Algorithms for 3D Printing and Other Manufacturing Processes

Introduction

Dan Halperin

School of Computer Science

Tel Aviv University

Spring 2017

3D Printing

3D printing, also known as **additive manufacturing** (**AM**), refers to processes used to synthesize a three-dimensional object[1] in which successive layers of material are formed under computer control to create an object.[2] Objects can be of almost any shape or geometry and are produced using digital model data from a 3D model or another electronic data source such as an Additive Manufacturing File (AMF) file.

3D Printing, videos

- Intro: https://www.youtube.com/watch?v=Ev-MM9cGKiQ
- GE: https://www.youtube.com/watch?v=l0SXlkrmzyw
- SLM (4 lasers): https://www.youtube.com/watch?v=sbPpFHZL_cU

Other manufacturing processes

- CNC machining (subtractive)
- Molding and casting
- Assembly
- Many more …

A typical industrial 3DP cycle

- Obtain digital models
- Fix models
- Orient part model
- |Add support|
- |Nest many part models|
- Print assembly
- Unpack assembly
- Clean parts

About the course

The main veins of the course

- Algorithms – theory
- Algorithms – practice!
- 3DP background, technologies
- Actual printing practice: preparation, slicing, gcode, …
- Open source software: Cura, CGAL, Meshlab, …

Algorithmic topics as time permits

- Part orienting, minimizing height
- Digital surface simplification

Contest I: approximate height minimization

- Movable separability in tight quarters
- Nesting/packing

Contest II: (2D) few-parts multi-copies packing

- Mutual relations between robotics and 3DP

Course mechanics and grade

- 70%
  - Assignments (3-4), including programming and 3D printing
  - Small project with algorithmic component, e.g., object restoration, 3D-printing of an interesting mechanism, contribution to any relevant open source project; modest budget available
- 5%
  - Presentation, 15 mins, relevant topic of student's choice
- 25%
  - Exam (2 hours)

This is the first time the course is given – you can influence its evolution!

Team

- Dan Halperin

danha@post.tau.ac.il
http://acg.cs.tau.ac.il/danhalperin
Schreiber 219
Office hours: Monday, 19:00 – 20:00

- Efi Fogel

efif@post.tau.ac.il
http://acg.cs.tau.ac.il/people/efifogel
Schreiber M18 (floor -1) the robotics lab
Office hours: Monday, 15:00 – 16:00

Actual 3D printing

- Part of the assignments
- In addition each student will have an allowance of free printing
- Each student will get a key to the Ultimaker3 niche on the 2nd floor of Schreiber, just behind the staircase

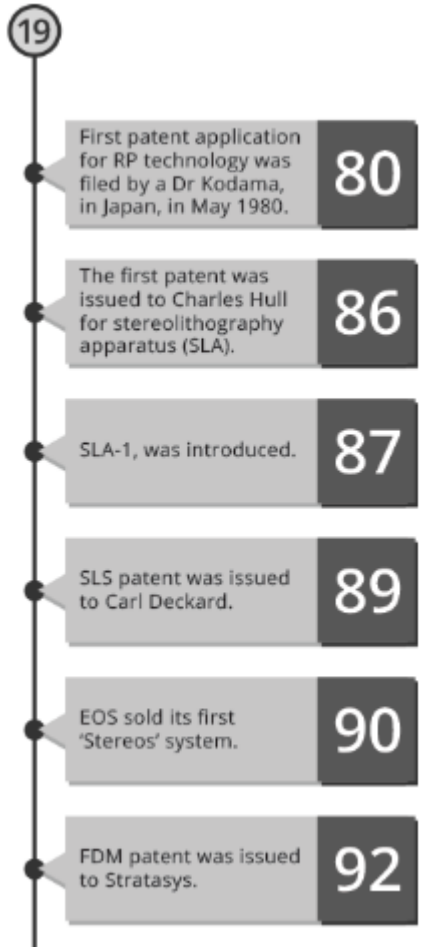
Efi Fogel will instruct and assist with the printing

More background

The (archived) starting point

3dprintingindustry.com/3d-printing-basics-free-beginners-guide/history/

The earliest 3D printing technologies first became visible in the late 1980's, at which time they were called Rapid Prototyping (RP) technologies. This is because the processes were originally conceived as a fast and more cost-effective method for creating prototypes for product development within industry. As an interesting aside, the very first patent application for RP technology was filed by a Dr Kodama, in Japan, in May 1980. Unfortunately for Dr Kodama, the full patent specification was subsequently not filed before the one year deadline after the application, which is particularly disastrous considering that he was a patent lawyer! In real terms, however, the origins of 3D printing can be traced back to 1986, when the first patent was issued for stereolithography apparatus (SLA). This patent belonged to one Charles (Chuck) Hull, who first invented his SLA machine in 1983. Hull went on to co-found 3D Systems Corporation – one of the largest and most prolific organizations operating in the 3D printing sector today.



...

Lingua franca in 3DP

- High level: STL and expansions

v

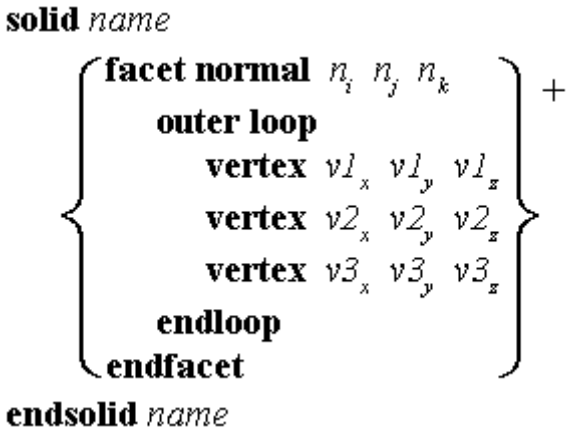
slicer

v

- Low level: gcode and derivatives

STL file format

- http://www.fabbers.com/tech/STL_Format
- https://en.wikipedia.org/wiki/STL_(file_format)



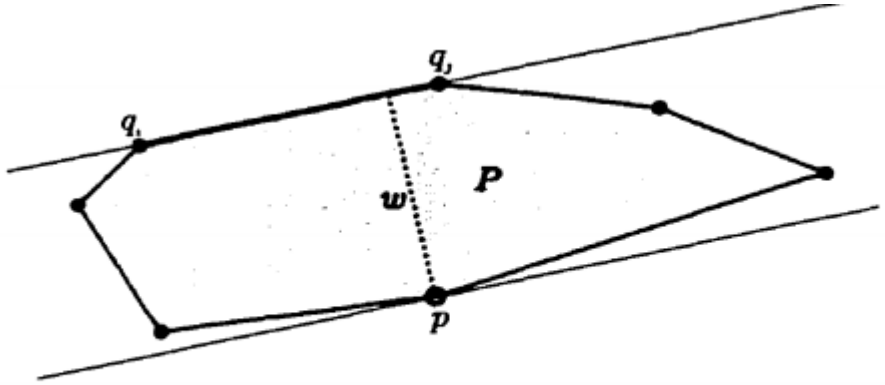
- There are various support tools on the web including STL viewers
- Pay attention whether you use ASCII STL or Binary STL (for very big files ASCII may be too wasteful)

Two simple algorithms

Problem 1: Finding the width of a polygon

- Input: a simple polygon P, given by the vertices in CCW order
- Output: the minimum distance between two parallel supporting lines of P

In addition to the width, we will output the direction normal to the parallels



The width w of P is determined by lines of support through p and q_iq_j.

[H. Pirzadeh]

The outer-normal diagram of a convex polygon

- A mapping between the boundary P of a convex polygon and the unit circle S^1, where a point u in S^1 represents the direction from the center of S^1 to u
- Adaptation of the Gaussian map of smooth surfaces to the case of the boundary of convex polytopes
- Interpretation I: a set-valued function, which assigns to each point p on P the set of outer normals of support lines to P at p
- Def: Extremal vertex of P in direction u
- Interpretation II: the decomposition of S^1 into maximal components in each of which the set of extremal vertices in the corresponding directions is the same
- We will call this diagram the Gaussian map (a slight abuse)

The width

- Directional width $w_Q(u)$ of the polygon Q in direction u, is the distance between the support lines to Q with outer normals u and –u
- Overlaying the G-map with a copy of the G-map rotated by π, gives us the relevant pairs of features
- The width of Q := min $w_Q(u)$ over all directions u

Width, algorithm wrap-up

- Compute the convex hull CH(P)

  O(n)

  (reasonable to do in O(n log n) time in practice)
- Overlay the Gaussian map with a copy rotated by π

  O(n)
- Find the vertex-edge pair that gives the minimal distance

  O(n)

Width, remarks

- In 3DP: reduce height for polyhedral parts
- Part orienting for handing the part to a robot, using the width function; video: the parallel jaw gripper
- Part orienting from stable poses
- The gomboc: a convex mono-monostatic 3D homogeneous body: when resting on a flat surface, has just one stable and one unstable point of equilibrium

Width, further remarks: pure rotations

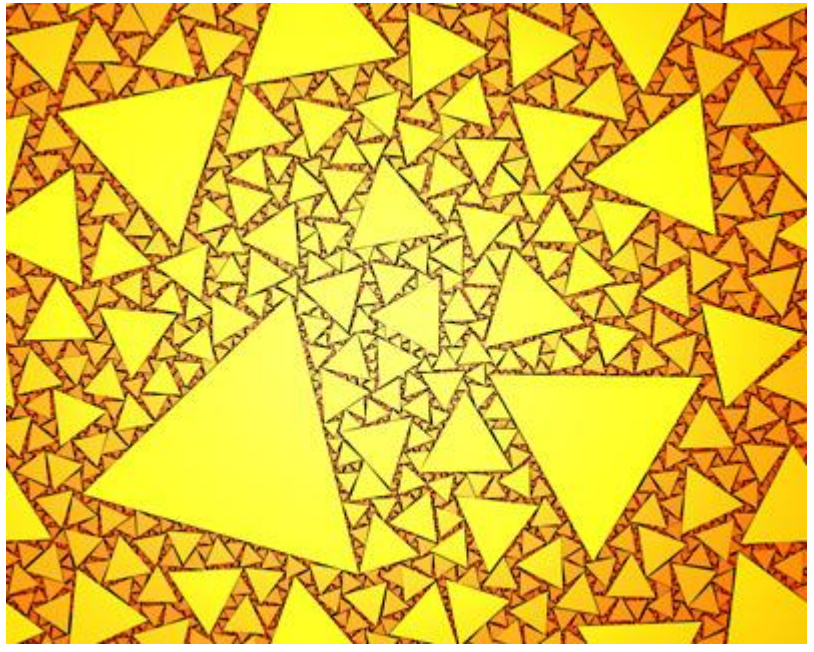
- Reduce height requires part re-orienting
- For example in Ex 1.1 we need to rotate the planar part by angle θ

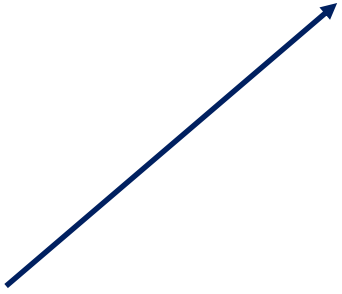
$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- At later stages we will be concerned with computing pure rotations, which will require special number types
- Number types at our disposal: arbitrary length integers, arbitrary length floating-point numbers, rationals, algebraic numbers of various sorts
- For (approximate) pure rotations we will use a result by Canny et al [CDR 92], which has several available implementations

Problem 2: Disassembly sequences

- Input: a set of pairwise interior-disjoint convex parts in the plane and a direction
- Output: an ordering of the parts, such that we can move each part in turn along the given direction, without colliding with the parts that have not yet been moved

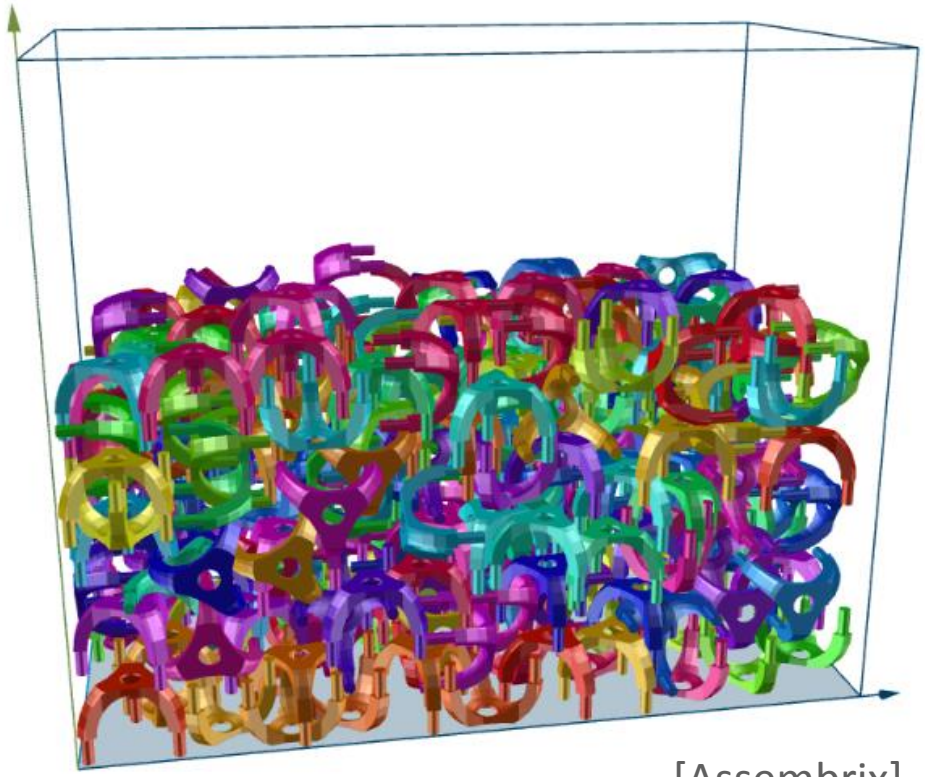




http://paulbourke.net/texture_colour/randomtile/

Disassembly, cont’d

- Unpacking the build assembly in 3DP
- Verifying a product assemblability
- Planning assembly sequences (in reverse)



[Assembrix]

Disassembly
Step I: Segments

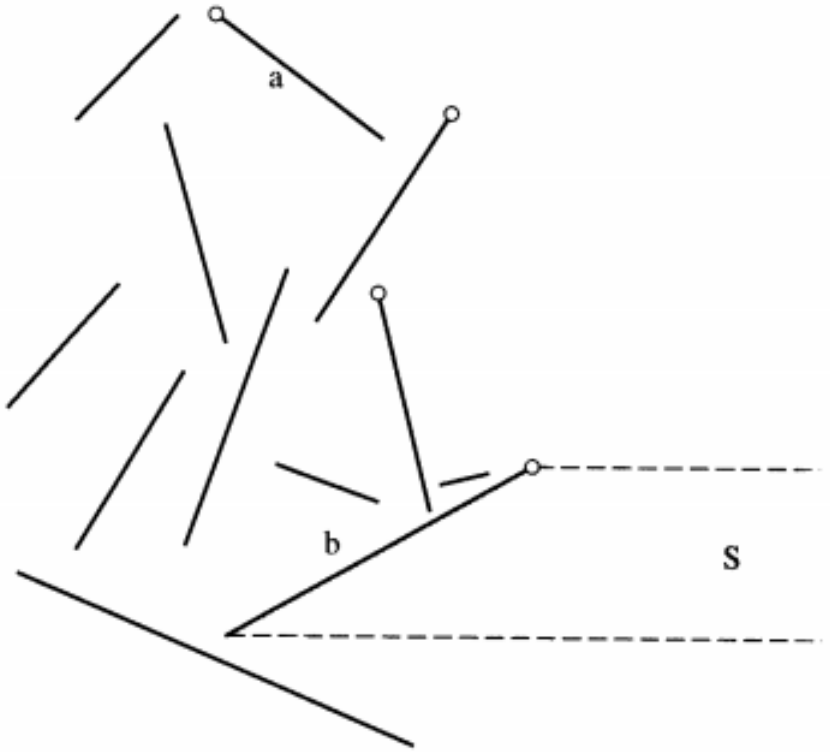


FIGURE 8.30 One segment b is always illuminated from $x = +\infty$.

Lemma 8.7.1. *In any collection of disjoint line segments, there is always at least one that is completely illuminated from* $x = +\infty$.

[O'Rourke, CG in C]

Algorithm for segments

- The blocking relation for segments
- Compute the (directional) blocking graph

naively in $O(n^2)$ time
efficiently in O(n log n) time

- Topologically sort the n vertices (segments)

O(n) time

- Total O(n log n) time

[Guibas-Yao ‘83]

Disassembly
Step II: Convex polygons

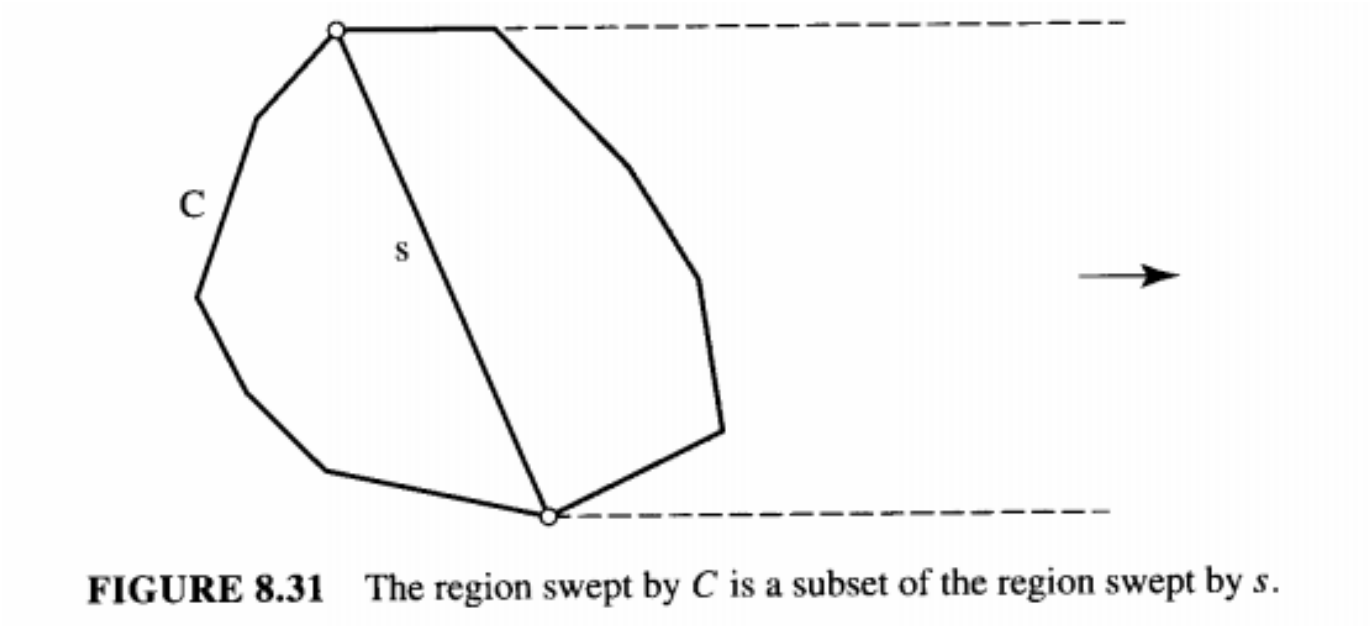


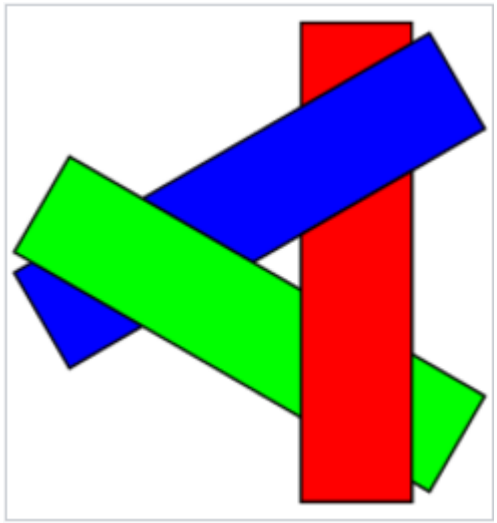
FIGURE 8.31 The region swept by C is a subset of the region swept by s.

[O'Rourke, CG in C]

Algorithm for convex polygons

- Input: m pairwise disjoint convex polygons with a total of n vertices
- Output: as before
- Find the segment connecting the topmost and bottommost points in each convex polygon
- Apply the segment-algorithm
- Total running time $O(n + m \log m)$ time

Does not extend to 3D



[Wikipedia]

THE END

[Gaither, ArtByAI, CGAL arrgs]