Fabricating Functionally Graded Material Objects Using Trimmed Trivariate Volumetric Representations

Ben Ezair
Computer Science Dept.,
Technion,
Israel Institute of Technology,
Haifa, Israel
benezair@cs.technion.ac.il

Daniel Dikovsky
Stratasys Ltd.,
Rehovot, Israel

Gershon Elber
Computer Science Dept.,
Technion,
Israel Institute of Technology,
Haifa, Israel

Abstract

The methods introduced in this paper allow direct slicing and manufacture of freeform functionally graded material (FGM) objects using additive manufacturing (AM). The FGM objects received as input are specified as complexes of trimmed parametric trivariate volumetric cells, and contain the design for both the geometry and (heterogeneous) material composition of the model.

We present efficient methods that enable the fabrication of general volumetric freeform designs, following the modeling generality of contemporary B-rep geometric modeling systems, while fully exploiting multi-material 3D printers. Our methods allow the application of any material function to the volume of a model, from simply another (possibly trimmed) trivariate function, through a volumetric discrete texture to a procedural function. We complete this work by demonstrating these introduced capabilities by fabricating several functionally graded material objects, using a modern multi-material 3D printer.

1 Introduction

Functionally graded materials (FGMs) are heterogeneous materials created by mixing/interleaving two or more materials, inside the volume of some object, in a way that varies according to position [4]. The spatial variation in FGMs is usually designed to (optimally) achieve some overall physical property, in the object. FGMs are becoming more popular in engineering due to their increased availability through additive manufacturing (AM), and because objects made using FGMs can often achieve functional properties that are superior to those that can be achieved by objects made from a homogenous material.

Current 3D printers, such as the one described in [8], can produce many different FGM objects using AM. That said, the state-of-the-art capabilities of contemporary freeform modeling tools are boundary representation (B-rep) based, describing only the outlines (or surface) of a 3D object. B-reps are incapable of supporting FGM objects without introducing additional information to describe the internal structure of the model.

In this work, we aim to provide heterogeneous AM support via an emerging volumetric representation (V-rep) framework [28, 16], that supports the same general modeling space as B-rep modeling tools, but uses volumetric trivariates. Figure 1 shows an example of a trimmed parametric V-rep. V-reps are compact and can represent both complex geometry and material composition. The paramount success of the parametric B-rep approaches for the last four decades, leads us to believe in the usefulness of V-reps, that can capture the generality of B-reps for geometric design, while providing a simple integration path with the design and analysis processes of FGM objects. The shared parametric domain of geometry and material properties that V-reps offer, allows a designer to easily anchor material properties to geometric features, and map with ease analysis results back into the V-rep (and its material properties).

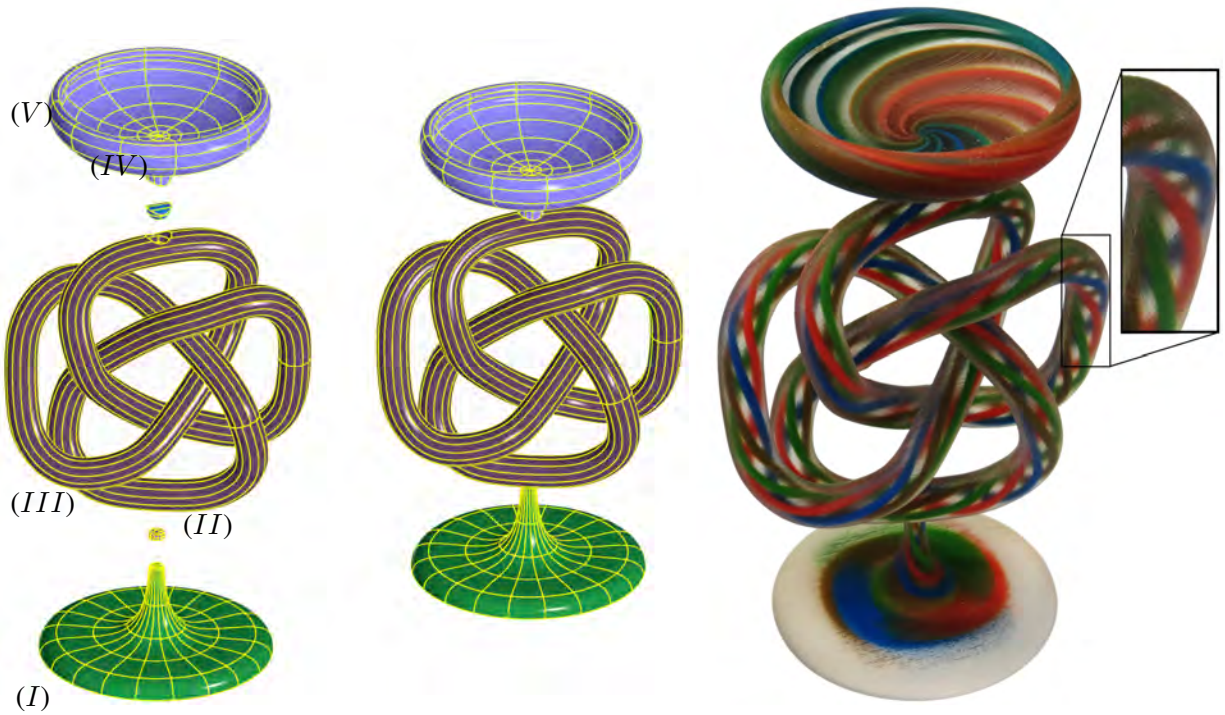

Figure 1 : *A freeform V-rep model using trimmed parametric trivariate B-spline functions. (a) shows an exploded view of the five volumetric cells of the chalice V-rep model in (b). Three of the cells are contained in one trivariate, whereas the two small ones are unions of two different trivariates. (c) shows a fabricated graded-materials (FGM) object (a chalice) based on the model. Printed courtesy of Stratasys Ltd.*

Our first contribution in this paper is the ability to slice and manufacture trimmed V-reps using AM. In other words, given an input FGM object expressed as a V-rep, the methods we describe in this paper enable the generation of printing instructions for AM needed to manufacture the model. Additionally, we enable the efficient retrieval of parametric values for large numbers of Euclidean points within V-reps. This, in turn, allows material composition to be specified in a flexible and robust manner as a general three variable function over the V-rep's parametric domain. We believe this kind of representation will be useful as the end result for FGM design, and can be an additional tool in the fabrication of FGM objects. Figure 1 shows an example of an object based on a V-rep design that was manufactured using the methods we describe in this paper. In this context, using V-reps gave us two advantages: we were able to model the relatively complex geometry as a single V-rep, and the V-rep parametrization allowed us to easily assign and relate material properties to the geometry.

2 Previous Work

3D printers that support heterogeneous materials are becoming more and more common, in recent years. Multi-material 3D printers based on the PolyJet 3D printing technology (e.g. Objet Connex series [8]) are capable of producing FGM objects of almost any shape, and with many material properties by combining different materials at high resolution. Other examples for printers that use a similar ink jet approach include the ProJet [1] and the Jet Fusion [12]. The increased availability of multi-material 3D printers is leading to an increased interest in the use of heterogeneous materials, and in the manufacturing of FGM objects.

Other technologies that can support the creation of an FGM object are those that use powder based printing. In [20], the mixing of metallic powders in different proportions is discussed, while the printed examples in [14] show results created using a Z Corporation's (powder based) 3D printer by using binders of different colors, claiming to simulate material grading.

There are also fused deposition modeling (FDM) printer designs that claim to incorporate the ability to use multiple materials [22], by mixing filaments (in a dedicated mixing chamber) to dynamically change the extrusion material. Another approach suggests mixing the filaments outside the printer, then using the mixed filament in a standard FDM printer [19].

The modeling of FGM objects and their properties is a major topic for research, in recent years. The body of work on the modeling and representation of FGM objects is quite extensive, and this overview is only a sampling of the major approaches as we see them. There have been many different strategies proposed for the specification, design, and manufacturing of FGM objects, with no single strategy emerging as dominant. Indeed, while there is some standardization effort in the representation of objects made from several materials, such as the AMF standard [2] that can support a discrete number of volumes filled by a discrete number of materials, there is still no clearly accepted way to represent FGM objects. A more thorough (but not very recent) review can be found in [15].

Voxels are perhaps the most straightforward way of representing an FGM object, and they fit very well with the sort of input some 3D printers expect, like the input described in [8] (a series of bitmaps). However, memory and computational requirements make them problematic to use for large objects when a high resolution is needed [15], or for other needs such as design and analysis. Approaches such as the one found in [21], can alleviate some of these deficiencies by using a sparse hierarchical representation rather than a full grid to represent a collection of voxels. An example of a system that uses voxels for modeling can be found in [17].

Another option for modeling FGM objects is a piecewise linear volume mesh. In [13], FGM objects are modeled by subdividing them into simple cells (finite element meshes), with material properties defined analytically within each cell. The implementation used in [13] subdivides objects into tetrahedral meshes, of limited continuity, and is somewhat reminiscent of a tessellation operation adapted to volumes.

Like our own work here, [27] also deals with the step that transforms the model produced by the designer, into a set of fabrication instructions. The work in [27] presents an architecture that is inspired by the rendering pipeline commonly used in computer graphics hardware. The input used is a polygonal mesh, and programmable shaders (short programs called fablets) are used to specify the material properties. The focus in [27] is on producing a generic architecture and an FGM representation that can be efficiently converted to AM instructions using their fablet architecture. This, however, requires the design stage (and the designers) to produce models that include program code for the aforementioned fablets which may not be convenient. Additionally, if the material specification is produced by an automated analysis process, the conversion to fablet code may not always be feasible.

An FGM object can also be modeled in a constructive procedural manner. To evaluate the material properties at any point, we run a program that determines the material composition at that point. In [5], this program takes the form of a "reducer-tuner model", a construct that creates the required program based on design goals. In [3], external and user-generated data sets are used to evaluate material distributions.

The material composition of an FGM object at a point (x, y, z) can also be represented by an explicit function $f(x, y, z)$ as in [25]. This representation allows an easy evaluation of the material properties at any point, but is difficult to relate to an object with a complex geometry since an arbitrary coordinate system is used. Other efforts, such as [14], attempt to parametrize the object according to distances from "material features". Using material features, the material properties at any point are determined as a function of the distance to the features.

As mentioned before, parametric volumes were also used toward the modeling of FGM objects. In these representations, FGM objects are modeled as extensions of parametric curves and surfaces by adding a third, volumetric, degree of freedom and additional material information to the control points of these entities. In [23], for example, this is done by specifying control points for the trivariate as: $P_{ijk} = (\bar{x}_{ijk}, M_{ijk})$. In this case, $\bar{x}_{ijk}$ represent positions, and the values of M_{ijk} represent any additional material information needed. M_{ijk} is usually a vector, that has the same number of coordinates as the number of available materials, with each element in the vector representing the volume fraction (out of 1) of a single material. In a work similar to our own [24], a method for directly slicing trivariate volumes is shown. Material information is represented by a B-spline trivariate function (just like the geometry), and is produced by fitting the function to a given set of points that assign material properties to spatial coordinates. They also present some adaptations needed to sample material information for printing techniques that require relatively slow changes in material composition (relative to changes in position). The methods in [24] are only applied to low printing resolutions (tens of thousands of sampled points), and may not scale well to producing the billions of sample points as we use here to work with the native resolution of modern FGM printers.

None of the above offers direct slicing of a trimmed trivariate representation of the model. Supporting trimmed trivariates gives us the capability to capture the space of all B-rep models, as volumetric functions suitable for FGM design.

3 Representing and Fabricating an FGM Object

We model an FGM object (FGM_{obj}) as a cell complex, following [16], of (potentially trimmed) trivariate parametric functions. Each trivariate is assumed regular (with a non-vanishing Jacobian), and without global self intersections. The cells in an FGM_{obj} correspond to the cells of a V-rep in [16], and are denoted as FGM_{cell}. However, unlike the cells in [16], our representation mandates only a single parametrization for each cell, to avoid ambiguities. Each FGM_{cell} is defined as follows:

$$FGM_{cell}(u, v, w) = \{V(u, v, w), M(u, v, w), \mathcal{S}\} \ , \tag{1}$$

where trivariate $V(u, v, w)$ along with its set of trimming surfaces, $\mathcal{S}$, represent the geometry of the FGM_{cell}:

$$V(u, v, w) = (v_x(u, v, w), v_y(u, v, w), v_z(u, v, w)) = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} \sum_{k=0}^{n_w} P_{ijk} B_i(u) B_j(v) B_k(w) \, , \tag{2}$$

where $P_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk})$ and $B_i(u)$, $B_j(v)$, $B_k(w)$ are the control points and blending functions of the trivariate volume. $\mathcal{S}$ is a set of trimmed surfaces that trim the (cuboid topology of the) volume of $V(u, v, w)$ and define the boundaries of the cell. The material composition is defined by a second independent trivariate (vector) function $M(u, v, w)$, spanning the same (u, v, w) domain. For example, the specific FGM object shown in Figure 1 has five cells in the complex. It was created using Boolean operations on three trivariates: the cup, the knot, and the base. Five cells were created overall: two cells (Figure 1(a) (II) and (IV)) consist of the volumes where two of the three original trivariates intersect, and three cells (Figure 1(a) (I), (III) and (V)) consist of the intersection-free volumes, all represented as trimmed trivariates.

In our representation, the material composition function, $M(u, v, w)$, can be any function: for example, a refinement of the spline space of V, a procedural one as in [27], or, may make use of analysis results or material features if needed. We chose this approach because it allows the designers the freedom to use whatever is appropriate for their specific designs. Again, referring to the FGM object shown in Figure 1, the material function in this case was procedural, one function for each of the five cells.

We apply the standard AM approach of slicing [18], to convert the above representation of an FGM object to a set of instructions that can be used to manufacture the object using AM. Figure 2 shows the

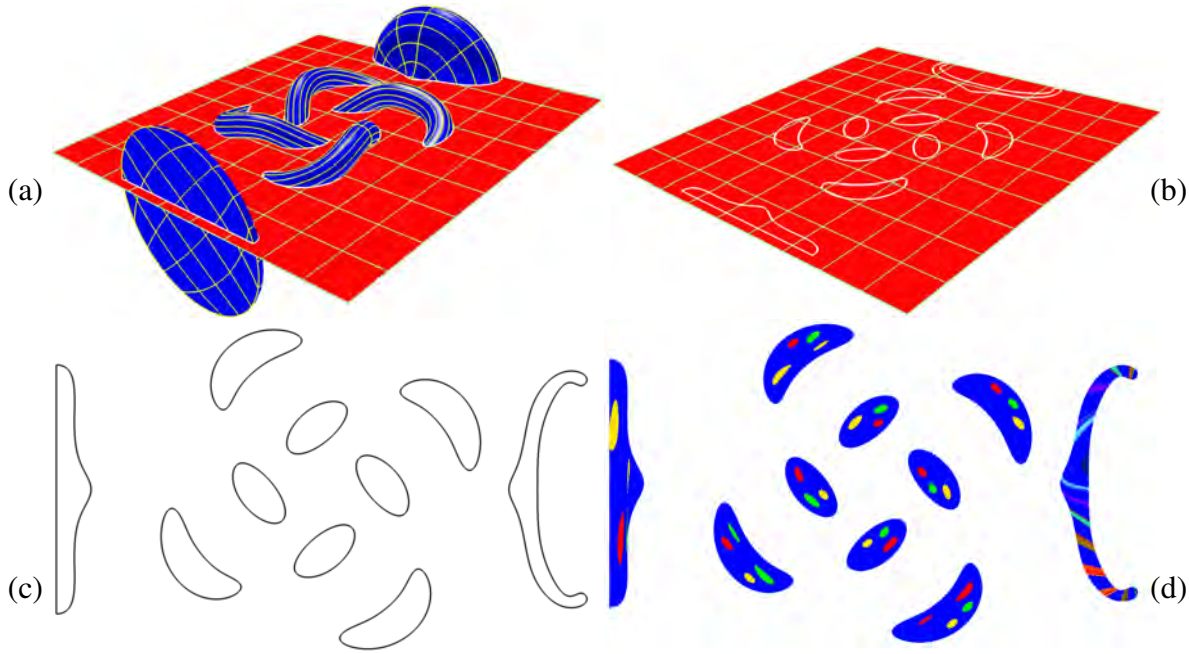

Figure 2 : *Using slicing, the trimmed V-rep model (the chalice from Figure 1) is intersected with planes. In (a), one plane is shown, resulting in a planar outline for the slice, in (b) and (c). Each Slice is then filled with color coded material information (d), by examining the interior of the V-rep model. Compare to Figure 1, where blue reduces to transparent material.*

outline of this process as used in our work. We start with a V-rep and 'slice' its boundaries ($\mathcal{S}$ in Equation (1)) by intersecting it with a plane (at a given z height), as in Figure 2 (a). The result is an outline of the current slice, see Figure 2 (b) and (c). This slice is then filled with material information and can be used by the printer to manufacture that slice (Figure 2 (d)). When using a Connex printer (as in our case), the material information for the slices is given as a set of bitmaps that encode the material composition in each point inside the slice. The difficulty in implementing the slicing procedure described above, for FGM objects in Equation (1), can be found in the requirement to evaluate and specify the material composition at every point inside the slice. When slicing an FGM object, the only information available for a point is its position in Euclidean space. However, the material composition $M(u, v, w)$ is a function of the parametric domain.

As shown in Equation (1), both the geometry and material composition of an FGM_{cell} are functions over the same parametric domain. Given a location in the parametric domain, (u, v, w), retrieving the Euclidean position $V(u, v, w) = (x, y, z)$, and the material composition $M(u, v, w)$ for that parametric location is trivial, and amounts to the evaluation of V and M, at (u, v, w). However, given a point in Euclidean space, (x, y, z), it is more difficult to determine its material composition, since we must solve an *inverse problem* and retrieve the parametric values for that point, if they are defined.

Given a Euclidean point (x, y, z), consider the following system of three (often non-linear piecewise polynomial) constraints and three unknowns (u, v, w):

$$x = v_x(u, v, w)\,, y = v_y(u, v, w)\,, z = v_z(u, v, w)\,, \tag{3}$$

with $v_x(u, v, w)$, $v_y(u, v, w)$, $v_z(u, v, w)$ as defined in Equation (2). Equation (3) prescribes a classical inverse problem. By solving this system of constraints, for example using [11], we can obtain the (u, v, w) values for the given point (x, y, z). However, we expect to solve Equation (3) millions or even billions of times, and to make the process more efficient we also rely on the following result:

Lemma 1. *The solution for the system of constraints presented in Equation (3), (u_0, v_0, w_0), exists and is unique for every point (x_0, y_0, z_0) inside a self intersection free and regular volume $V(u, v, w)$.*

Proof. The uniqueness of the solutions to Equation (3) follows directly from our assumption of $V(u, v, w)$ being regular and without (global) self intersections.

□

Later, we will see how the uniqueness of the solution allows us to optimize the computation of the parametric values, for a given Euclidean point (x, y, z). Once the (u, v, w) value for a given point (x, y, z) is computed, the material composition at that point can be calculated, by simply evaluating $M(u, v, w)$.

As stated before, solving Equation (3) is computationally intensive, and we would like to avoid it whenever possible. On the other hand, as stated before, considering the average printing resolution, one can expect to solve Equation (3) millions or even billions of times, for one model. To efficiently classify points as inside or outside the model, we use the outline of the current slice (see Figure 2(b) and 2(c)). Given the outline, we aim to sample and generate material composition information only for the pixels that are found inside the outline of the slice. We chose the simple rasterization strategy of sampling along straight lines aligned to the x or y axis, as in scan conversion used in computer graphics [10]. Sampling along lines allows us to use two optimizations:

(1) To determine if we are inside/outside the outline, we first find the intersections between the outline and our sampling lines. Since the lines are axis aligned, the process is simple and can be implemented efficiently. Given a set of y-axis aligned lines, L_i, at x positions x_i, $i = 1, \ldots, N_L$, and the outline curve $O(t) = (o_x(t), o_y(t))$, the t values at the intersections would be the solutions to $o_x(t) = x_i$. Once the intersections between the outline and the lines are known, the lines are clipped to the segments inside the outline. Then, pixels (represented by their center point) are sampled along these internal segments. Using this optimization to determine if a sample point (pixel) is inside or outside the current slice saves us from unnecessarily attempting to solve Equation (3) for points outside the current slice.

(2) The solver presented in [11] is general and aims to find all solutions, globally, which is computationally demanding. In our case, however, it is already known that a unique solution exists if the Euclidean point is inside (the outline of) V (i.e. Lemma 1). This fact allows us to optimize our use of [11]. The optimization relies on the following procedure: once we use the full solver to get the solution for a point, the solution for any other close-by point can be attempted by employing a numeric tracing (i.e. a Newton iteration method), using the previous solution for any close-by point as an initial guess. By Lemma 1, if this numeric tracing converges to a solution, then this is the only valid solution. If the Newton method fails to converge, we simply invoke our full solver [11] and find the solution. In practice, a single invocation of the full solver might be sufficient for a whole area enclosed by one outline. In our experimental results, we will see that, indeed, in practice, the full solver has to be used very rarely.

Our algorithms calculate generic (device independent) material information, with a mix of materials in a single point/voxel. This kind of information may not be used as-is by certain printers. For example, many printers (including the Connex we're using [8]) only accept a single material for each point/voxel. To convert the information to a form acceptable to the printer, we apply the well known error diffusion dithering algorithm [9] on the current (2D) slice to perform this conversion. More advanced dithering algorithms, such as [6], may also be used, in this stage, if needed.

Algorithm 1 outlines the entire process of generating the required input for the printer, for a single slice. Lines 1 to 3 of Algorithm 1 perform some preliminary calculations, such as determining the bounding rectangle of the printed slice, and the size of the needed bitmaps. The loop in lines 4 to 21, traverses over all $FGM_{cell(i)}$ in the input FGM object, and fills the correct pixels in the matrix M_g, for each cell. Inside the cell loop, we calculate the outline of the current cell (line 6), and then run a loop on each pixel (in lines 7 to 20) calculating the (u, v, w) value for each pixel. M is evaluated, in line 17, with those parameters to find the

Algorithm 1 GenerateSliceBitmaps

Input:

(1) $FGM_{obj} = \{FGM_{cell(1)}, \ldots, FGM_{cell(n)}\}$, a complex of n FGM cells;

(2) m, the number of materials;

(3) z, the height of the slice;

(4) x_{res}, y_{res}, the resolution (in pixels) of the required bitmap(s);

Output:

(1) $B = \{B_1, \ldots, B_m\}$, a set of bitmaps;

Algorithm:

1: $(x_{min}, x_{max}, y_{min}, y_{max}) := BoundingRectangleXY(FGM)$;
2: $(dx, dy) := (\frac{x_{max}-x_{min}}{x_{res}}, \frac{y_{max}-y_{min}}{y_{res}})$;
3: $M_g := EmptyGrid(x_{res}, y_{res})$; // **A** 2D matrix of vectors.
4: **for** $i = 1; i \leq n; i = i + 1;$ **do** // Iterate over all the cells.
5: $(V, M, S) := FGM_{cell(i)}$;
6: $\mathcal{O} := SSI(\mathcal{S}, Plane(z))$; // Surfaces-plane outline intersection.
7: **for** $x = x_{min}; x \leq x_{max}; x = x + dx$ **do**
8: $\mathcal{Y}$:= Intersection($Line$(X = x), $\mathcal{O}$); // A set of y-aligned intervals.
9: **for all** $Y \in \mathcal{Y}$ **do**
10: $(u, v, w) := FullSolve(V, x, \min_y(Y), z)$;
11: **for** $y = \min_y(Y); y \leq \max_y(Y); y = y + dy$ **do**
12: $(u, v, w) := NumericSolve(V, x, y, z, u, v, w)$;
13: **if** $V(u, v, w) \neq (x, y, z)$ **then**
14: $(u, v, w) := FullSolve(V, x, y, z)$;
15: **end if**
16: $(x_i, y_i) = \left(floor\left(x_{res}\frac{x-x_{min}}{x_{max}-x_{min}}\right), floor\left(y_{res}\frac{y-y_{min}}{y_{max}-y_{min}}\right)\right)$;
17: $M_g(x_i, y_i) := M(u, v, w)$;
18: **end for**
19: **end for**
20: **end for**
21: **end for**
22: $M_{dither} := Dither(M_g)$;
23: $B := BitmapPerMaterial(M_{dither}, m)$;
24: Return B;

material information and save it in the matrix M_g. The calculation of (u, v, w) happens as described before. We first try and use the fast numeric tracing method ($NumericSolve$) to find the correct unique solution to Equation (3) in line 12, and if that fails, we resort in line 14 to $FullSolve$: the full, slower, solution of Equation (3), using the solver from [11].

Lines 22 to 23 perform the final steps, transforming the material information produced, into a bitmap for each material as needed by the printer we use. These steps take M_g with fractional material information, transform it, through dithering, into an image with one material per pixel, and then split the image into one binary bitmap per material. The trimmed trivariate volumes we use (from [16]) inherently support the modeling of any volume as geometrically mutually exclusive cells. This allows us to use a naive procedure that just chooses the last cell with non-empty material information for a pixel (as aside from along the

boundaries, there should be exactly one such cell), and sets it as the material information for that pixel in M_g. Finally, one should note that, in our case, the support structure is generated automatically by the printer.

4 Experimental Results

To test the methods outlined in Section 3 we have implemented them as a C/C++ program. We ran our tests on an Intel i7-4770 3.4 GHz windows 7 machine.

We start by examining the benefits of using numeric tracing instead of invoking the full solver as described before. To illustrate the effects of this optimization, we ran tests on the trivariate volume shown in Figure 3 (a). The results of our tests are shown in Table 1. We examine four different optimization methods for using numeric tracing. None, meaning no optimization was used. Previous point tracing, where only the previous pixel in the line is used as an initial value for tracing. Previous line tracing, where either the previous pixel in the line or the same pixel on the previous line are used as an initial value for tracing. Full tracing, where any adjacent pixel on the current or previous lines (including diagonally placed pixels) are used as an initial value for tracing. We examined two metrics in these tests: the time to calculate (u, v, w) values for all the pixels in the slice, and the percentage for which we invoked the full solver. As the results in Table 1 show, using numeric tracing to avoid the use of the full solver brings about a great improvement in overall run time. Using the previous point optimization results (as expect) in one full invocation of the full solver per line. Allowing the use of the pixel from the previous line improves the results. The results show that when the optimization uses the results of any nearby pixels, the full solver needs to be invoked only a negligible number of times compared to the total number of pixels.

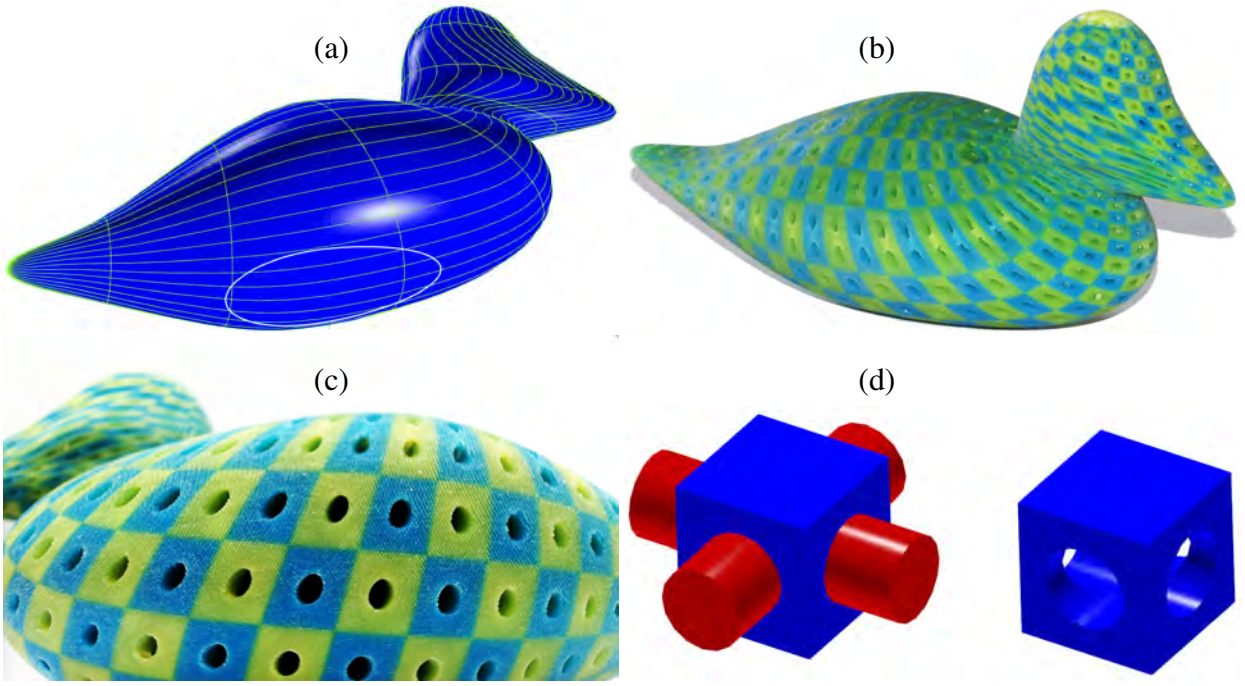

Figure 3: *Image (a) shows a trivariate volume shaped like a duck. The outline of one slice, is highlighted. Image (b) presents a fabricated graded-materials duck object based on the model in (a). In (c) the fabricated model is seen from a different view showing more details of the micro-structure. The Image in (d) shows the micro-structure tile model that was created by subtracting two cylinders from a cube and alternating between green and blue colors. Printed courtesy of Stratasys Ltd.*

Figure 4 : *A fabricated object with graded materials based on a volumetric model of the Utah teapot. Printed courtesy of Stratasys Ltd.*

Table 1 : *Statistics on the use of the full solver, for different optimizations, in the slice highlighted in Figure 3.*

Optimization	Time [sec]	Full Solve [%]
none	22590	100
previous point tracing	71	0.2532
previous line tracing	18	0.0205
full tracing	15	0.0044

Figures 1, 3, and 4 show fabricated objects that were created using the methods outlined in this paper. These objects were printed on a Stratasys J750 printer with their Voxel Print interface [26], allowing voxel-level control. The printer xy resolution was 600x300 DPI (each pixel is 0.0423 by 0.0846 millimeters), the resolution in the z axis is 0.03 millimeters per slice. As the focus of this work is the fabrication of V-reps rather than the actual design process of FGM objects, these models were created mostly with appearance in mind, to showcase what V-reps can offer in terms of design freedom. While the examples given here use only procedural or discrete textures, the results of tools such as iso-geometric analysis ([7]) could also be applied to V-reps and fabricated as FGM objects using our methods.

Figure 1 shows an object created as a volumetric model of a chalice. The model is made of two order $(4, 4, 4)$ piecewise-polynomial trivariates, and one order $(3, 4, 2)$ piecewise-rational trivariate (that result in five volumetric cells in the final model). The knot portion can be described by the sweep of a disc along a knot shaped curve, while the cup and base can be described using the rotation of surfaces. The model was sliced along its length (as in Figure 2). Material composition was again determined procedurally with most of the volume printed in a transparent material. The twisting colored curves inserted into the model were defined in the parametric domain (of all five trimmed trivariates in this model). For each $w = w_0$ plane in the parametric domain of $V(u, v, w)$, several points (each with its own color) were specified. In each plane, the points are also rotated by an angle (or shifted a certain distance) dependent on the w parameter. When evaluating the material composition, any location whose distance in the u, v plane is less than some defined distance from one of the points is colored in the point color, otherwise the transparent material is used. The embedding of these lines demonstrates how easy it is to relate material information to the geometry given

the common parametrization. Without this shared parametrization, using voxels or a B-rep representation for example, applying a volumetric texture that would follow the geometry would require a much more complex procedure.

Figure 4 shows an object created as a volumetric model of the Utah teapot. The model is made of two order $(4, 4, 2)$ and one $(4, 4, 4)$ piecewise-polynomial trivariates (that result in six disjoint volumetric cells in the final model, two from the intersection of the handle and the body, one from the intersection of the spout and the body, and three from the intersection-free volumes of the original three trivariates). The material composition for the body of the teapot is determined by a 2D texture that was applied to the entire thickness of the hull (using only the u, v parts of the parametrization). The spout and handle are made from a rubber like transparent material, and have 3D twisting colored curves embedded within, similar to the ones used in the chalice model. Both of these features demonstrate how easy it is to reuse and apply the same textures to different V-reps, just like 2D textures may be easily applied to different surfaces in traditional 2D texture mapping (as used in computer graphics).

Figure 3 is made of a single order $(3, 3, 4)$ piecewise-polynomial trivariate. Figure 3 shows how the original geometry can be modified by specifying locations where no material is present, allowing the fabrication of porous geometry that is made of heterogeneous materials. The same approach can be used to apply any sort of porous texture to the volumes of objects. In this case, we procedurally defined the cavities within the trivariate model of a duck by first dividing the model's cubic parametric domain into smaller boxes or tiles. Each tile is a box from which two orthogonal cylinders are subtracted, creating the void (see Figure 3 (d)). The coloring in the tiles is alternated between green and blue based on the index of the tile within the overall parametric domain.

The time needed to generate the bitmaps for these models generally depends on the volume of the geometry of the model, or more specifically on the number of printer voxels inside the the model. Generating the bitmaps for the object in Figure 3 with its about 2.69 billion voxels takes about 30 hours. However, as can be seen in Algorithm 1, the bitmaps for each slice can be calculated independently, without relying on the bitmaps for other slices. This means that bitmap generation for multiple slices can be run in parallel. Generating the bitmaps for the same model in Figure 3, using four concurrent processes, takes only about 8.5 hours, about 28 percent of the time needed for a single process. Table 2 presents some statistics for the presented objects (identified by their figure number). The results show that, overall, the full solver needed to be invoked very few times. Additionally actual failures of the numeric tracing method were virtually non-existent in our experiments. The times mostly increase when the number of model voxels increases, except for the chalice model, that is significantly more complex (in terms of the underlying B-spline function) than the others. All these times include writing the bitmaps to the disk, and are for four processes working concurrently. The three printed models can also be seen in `https://youtu.be/mQC5rfk-jfI`.

Table 2 : *Statistics for the models in our fabricated examples.*

Figure	Bitmap Size	# Slices	# Voxels Total, and Inside Model [$\times 10^9$]	Average Time Per Slice (4 Threads) [sec]	Total Time (4 Threads) [hour]	Full Solve [%]
1, 2	(2944 x 1466)	1914	8.2, 0.45	5.75	3.05	0.1033
4	(2976 x 1477)	2110	9.2, 0.54	3.14	1.84	0.1008
3	(3040 x 1512)	2918	13.4, 2.69	10.4	8.5	0.0055

5 Conclusions and Future Work

In this paper, we presented how trimmed V-reps can be used to model and fabricate FGM objects. We hope that the ability to fabricate FGM objects directly from a V-rep model, will motivate the creation of advanced CAD tools and design and analysis environments to support V-rep modeling. V-reps are natural candidates

for augmenting existing B-reps in CAD design. V-reps offer the same modeling strengths that made B-reps so dominant for several decades as well as support new modeling challenges like FGM object modeling, and analysis using the shared geometric/material function space.

Throughout this work, we encountered several subjects that may warrant further investigation: We designed the algorithm presented here to work with printers that require bitmaps as input. However, the principles we based our algorithm on can be applied to many other required printer inputs. As long as the required material information can be reduced to a discrete number of samples, which is a reasonable assumption for any printer, our algorithm can be easily adapted to produce that information.

In our implementation, we have chosen to sample pixels along lines in a similar manner to a raster-scan. It is possible an alternate sampling strategy, similar to a flood-fill, would yield good, and maybe even better, results. Improved results are likely, as such a strategy would be able to identify neighboring pixels more robustly than our current scan conversion strategy. This would allow the employment of numeric tracing in even more cases, with a more efficient solution. In addition, using results from a previous layer as initial values for tracing can possibly further improve results, potentially making it possible to employ one full solver solution, for the entire model.

6 Acknowledgments

We thank all the reviewers for their valuable input. This research was supported in part by the ISRAEL SCIENCE FOUNDATION (grant No.278/13). This work was also supported in part by DARPA, under contract HR0011-17-2-0028. All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies, nor should any endorsement be inferred.

References

[1] 3D Systems, Inc. ProJet MJP 5500X. `https://www.3dsystems.com/3d-printers/projet-mjp-5500x`, 2017.

[2] ISO ASTM. Astm52915-13, standard specification for additive manufacturing file format (amf) version 1.1. *ASTM International, West Conshohocken, PA*, 52915:2013, 2013.

[3] Christoph Bader, Dominik Kolb, James C Weaver, and Neri Oxman. Data-driven material modeling with functional advection for 3d printing of materially heterogeneous objects. *3D Printing and Additive Manufacturing*, 3(2):71–79, 2016.

[4] Victor Birman and Larry W Byrd. Modeling and analysis of functionally graded materials and structures. *Applied mechanics reviews*, 60(5):195–216, 2007.

[5] Desai Chen, David IW Levin, Piotr Didyk, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Spec2fab: a reducer-tuner model for translating specifications to 3d prints. *ACM Transactions on Graphics (TOG)*, 32(4):135, 2013.

[6] Wonjoon Cho, Emanuel M Sachs, Nicholas M Patrikalakis, and Donald E Troxel. A dithering algorithm for local composition control with three-dimensional printing. *Computer-aided design*, 35(9):851–867, 2003.

[7] J Austin Cottrell, Thomas JR Hughes, and Yuri Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[8] EL Doubrovski, EY Tsai, D Dikovsky, JMP Geraedts, Hugh Herr, and Neri Oxman. Voxel-based fabrication through material property mapping: A design method for bitmap printing. *Computer-Aided Design*, 60:3–13, 2015.

[9] Robert W Floyd. An adaptive algorithm for spatial gray-scale. In *Proc. Soc. Inf. Disp.*, volume 17, pages 75–77. SID, 1976.

[10] James D Foley, Andries Van Dam, et al. *Fundamentals of interactive computer graphics*, volume 2. Addison-Wesley Reading, MA, 1982.

[11] Iddo Hanniel and Gershon Elber. Subdivision termination criteria in subdivision multivariate solvers using dual hyperplanes representations. *Computer-Aided Design*, 39(5):369–378, 2007.

[12] HP Development Company, L.P. Jet Fusion. `http://www8.hp.com/us/en/printers/3d-printers.html`, 2017.

[13] TR Jackson, H_ Liu, NM Patrikalakis, EM Sachs, and MJ Cima. Modeling and designing functionally graded material components for fabrication with local composition control. *Materials & Design*, 20(2):63–75, 1999.

[14] XY Kou and ST Tan. A hierarchical representation for heterogeneous object modeling. *Computer-Aided Design*, 37(3):307–319, 2005.

[15] XY Kou and ST Tan. Heterogeneous object modeling: A review. *Computer-Aided Design*, 39(4):284–301, 2007.

[16] Fady Massarwi and Gershon Elber. A b-spline based framework for volumetric object modeling. *Computer-Aided Design*, 78:36 – 47, 2016. SPM 2016.

[17] Panagiotis Michalatos and Andrew O. Payne. Project monolith. `http://www.monolith.zone/`, 2017.

[18] Pulak Mohan Pandey, N Venkata Reddy, and Sanjay G Dhande. Slicing procedures in layered manufacturing: a review. *Rapid prototyping journal*, 9(5):274–288, 2003.

[19] Mosaic Manufacturing Ltd. Series enabled extrusion of materials. `https://www.mosaicmanufacturing.com/pages/technology`, 2016.

[20] Pierre Muller, Pascal Mognol, and Jean-Yves Hascoet. Modeling and control of a direct laser powder deposition process for functionally graded materials (fgm) parts manufacturing. *journal of materials processing technology*, 213(5):685–692, 2013.

[21] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG)*, 32(3):27, 2013.

[22] J.S. Page. Material dispensing system, April 2 2015. US Patent App. 14/501,896.

[23] Xiaoping Qian and Debasish Dutta. Feature-based design for heterogeneous objects. *Computer-Aided Design*, 36(12):1263–1278, 2004.

[24] Yuhi Sasaki, Masahito Takezawa, Seungki Kim, Hiroshi Kawaharada, and Takashi Maekawa. Adaptive direct slicing of volumetric attribute data represented by trivariate b-spline functions. *The International Journal of Advanced Manufacturing Technology*, pages 1–17, 2016.

[25] Ki-Hoon Shin and Debasish Dutta. Constructive representation of heterogeneous objects. *Journal of Computing and Information Science in Engineering*, 1(3):205–217, 2001.

[26] Stratasys Ltd. Voxel print program. `http://www.stratasys.com/industries/education/research`, 2016.

[27] Kiril Vidimče, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. Openfab: a programmable pipeline for multi-material fabrication. *ACM Transactions on Graphics (TOG)*, 32(4):136, 2013.

[28] Bing-Quan Zuo, Zheng-Dong Huang, Yan-Wei Wang, and Zi-Jun Wu. Isogeometric analysis for csg models. *Computer Methods in Applied Mechanics and Engineering*, 285:102–124, 2015.