Additive Manufacturing Impacts on CAD

George Allen
Siemens PLM Software
Outline

- Additive manufacturing background

- Contemporary CAD background

- Problems and possible solutions
  - Graded materials
  - Lattice structures

- Emerging data standards
Why Is Additive Manufacturing Important?

- It allows you to place material of the desired composition exactly where you want it (and nowhere else).

- You can make things that are impossible to make in any other way:
  - Internal voids, webs, honeycombs, lattice structures
  - Assemblies of parts (in one shot)
  - Internally embedded components
  - Parts with custom non-homogeneous (graded) materials

- Material composition and placement become design variables.

- Engineering performance of parts can be dramatically improved.

- The “catalyst for the next industrial revolution” ???
Some Amazing Things ... 

The next industrial revolution ?? Probably not (IMO), but, some amazing possibilities
But There Is A Problem ...

- To print something, you have to design it, which requires effort, creativity, and software.
- Today’s typical solutions: scanning or downloading models from online libraries.
- Result: lots of rabbits.
Current 3D Printing Process

1. **CAD model**
2. **Facets (STL)**
3. **Slices**
   - Compute slices
4. **Filling**
   - Each slice is “filled”
5. **G-codes**
Many different CAD systems are used. The system just needs to be able to export STL files. Popular ones with hobbyists are Autodesk 123D, OpenSCAD, Blender, Sketchup, etc. These are all free.

The boundary files are almost always STL files nowadays, which is the source of many problems.

Cleanup/repair often takes significant time and effort. Tools are MeshLab and netfabb. Netfabb also does slicing.

Common slicing/filling apps are Slic3r, Skeinforge, KISSlicer. Slic3r also does some cleanup. Supported formats are:
- Slic3r: Imports STL, OBJ, or AMF
- Skeinforge: Imports STL, GTS (Gnu Triangulated Surfaces), OBJ Wavefront 3D OBJ, SVG, etc.
- KISSlicer: Imports STL

G-codes are vector-based “move/draw” and machine control commands, as used in NC milling. To check before printing, there are g-code viewer programs like Pleasant3D (Mac) and a g-code viewer add-on for Blender. Also, Repetier-Host has a built-in g-code viewer and editor.

Programs like Printrun/Pronterface, Repetier-Host, ReplicatorG send g-codes to the printer. These programs can also invoke a slicing/filling program, too.
Outline

- Additive manufacturing background
  - Contemporary CAD background
- Problems and possible solutions
  - Graded materials
  - Lattice structures
- Emerging data standards
Typical Geometric Objects

**SURFACES**
- Plane
- Circular cylinder
- Circular cone
- Sphere
- Torus
- NURB surface
- Surface of revolution
- Extruded surface
- Offset surface [1]
- Rolling ball fillet surface [1]
- Foreign surface [2]

**CURVES**
- Line
- Circle
- Ellipse
- NURB curve
- Parameter-space curve
- Intersection curve [1]
- Foreign curve [2]

**MISSING**
- Subdivision surfaces
- Triangular surfaces
- General implicit forms

---

[1] These are precise “procedural” objects, not spline approximations
[2] Defined by user-supplied “parametric evaluator” functions – rarely used

**ISO 14306**
B-Rep Topological Entities

- Standard, conventional boundary representation
- “Fin” = co-edge, or edge-use
Body Types: Solid, Sheet, General

- **Solid body**
- **Sheet body**
- **Cellular body** -- regions with common faces
- **Wire body**
- **Non-manifold body**
- **Mixed dimension body**
History Mechanism

- Parent-child relations stored in a “tree” structure representing design process ( “history”)
- Explicit b-rep is stored for each body, and is the primary input for applications
- General procedural rep mechanism – not just for features and b-reps
Material Properties in CAx Systems

- A material is a property of a body or region
- So, regions assumed to be homogeneous
- Different apps have different material definitions

<table>
<thead>
<tr>
<th>Application</th>
<th>Usage. Material determines ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td>Color, reflectivity, etc.</td>
</tr>
<tr>
<td>Drafting</td>
<td>Hatch patterns</td>
</tr>
<tr>
<td>Mass props</td>
<td>Density</td>
</tr>
<tr>
<td>CAE</td>
<td>Young’s modulus, etc.</td>
</tr>
<tr>
<td>CAM</td>
<td>Allowable feedrates</td>
</tr>
<tr>
<td>Other</td>
<td>Cost estimation</td>
</tr>
<tr>
<td></td>
<td>Recyclability analysis, etc.</td>
</tr>
</tbody>
</table>
Fields in NX CAE

- A “field” is a general real or vector-valued mapping defined on a domain in $\mathbb{R}^2$ or $\mathbb{R}^3$

- Two types:
  - Formula fields – defined by NX expressions
  - Table fields – defined via (interpolated) tabular data

- Provides various types of interpolation, including an “inverse squared distance” type

- Already able to create a new (CAE) material with spatially-varying properties defined by a field
Outline

- Additive manufacturing background
- Contemporary CAD background
- Problems and possible solutions
  - Graded materials
  - Lattice structures
- Emerging data standards
New Characteristics of AM Parts

- Objects designed for AM have some new aspects to them:
  - Multiple “chunks” consisting of different materials
  - Graded materials – where material composition varies as a function of location
  - Fine-grained lattice structures

- We need representations of these things, and our applications have to be modified to handle them
Multiple Chunks of Material in a Single Body

- Easy: use Parasolid “cellular body” capability
- A cellular body has internal faces that partition it into several regions
- A different material could be assigned to each region
- Cellular bodies not supported fully, but maybe enough.
- Possible workaround – just use an “assembly” of touching bodies

Metal region embedded within a plastic region

Figure 16-9 General body partitioned by an internal face
Outline

- Additive manufacturing background
- Contemporary CAD background
- Problems and possible solutions
  - Graded materials
  - Lattice structures
- Emerging data standards
Graded Materials

- By mixing materials at the droplet level, we can essentially create new materials from old ones.
- If the material composition varies from one place to another, the material is said to be “graded.”
- Material composition becomes a design variable.
- Problem: predictability of properties??
Representing Graded Materials

- NX “field” concept seems like a good starting point
- For example, for two materials, need a function $f:\ [0,1] \times [0,1] \times [0,1] \to [0,1]$

Here $f(x, y, z)$ describes fractional material composition at point $(x, y, z)$. Fraction of other material is just $1 - f$.

- User interface ideas
  - Powerpoint “gradient” idea
  - NX expressions (formulas)
  - Values on a rectangular grid (to be interpolated)
  - Values on certain base sets, plus transfinite interpolation based on distances (Vadim’s idea)

- Enhance expression subsystem with distance functions, especially a distance-to-boundary function
- Maybe compute and store a b-spline approximation of $f$, for performance reasons.
Experiment in Current NX

Field function: $f(x, y, z) = \text{distance from } (x, y, z) \text{ to boundary of shape}$
Impact on Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Changes needed to handle graded materials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td>No special display needed most of the time? Use CAE trick – colored spheres? Use 3D texture?</td>
</tr>
<tr>
<td>Drafting</td>
<td>Don’t need to handle graded materials in cross-hatching, IMO</td>
</tr>
<tr>
<td>Mass props</td>
<td>Need completely new algorithm, it seems to me. See below.</td>
</tr>
<tr>
<td>CAE</td>
<td>Looks like they already know how to handle graded materials</td>
</tr>
<tr>
<td>CAM</td>
<td>Feedrate computations have to be modified to consider material at current point?</td>
</tr>
<tr>
<td>Data exchange</td>
<td>Status in STEP not clear. Some support in AMF; nothing in 3MF.</td>
</tr>
</tbody>
</table>

Mass properties calculation. Essentially converts body into voxels. Embarrassingly parallel.

```plaintext
Body bounding box = [minX, maxX] x [minY, maxY] x [minZ, maxZ];
dx = (maxX - minX)/n  ;  dy = (maxY - minY)/n  ;  dz = (maxZ - minZ)/n  ;
mass = 0;
For (i,j,k) = (0,0,0) to (n,n,n)
    x = minX + i*dx  ;  y = minY + j*dy  ;  z = minZ + k*dZ  ;
    if (x,y,z) is inside the body then
        mass = mass + density(x,y,z)*dx*dy*dz;
    End if
End For
```
Outline

- Additive manufacturing background
- Contemporary CAD background
- Problems and possible solutions
  - Graded materials
    - Lattice structures
- Emerging data standards
Examples of Lattices

Coarse -- No Problem
Just use curvy b-reps, same as today.
Everything works

Medium – Big Problem
A lattice with a few billion faces.
Too fine to use a b-rep, but too coarse to be considered homogeneous.
Probably need some new approach.

Very fine – Smaller Problem
The lattice structure is so fine that the material can essentially be considered homogeneous. Many things just work. Mfg still a problem.
Lattice Definition Today

- Today, fill patterns are specified at the slicing/filling step of the process
- User chooses from menu of patterns
- Many exotic options like hilbertcurve, octagramspiral, Archimedean chords, etc, etc.
- Architecturally, this seems wrong. The fill pattern is unknown to the CAD system, so it can’t be used in CAD functions like mass properties analysis or stress analysis.
Lattice Data Size Problems -- $10^9$ cells

- Number of cells in a lattice can be very very large:
  - $1 \text{ m}^3$ part with cell size of $1 \text{ mm}^3 \rightarrow 10^9$ cells
  - 1 byte per cell $\rightarrow$ 1 gigabyte of data

- Problems if we try to create an explicit (evaluated) representation of the entire lattice, like a facetted or curvy b-rep, or a voxel rep

- Idea: use small procedural representations, and generate explicit reps lazily, as needed (and locally).

- Procedural reps with size/shape parameters given by simple functions (NX expressions)

- Possible only if lattice has some regularity

- Many different types of procedural reps are possible.
  - Periodic implicit surfaces (Pasko and others)
  - Parameterized rod/ball networks
Idea: Parameterized Rod-Ball Networks

- **Topology:**
  - User chooses connectivity pattern from menu of standard ones, or
  - Explicit connectivity info specified

- **Geometry:**
  - Ball and rod diameters, and ball centers
  - Either constant or varying according to simple functions of coordinates or distance from boundary
  - Rectangular, cylindrical, spherical, or curvilinear coordinate systems
  - Trimmed to the outer boundary of the body (or an offset of the boundary)

- Possible refinements
  - Junctions filleted
  - Tapered rods

- Limited generality/flexibility, but maybe good enough, for now.
Two Stages of Evaluation

- For irregular lattice, with no parametric description, need to store skeleton
- Parameters might be given by formulae, rather than constants

**Parameters**
Type = regular rectangular
Repeats: nX = 4; nY = 2
Spacing: dX = 5; dY = 10
Ball_diameters = 2
Rod_diameters = 1

**Skeleton**
- Ball centers
- Ball connectivity
- Ball_diameters = 2
- Rod_diameters = 1

**Explicit Reps**
- Curved faces
- Facets
- Voxels

**Evaluate**
Parameters $\rightarrow$ skeleton

**Evaluate**
Skeleton $\rightarrow$ explicit reps
Clipping Lattice To Body Boundary

- Lattice will typically be “clipped” using the boundary of the body (or an inward offset of the boundary)
- We have to represent this “combined” body consisting of the outer shell and the interior lattice
- Possibly the clipping operation can be left in unevaluated form
Another Approach – Implicit Surface Reps

- Another procedural approach
- Sometimes called F-reps
- See many papers by Alexander Pasko
- Object = \(((x, y, z) \in \mathbb{R}^3: g(x, y, z) \leq 0)\), where \(g\) is some suitable function
- Start with three arrays of slabs
  \[
  s_x(x, y, z) = \sin(q_x x + p_x) - l_x \\
  s_y(x, y, z) = \sin(q_y y + p_y) - l_y \\
  s_z(x, y, z) = \sin(q_z z + p_z) - l_z
  \]
- Bars = intersections of slabs
  \[
  r_x(x, y, z) = s_y \land_s s_z \\
  r_y(x, y, z) = s_x \land_s s_z \\
  r_z(x, y, z) = s_x \land_s s_y
  \]
- Lattice = union of bars
  \[
  g(x, y, z) = r_x \lor_s r_y \lor_s r_z
  \]

Blends can be built into the “combine” function
Regularity – More Examples
Taxonomy of Irregularity

- Topology
  - Regular (repeating)
  - Irregular

- Sizing
  - Constant
  - Procedural
  - Irregular

- Spacing/positioning
  - Constant
  - Procedural
  - Irregular

- Reduced regularity ⇒ increased explicit storage
- Problem: optimization introduces irregularity
Spectrum of Lattice Representations

Number of cells

Many ($10^{12}$)

Few ($10^4$)

Regular

Irregular

Ignore cell geometry (too small to be important)

Procedural representations (evaluated lazily)

Current explicit boundary representations

Explicit skeletal reps

Regularity

Spectral analysis of lattice representations highlighting regular and irregular geometries, with consideration for the number of cells and their geometrical complexity.
# Operations on Rod-Ball Networks

<table>
<thead>
<tr>
<th>Application</th>
<th>Possible Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td>Generate pixels directly from procedural reps (no b-rep, no tessellation)</td>
</tr>
<tr>
<td>Compute volume</td>
<td>(1) Facet-based: add up pyramid volumes whose bases are facets (like today) &lt;br&gt; (2) Voxel-based: add up voxels that are inside rods/balls and inside body, or&lt;br&gt; In both cases we cycle through a very large set of items (voxels or facets).&lt;br&gt; Computations are very simple and highly parallel. Maybe use GPU.</td>
</tr>
<tr>
<td>Structural</td>
<td>Rods become 1D rod or beam elements in FEM. Easy, except for huge number.</td>
</tr>
<tr>
<td>3D printing</td>
<td>On each slicing plane ...&lt;br&gt; (1) Facet-based: Tesselate, slice, fill.&lt;br&gt; (2) B-rep based: Generate b-rep, slice it, and fill.&lt;br&gt; (3) Voxel-based: Generate a “material bitmap”. Only a few printers can consume.&lt;br&gt; Output list of g-codes will be enormous. Stream to printer?</td>
</tr>
<tr>
<td>Data export</td>
<td>We could export b-reps, facets, or voxels, or g-codes. But these would all be huge.&lt;br&gt; No way to import/export procedural reps, as far as I know.</td>
</tr>
</tbody>
</table>
Outline

- Additive manufacturing background
- Contemporary CAD background
- Problems and possible solutions
  - Graded materials
  - Lattice structures
- Emerging data standards
STL Problems and Solutions

• Almost all 3D printing today relies on STL data.
• But the STL format has many problems:
  • Verbosity
  • No connectivity (topology) information.
  • No units, color, or material information
• Two new file formats: AMF and 3MF

AMF Format
• Mostly designed by Hod Lipson from Cornell
• ISO standard ??? ISO / ASTM52915 – 13
• XML text format, then (optionally) zipped
• Explicit mesh topology
• Colors for material, volume, vertex, or triangle
• Graded materials defined by composition functions
• A “constellation” is an array of copies
• Curved triangles, optionally

3MF Format
• Invented by Microsoft, November 2013
• Zipped XML, like MS Office docs
• Triangle mesh or as set of “slices”
• Slices consist of strings of curves
• Uses tiled 3D texture for regular lattice
• Rectangular lattices, but not cylindrical
• No graded materials?
• Integrated into Win8.1 print pipeline
Graded Materials in the AMF Format

```xml
<material id="1">
    <metadata type="Name">StiffMaterial</metadata>
</material>

A basic material

<material id="2">
    <metadata type="Name">FlexibleMaterial</metadata>
</material>

Another basic material

<material id="3">
    <metadata type="Name">MediumMaterial</metadata>
    <composite materialid="1">0.4</composite>
    <composite materialid="2">0.6</composite>
</material>

A homogeneous mix of the first two materials, defined using the “composite” keyword

<material id="4">
    <metadata type="Name">VerticallyGraded</metadata>
    <composite materialid="1">z</composite>
    <composite materialid="2">10-z</composite>
</material>

A non-homogeneous (graded) mix where the composition varies as a function of z

<material id="5">
    <metadata type="Name">Checkerboard</metadata>
    <composite materialid="1">
        floor(x+y+z%1)+0.5
    </composite>
    <composite materialid="2">
        1-floor(x+y+z%1)+0.5
    </composite>
</material>

Another non-homogeneous mix using more complex functions of x, y, z. This produces a “checkerboard” mix pattern.
New vs. Old Printing Workflows

Old Way (Today)

New Way (MS proposal)

Benefits of New Way:
- Simpler for user: slicing/filling/sending all hidden inside application’s “Print” function and printer driver
- Clear separation: only the driver knows about the printer; upstream processes are device-independent
- Familiarity: leverages user’s knowledge of existing 2D printing infrastructure (discovery, spooling, etc.)
- But, making this work might require a hugely complex “Print” dialog
Microsoft 3D Builder App

- Imports STL, OBJ, or 3MF files (not AMF).
- Allows simple viewing and transformation operations.
- Can print using standard print workflow, but only one or two printers have drivers, currently.
- I don’t see the point of this – printing should be done directly from the authoring app??
Summary

- AM technology has great potential (beyond the buzz). Won’t go the way of AI.
  - Parts with more complex shapes
  - Designs with better performance (e.g. strength/weight ratio)
- Production parts already, not just prototypes. Industry, not hobbyists.
- But many parts don’t really need optimal performance
- Design tools are weak in current (mainstream) CAD systems. Niche tools exist.
- Significant changes needed in basic representations. Boundary reps are not enough. But I think we will supplement them, not replace them
- Changes need to be done carefully
  - System-wide representations supporting many apps
  - No second chances
- Leading CAD vendors till in learning mode -- just getting started
  - Many apps have a “print” function. MS moves will encourage this.
  - Basic representation changes will take a while
Thank you
Questions?
Comments?