PRUSINKIEWICZ P.: Generating spatial distributions for multilevel models of plant communities. In *Proceedings of the Graphics Interface 2002 Conference, May 27-29, 2002, Calgary, Alberta, Canada* (May 2002), pp. 69–80. 8

[MAAS15] MAHDAVI-AMIRI A., ALDERSON T., SAMAVATI F.: A survey of digital earth. *Comput. Graph. 53*, PB (Dec. 2015), 95–117. 3

[MAWS15] MAHDAVI-AMIRI A., WHITTINGHAM P., SAMAVATI F.: Cover-it: An interactive system for covering 3d prints. In *Proceedings of the 41st Graphics Interface Conference* (Toronto, Ont., Canada, Canada, 2015), GI '15, Canadian Information Processing Society, pp. 73–80. 3

[McC08] MCCRAE J. P.: *Sketch-Based Path Design*. Master's thesis, University of Toronto, 2008. 3

[MS16] MARSCHNER S., SHIRLEY P.: *Fundamentals of Computer Graphics, Fourth Edition*, 4th ed. A. K. Peters, Ltd., Natick, MA, USA, 2016, ch. 11, p. 270. 8

[MWA*13] MUSIALSKI P., WONKA P., ALIAGA D. G., WIMMER M., VAN GOOL L., PURGATHOFER W.: A survey of urban reconstruction. *Computer Graphics Forum 32*, 6 (2013), 146–177. 3

[PDP*15] PANOZZO D., DIAMANTI O., PARIS S., TARINI M., SORKINE E., SORKINE-HORNUNG O.: Texture mapping real-world objects with hydrographics. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 65–75. 3

[PTC*15] PÉREZ J., THOMASZEWSKI B., COROS S., BICKEL B., CANABAL J. A., SUMNER R., OTADUY M. A.: Design and fabrication of flexible rod meshes. *ACM Trans. Graph. 34*, 4 (July 2015), 138:1–138:12. 3

[PWLSH13] PRÉVOST R., WHITING E., LEFEBVRE S., SORKINE-HORNUNG O.: Make It Stand: Balancing shapes for 3D fabrication. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 32*, 4 (2013), 81:1–81:10. 3

[PZM*15] PANETTA J., ZHOU Q., MALOMO L., PIETRONI N., CIGNONI P., ZORIN D.: Elastic textures for additive fabrication. *ACM Trans. on Graphics - Siggraph 2015 34*, 4 (aug 2015), 12. Julian Panetta and Quingnan Zhou are Joint first authors. 3

[SBR*15] SCHUMACHER C., BICKEL B., RYS J., MARSCHNER S., DARAIO C., GROSS M.: Microstructures to control elasticity in 3d printing. *ACM Trans. Graph. 34*, 4 (2015). 3

[SBVV17] SHARMA R. P., BÍLEK L., VACEK Z., VACEK S.: Modelling crown width–diameter relationship for scots pine in the central europe. *Trees 31*, 6 (Dec 2017), 1875–1889. 8

[SCP*14] SCOPIGNO R., CIGNONI P., PIETRONI N., CALLIERI M., DELLEPIANE M.: Digital Fabrication Technologies for Cultural Heritage (STAR). In *Eurographics Workshop on Graphics and Cultural Heritage* (2014), Klein R., Santos P., (Eds.), The Eurographics Association. 3

[SDW*16] SONG P., DENG B., WANG Z., DONG Z., LI W., FU C.-W., LIU L.: CofiFab: Coarse-to-fine fabrication of large 3d objects. *ACM Transactions on Graphics (SIGGRAPH 2016) 35*, 4 (2016). 3

[SOC*13] SELLERS G., OBERT J., COZZI P., RING K., PERSSON E., DE VAHL J., VAN WAVEREN J. M. P.: Rendering massive virtual worlds. In *SIGGRAPH '13: ACM SIGGRAPH 2013 courses* (2013), ACM. 3

[SPG*16] SCHÜLLER C., PANOZZO D., GRUNDHÖFER A., ZIMMER H., SORKINE E., SORKINE-HORNUNG O.: Computational thermoforming. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 35*, 4 (2016). 3

[SR16] SAMAVATI F., RUNIONS A.: Interactive 3d content modeling for digital earth. *The Visual Computer* (2016), 1–17. 3

[SVB*12] STAVA O., VANEK J., BENES B., CARR N., MĚCH R.: Stress relief: improving structural strength of 3d printable objects. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 48. 3

[TJ11] TELEA A., JALBA A.: Voxel-based assessment of printability of 3d shapes. In *Proceedings of the 10th international conference on Mathematical morphology and its applications to image and signal processing* (2011), Springer-Verlag, pp. 393–404. 3

[VGB*14] VANEK J., GALICIA J. A. G., BENES B., MECH R., CARR N. A., STAVA O., MILLER G. S. P.: Packmerger: A 3d print volume optimizer. *Comput. Graph. Forum 33*, 6 (2014), 322–332. 3

[VTP15] Virtual Terrain Project. http://vterrain.org/, 2015. 3

[XLF*11] XIN S., LAI C.-F., FU C.-W., WONG T.-T., HE Y., COHEN-OR D.: Making burr puzzles from 3d models. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 97. 3

[YCL*15] YAO M., CHEN Z., LUO L., WANG R., WANG H.: Level-set-based partitioning and packing optimization of a printable model. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 214:1–214:11. 3

[ZPZ13] ZHOU Q., PANETTA J., ZORIN D.: Worst-case structural analysis. *ACM Transactions on Graphics 32*, 4 (2013). 3

[ZSTR07] ZHOU H., SUN J., TURK G., REHG J. M.: Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 834–848. 3

[ZYZZ15] ZHANG Y., YIN C., ZHENG C., ZHOU K.: Computational hydrographic printing. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 131. 3