LayerCode: embeds barcode in 3D printed object (additive manufacturing)

可以做的改进(from sec6. evaluation)

- 研究鲁棒性：研究编解码方法和识别方法对于遮挡、同色背景的抵抗力（文中研究的基本都是没有遮挡、前背景大对比度）
- 研究速度：优化graph-based approach来降低解码时间/解码时间上界 -> feasible on mobile devices
- 研究信息容量：ternary or quaternary（三元/四元）打印可以提高信息容量
- 姿态估计

近期计划工作

- **跑跑看LayerCode的源码**
- **Tracking论文**

sec1. introduction

- Encoding algorithm
- Decoding algorithm (from single photo)
- Test with large dataset of virtual rendering **(4000+ shapes from Thingi10k dataset 2016, 99% successfully encode & decode)**
- Embedding optical tags: on, beneath, inside the surf (ref)
- Challenges:

1. robust encoding & decoding:

   standard barcode-maps every bit to a bar thickness

   we-encode bits based on local change of layer thickness (invariant under diff. orientation)

   image-paths along for decoding

2. need to introduce 2 distinguishable layer types:

   2 material printer OR diff deposition height OR mix NIR dye

sec1. introduction

- Attributes:
  - * conventional cam (NIR with filter and light source when detecting)
  - * redundancy: readable from multiple cam view and even broken
  - * **depth info for free**: parallel light pattern -> use structured light tech in CV

    how???
- Output of encoding algorithm: a series of slices along the printing direction to specify the thickness of each coding layer
- Cannot transfer flat barcode's pattern directly: because of spatially variant

  solution: use thickness ratio of 2 consecutive layers in a local region (because they share the approx. same surface tangent plane)

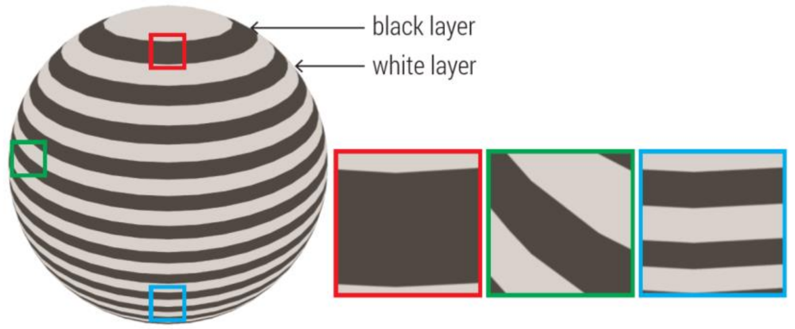


Fig. 4. Distorted thickness. A sphere is coded with black and white layers of equal thickness. On the captured image, curvature and perspective cause layers to appear spatially varying in size.

sec3. encoding

- Coding scheme details: bn -> an

  an: thickness

  bn: bit

  an+1 : an = 1 if bn+1 = 0

  an+1 : an = 1/M or M if bn+1 = 1

the proposed pipeline used M = 2

$$a_{n+1} = \begin{cases} a_n & \text{if } b_{n+1} = 0, \\ Mh & \text{if } b_{n+1} = 1 \text{ and } a_n = h, \\ h & \text{if } b_{n+1} = 1 \text{ and } a_n = Mh. \end{cases}$$

inverse map for decoding:

$$b_{n+1} = \begin{cases} 1 & \text{if } \log a_n - \log a_{n+1} = \pm \log M, \\ 0 & \text{if } \log a_n - \log a_{n+1} = 0. \end{cases}$$

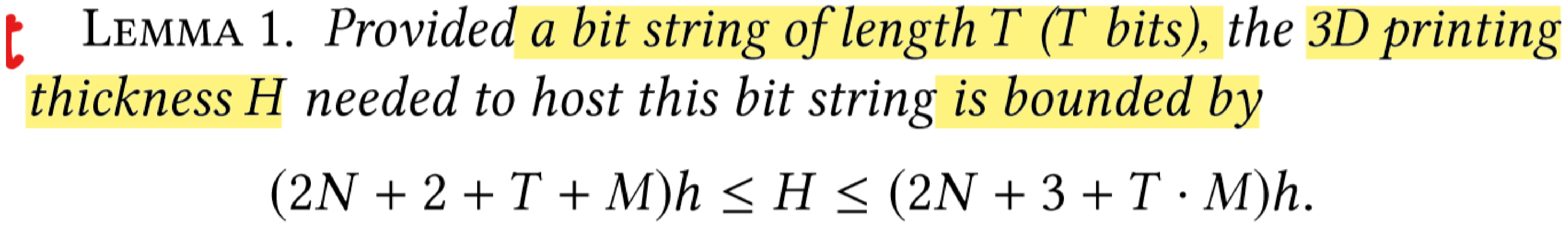
sec3. encoding

- Coding scheme details: bn -> an

 start of bit string: a layer with an = Nh (the proposed pipeline used M = 4), followed by 2 layer with thickness = h (bit 0 as the head of the bit string)

 end of bit string: a single layer appended to form bit 1 as the rear of the bit string, with the last layer is again an = Nh after the appended layer

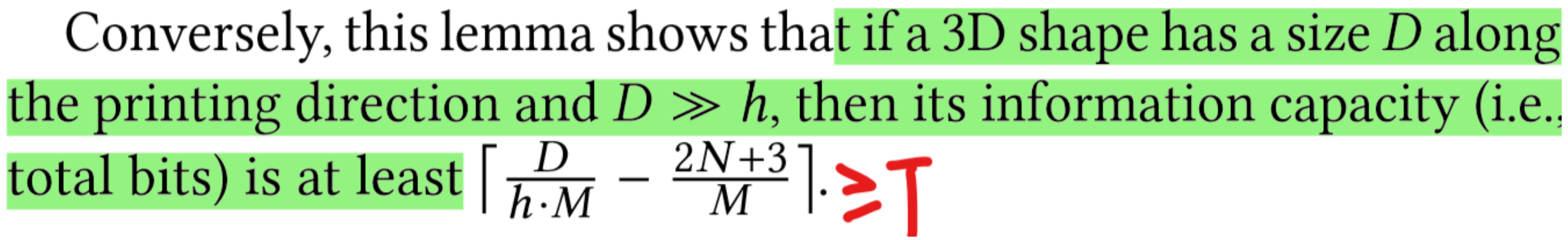
 the bit string always starts with 0 and ends with 1: direction or the barcode can be tell

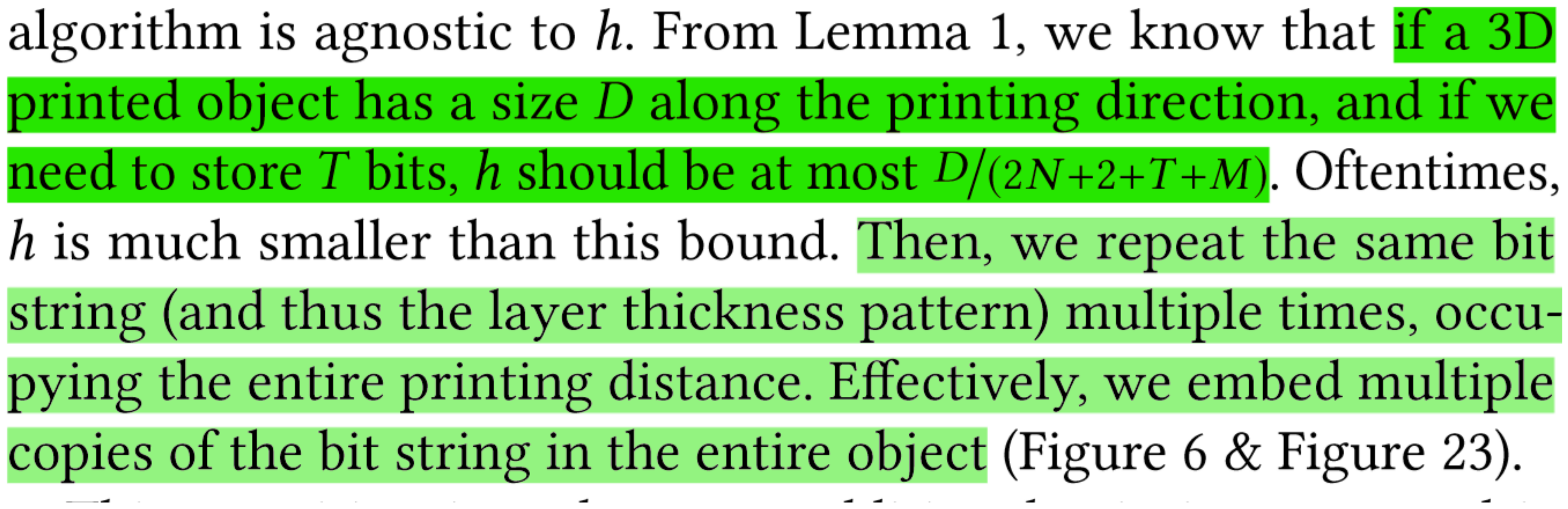


LEMMA 1. *Provided a bit string of length T (T bits), the 3D printing thickness H needed to host this bit string is bounded by*

$$(2N + 2 + T + M)h \leq H \leq (2N + 3 + T \cdot M)h.$$

sec3. encoding


Conversely, this lemma shows that if a 3D shape has a size D along the printing direction and $D \gg h$, then its information capacity (i.e., total bits) is at least $\lceil \frac{D}{h \cdot M} - \frac{2N+3}{M} \rceil$. $\geq T$


algorithm is agnostic to h. From Lemma 1, we know that if a 3D printed object has a size D along the printing direction, and if we need to store T bits, h should be at most $D/(2N+2+T+M)$. Oftentimes, h is much smaller than this bound. Then, we repeat the same bit string (and thus the layer thickness pattern) multiple times, occupying the entire printing distance. Effectively, we embed multiple copies of the bit string in the entire object (Figure 6 & Figure 23).

- Error correction: is able to carry any error-correction code as long as enough bit string length (no correction code in experiment to test pure performance)

Decoding pipeline

- Preprocessing: distinguish ‘white bar’ & ‘black bar’
- Graph construction
- Graph traversal

sec4. decoding

- Image preprocessing: see algorithm 1 & appendix A

1. 2-color printer:

    1) intensity thresholding: remaining pixels are either on black or white layer **color thresholding???**

    2) 2-way clustering through GMM in a sliding window: each window produces the label of its central pixel, repeat 3 time with diff. size **how???**

    3) pixel-wise majority vote of labels **how???**

    helpful to convert to LAB/HSV, focus on AB channel when clustering

sec4. decoding

- Image preprocessing: see algorithm 1 & appendix A

2. varying layer height objects:

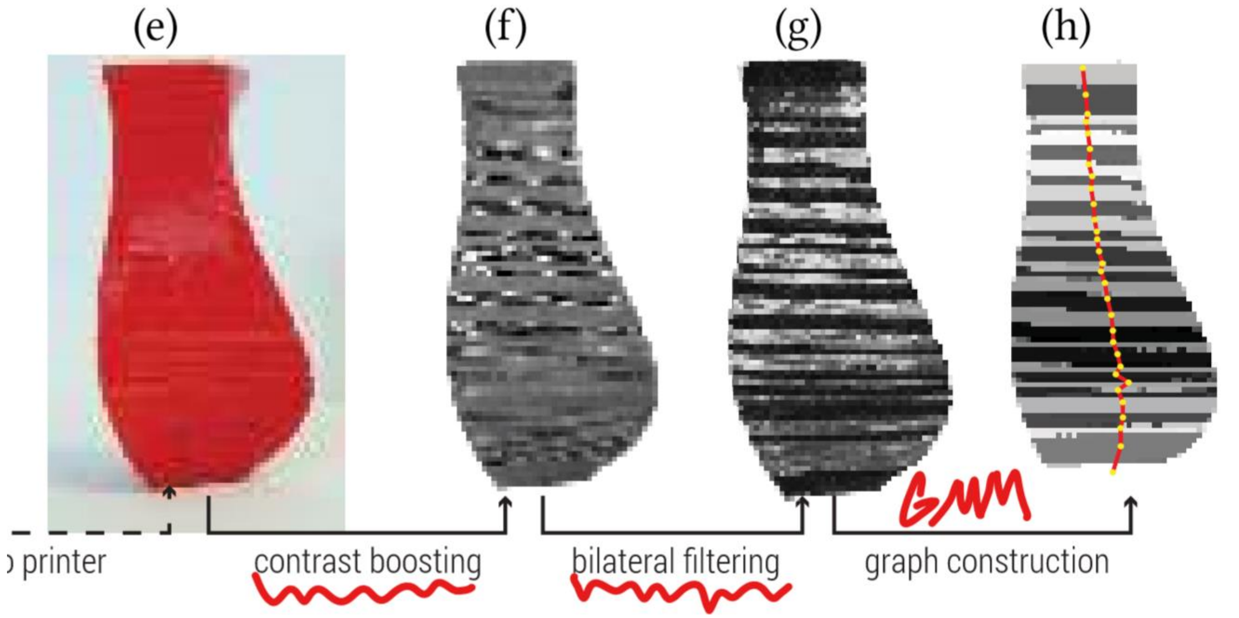
    1) remove the background (**color thresholding???**) -> greyscale

    2) morphological operation:

        bottom and top hat filtering -> increase contrast **how???**

        bilateral filter (ref) -> blur black & white layer regions **without** blurring boundary

        GMM-based clustering -> binarize the image **how???**



sec4. decoding

4.1 graph construction

- Flood fill process -> all pixels are either labelled black or white -> formed black regions & white regions -> each regions is a **node**, 2 nodes are connected if their regions are adjacent
- Edge e (connects node A & B)

vector v of e: denotes general direction to move A -> B

* identify **boundary pixels** in region A: pixels within \delta pixels away from another region (in paper \delta = 3) **how???**

* estimate **boundary normal direction** (quickest direction to get into another region from current boundary pixel **(深度优先搜索???)**

* v is the avg normal direction over all boundary pixels between A & B

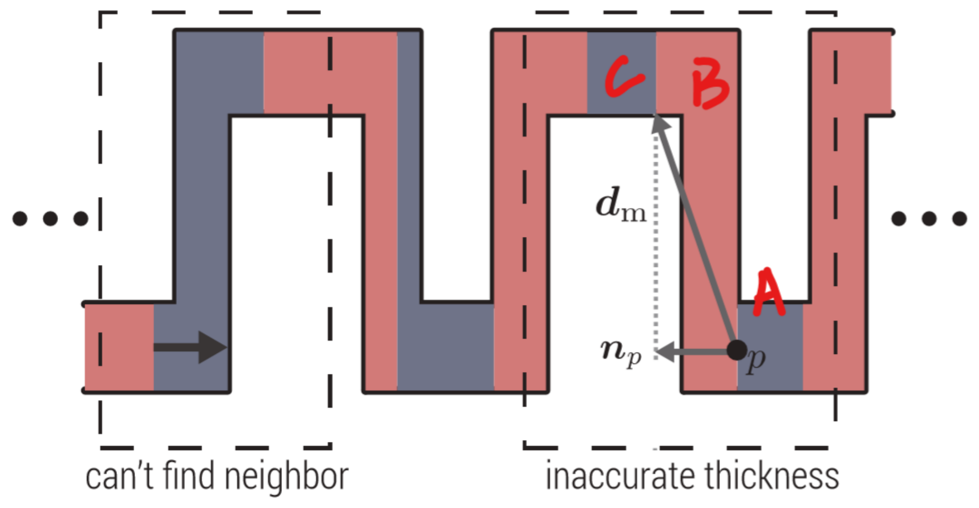
sec4. decoding

4.1 graph construction

- Edge e (connects node A & B)

  label r of e:

  * thickness estimation: for each boundary pixel p between A & B, find a shortest **image plane vector dm** between p and another region **(not A or B but C)** **(深度优先搜索???)**, project dm onto np to estimate thickness of a layer -> now hA(p) & hB(p) are known



* get bn: use 0.5logM as a threshold

$$b_{n+1} = \begin{cases} 1 & \text{if } \log a_n - \log a_{n+1} = \pm \log M, \\ 0 & \text{if } \log a_n - \log a_{n+1} = 0. \end{cases}$$

$|\log h_A(p) - \log h_B(p)| < \frac{1}{2}\log M$ (i.e., closer to 0), indicating the second case in (2) and suggesting a bit "0" encoded between A and B. On the other hand, if $|\log h_A(p) - \log h_B(p)| \geq \frac{1}{2}\log M$, it votes for label "1", suggesting the frst case in (2) and hence a bit "1". The fnal

sec4. decoding

4.1 graph construction

- Edge e (connects node A & B)

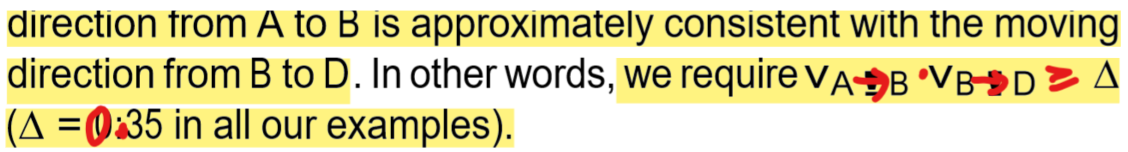
  feature label r: indicates the bit of the edge **(edge = 1 bit)**

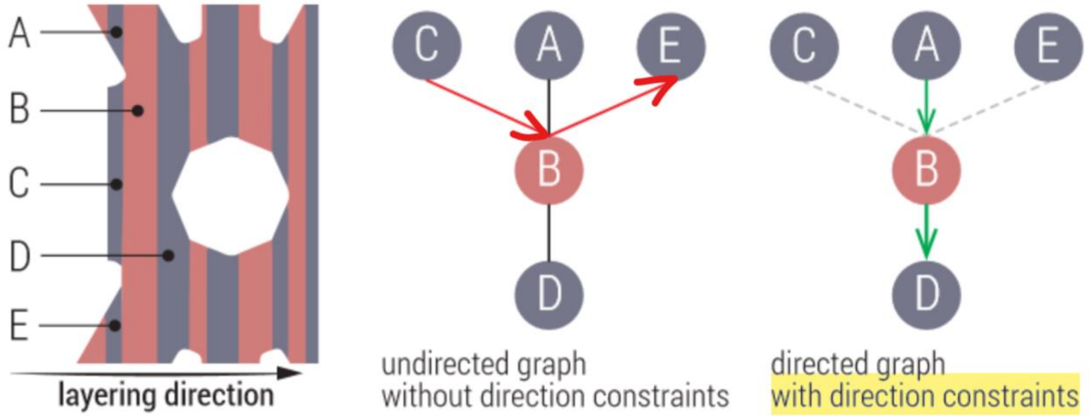
  unknown M? -> regard x = |loghA - loghB| as a random variable, the distribution of x must be a 2-peak GMM!!! Center 1 at 0, center 2 at logM -> use maximum likelihood estimation to estimate logM (p(logM) in GMM must have highest pdf except for p(0)) **h?** not imprt., can be set as prior OR encode it in the object

- Identify starting & ending nodes: outliers in GMM!!! Use a set S to contain

4.2 decoding through graph traversal

- DFS: repeatedly DFS start from a node in set S. use edge vector to make sure direction consistency (otherwise: loop)


direction from A to B is approximately consistent with the moving direction from B to D. In other words, we require $v_{A\rightarrow B} \cdot v_{B\rightarrow D} \geq \Delta$ ($\Delta = 0.35$ in all our examples).



A
B
C
D
E
layering direction
C
A
E
B
D
undirected graph without direction constraints
C
A
E
B
D
directed graph with direction constraints

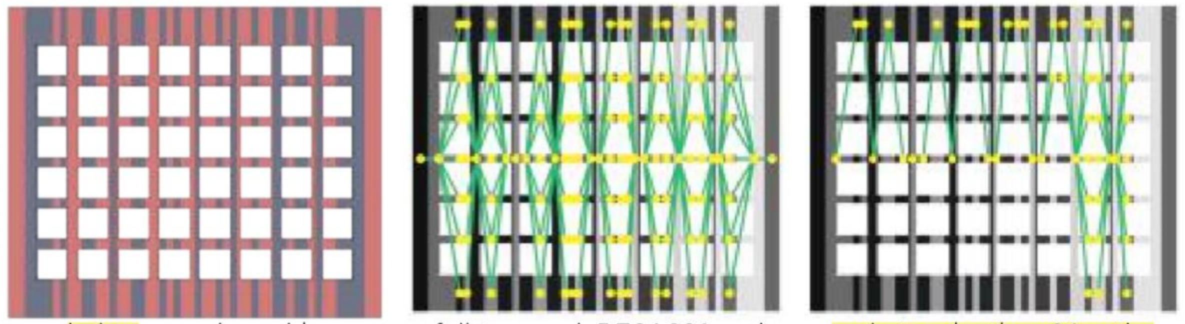
sec4. decoding

4.2 decoding through graph traversal

- traversal stops: another node in S is reached OR DFS runs out of unvisited(fatal error, because the bit string should always end with bit 0)

4.3 early termination

1. Terminate the graph traversal if we have surveyed a sufficient number of paths: terminate decoding once num_path ⩾ K (num_path is counted in DFS)
2. Terminate if: Current individual decoded bits (not string) have 80% in common


holes cause branching   full traversal: 5,764,801 paths   early termination: 64 paths

sec4. decoding

4.4 depth recovery

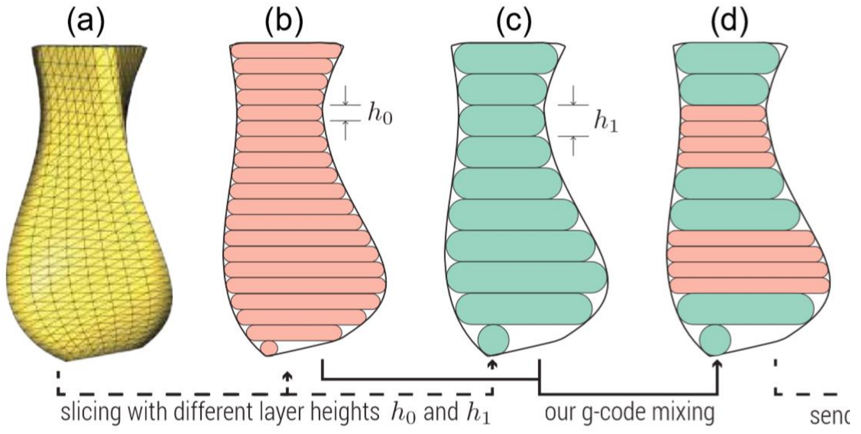
- **4.4 & Appendix B：没看懂**

sec5. fabrication

5.1 2-color fabrication (for multi-material 3D printers)

- Produce LayerCode without any modification to software/hardware

5.2 variable layer heights



- Black layers: h0 **for each layer?**

  white layers: h1 **for each layer?**

- specular high light

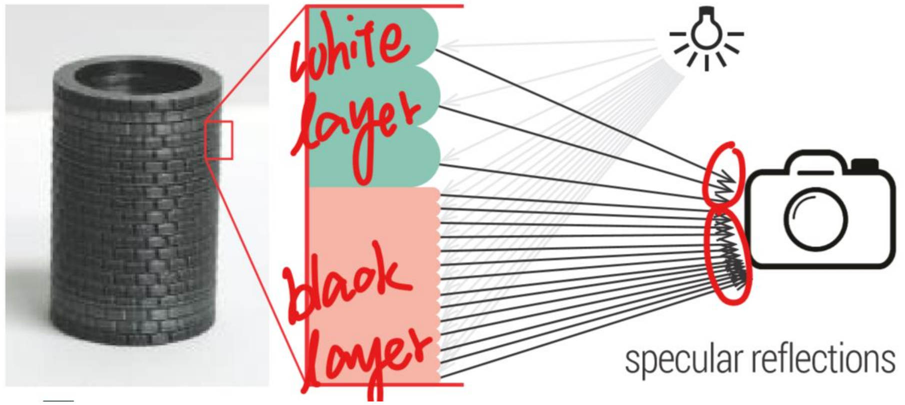
  green region: spare & granular(颗粒状的

  red region: dense & uniform

  difference of highlight distribution

  -> allows decoding to discern(分辨

  2 types of layers in preprocessing



sec5. fabrication

5.2 variable layer heights

- Interweave G-code for h0 and h1 at specific location, to construct alternating printing heights (the paper used h1 as integer multiple of h0, to ensure seamless switches across layer heights)

5.4 discussion on implementation & application

- 40 min to several hours
- 2-color & variable layer height: time cost comparable to that w/o LayerCode

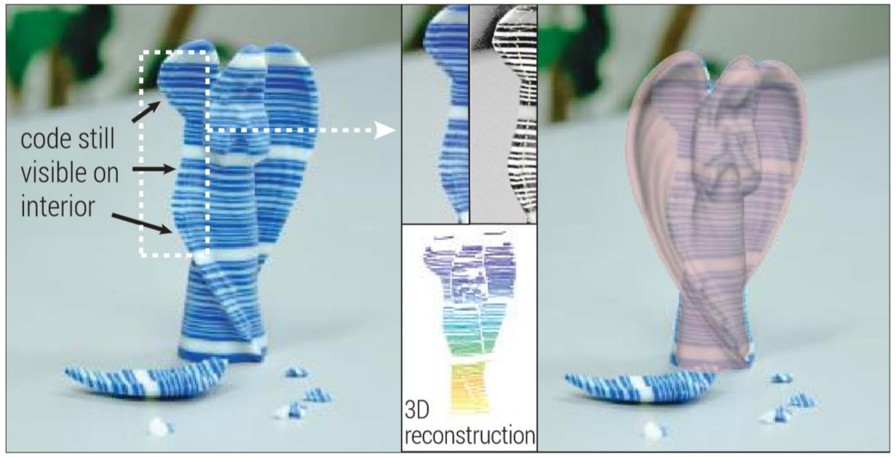


Fig. 17. LayerCode tag and augmented reality. (Lef) A fallen angel damages its wing. However, the LayerCode tag can still be read from the interior of the object. Decoding the damaged piece reveals the embedded tag, from which we know 1) the original 3D model, and 2) its 3D depth and position with respect to the camera. (Right) This information enables a virtual repair of the angel displayed in an augmented reality fashion.

sec6. evaluation

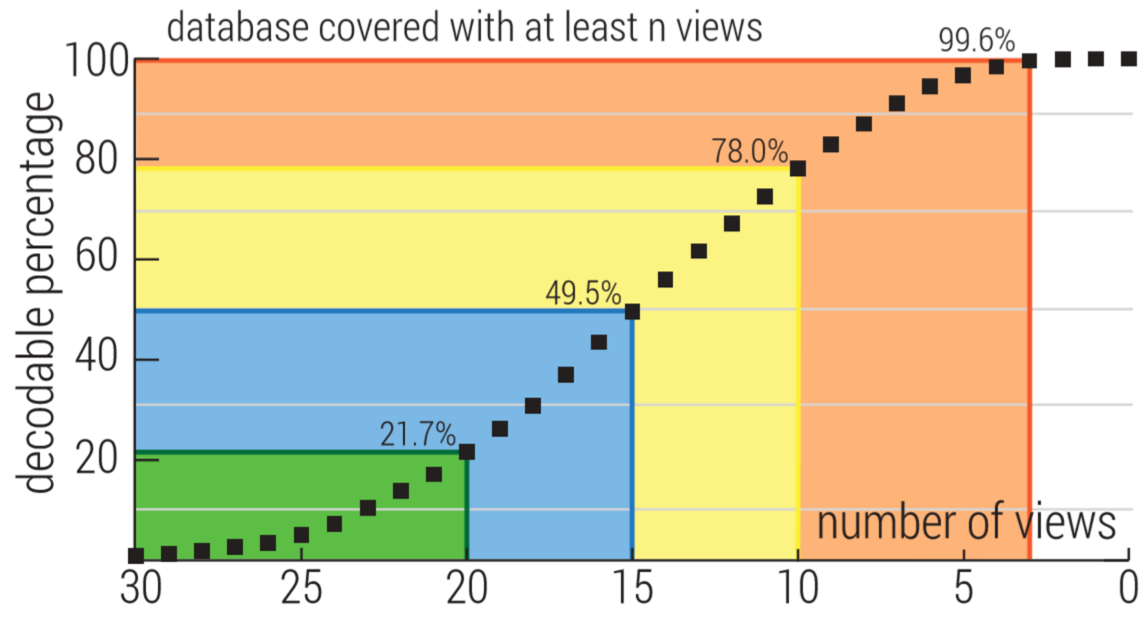
6.1 database construction

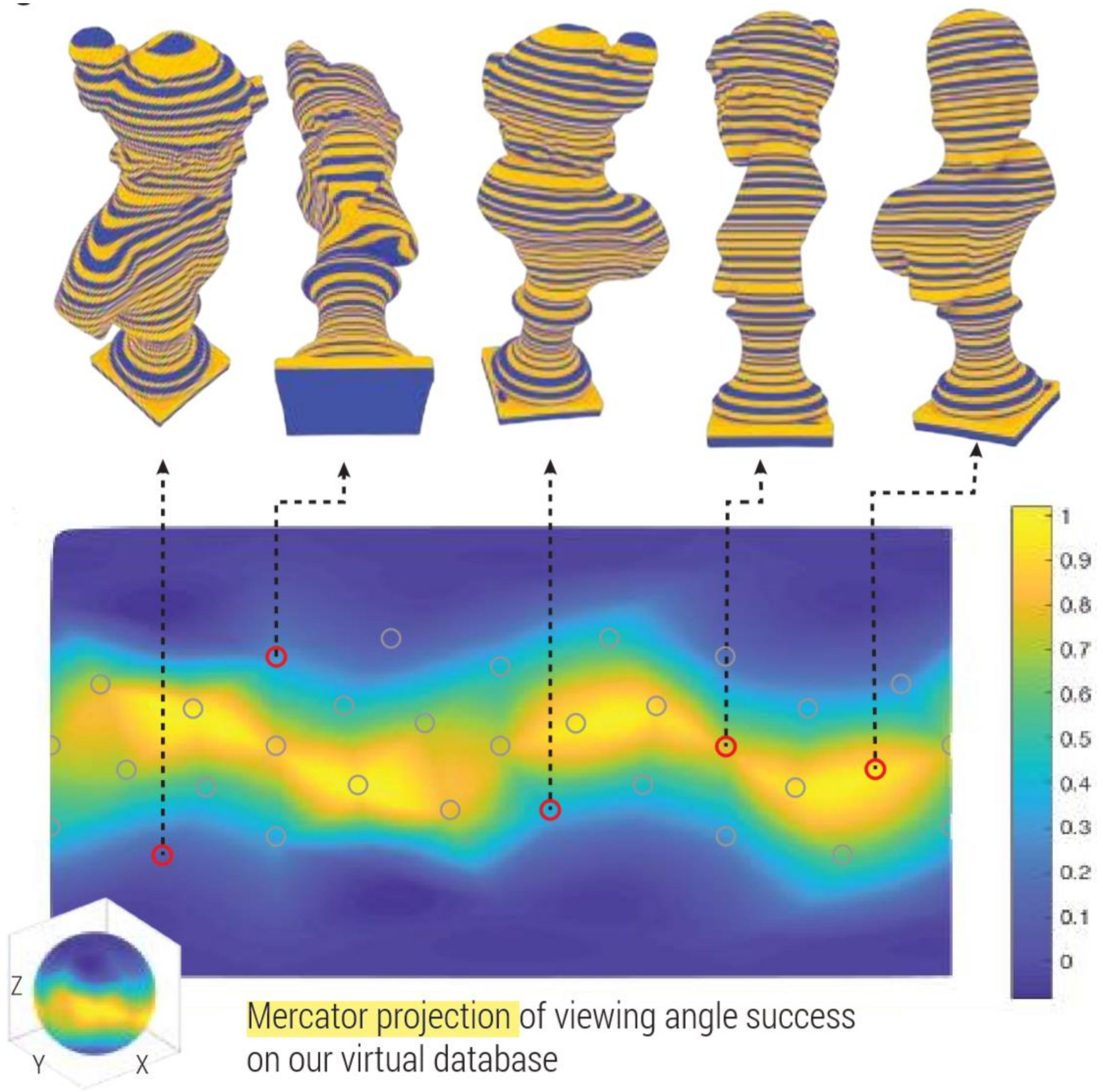
- 4835 Shape meshes from Thingi10k dataset (ref)
- Embed a LayerCode indicating model's ID
- Printing direction: longest dimension
- Baseline layer thickness h: set to repeat the tag 3 times
- Use Mitsuba(ref) to render the encoded output
- Test influence of viewpoint: sample 30 viewpoints

sec6. evaluation

6.2 result statistics

- Camera angle dependency

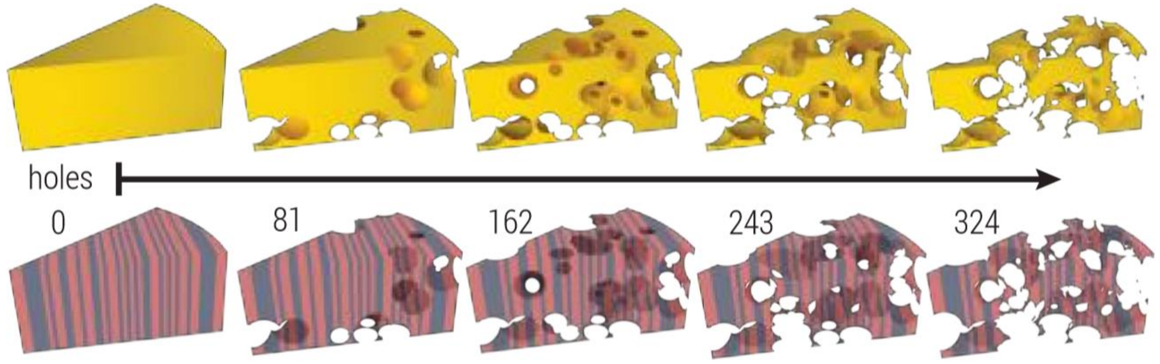




sec6. evaluation

6.2 result statistics

- Decoding time: seconds-5 min, complexity is derived from graph approach
  - -graph building operations (image processing, neighbor region dis, masking)
  - -image resolution
  - -shapes, holes, occlusions -> number of nodes -> decoding time
  - -partly because Matlab
  - -partly because wish to explore sufficient paths for robust decoding
- Lower bound of h
- Complex shapes: 4791 successful shapes
  , 44 failure cases
- Stress test
- Failure cases


holes
0
81
162
243
324