2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and
2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)

Blind Watermarking for 3-D Printed Objects using Surface Norm Distribution

Arnaud Delmotte, Kenichiro Tanaka, Hiroyuki Kubo, Takuya Funatomi and Yasuhiro Mukaigawa
Nara Institute of Science and Technology
8916–5 Takayama-cho, Ikoma, Nara 630–0192 Japan
Email: {arnaud.delmotte.zr3,ktanaka,hkubo,funatomi,mukaigawa}@is.naist.jp

***Abstract*—We present a new blind watermarking algorithm for 3d printed objects that has applications for copyright protection, proof of authenticity, object identification, and traitor tracing. It allows to embed a few bits of data in a 3d printed object and retrieve it by 3d scanning without requiring the original mesh. While prior methods embed the watermark in the vertex of the object, our method embeds in the histogram of shape to obtain the robustness for resampling and can thus work with any 3d printer and scanner. In addition, our method avoids the shape degradation by subdividing the bins of the histogram, and increases the robustness of bin localization by introducing bin margins. In the experiment, our method has been successfully tested with print simulation and with real print-scan.**

***Index Terms*—3D printing, blind watermarking**

I. Introduction

3D printing is a technology that expanded a lot recently, with affordable and reliable printers from a few hundred dollars to a few thousand dollars, with more and more materials supported such as plastic, metal, concrete, ceramic, or even food. 3D scanning technology becomes also more and more affordable, with photometric reconstruction, or with depth cameras. Due to digital right concerns for the 3d objects, 3d watermarking technology plays an important role. For example, embedding the author ID gives a proof of ownership in case someone else tries to distribute it as his own. Another example, called traitor tracing, consist of embedding a different ID in each object we sell, allowing to identify the user that illegally distributed the model. A user could do a 3d scan of the object and upload the reconstructed mesh, or directly distribute a copy of the physical object, and we would have interest to be able to prove our ownership or to identify the leak source.

A lot of research has been done on 3D mesh watermarking, that embeds an ID to the 3D mesh data. Most of the mesh watermarking methods rely on the topology of the mesh and are not applicable to 3d print-scan context because the topology is lost during the printing and scanning. Tetrahedral volume ratio (TVR) [1] is an example of this kind of methods, it relies on the vertex connectivity which will be different after the print-scan. A way to overcome this topology loss is to use the original mesh and align it with the watermarked mesh. Yamazaki et al. [2] have developed a spectral-domain watermarking for 3d printed objects that requires to align and compare with the original mesh. For copyright context, blind watermarking[1] is generally preferred to avoid the risk of theft [3]. In this paper, we focus on blind watermarking resisting to the print-scan process as illustrated in Fig. 1.

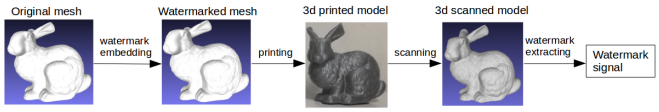


Fig. 1: 3d watermarking resisting to the print-scan process

Recently, Li et al. [4] have developed a method to embed a tag similar to a QR code below the surface of an object. The main limitation is that the surface needs to be flat or can be just slightly curved, which constrains the meshes on which it is applicable. Up to our knowledge, only one method [5] has been published yet for blind watermarking resisting to the print-scan process. It works by analyzing the layering artifacts caused by Fused Deposition Modeling (FDM) printing to find the printing axis, reorient the mesh and uses spread-spectrum watermark on a set of slices. The main limitation is that it doesn't work without the layering artifacts, which are not detectable with higher quality printer, other technology such as curving, or if the scanner has a lower resolution than the layer height. This also does not allow to reprint the model with a different orientation than the watermark.

In this paper, we propose a new blind watermarking algorithm to be applicable to the print-scan process.Our method is inspired by the vertex norm watermarking method for 3d mesh [6], [7]. Compared to previous work, the novelty of our approach is that we have less constraints to use it. Because it is rotation invariant, finding the original orientation is not necessary and thus we don't rely on the printing axis and the layering artifacts, meaning that we can use it on a printer with higher quality or with a different technology than FDM, and also that we can reprint the object from another orientation and still retrieve the watermark. .

II. Algorithm description

Our method is based upon the vertex norms watermarking algorithm ([6] and first method from [7]), but we did some improvements to make it suitable for 3d print-scan scenario. Vertex norm watermarking algorithm has been developed for

[1]Blind watermarking is a watermarking method that does not use any data from the original media

978-1-5386-5163-6/18/$31.00 ©2018 IEEE

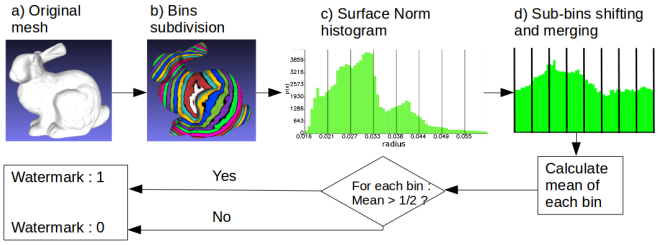


Fig. 2: Watermark extraction algorithm

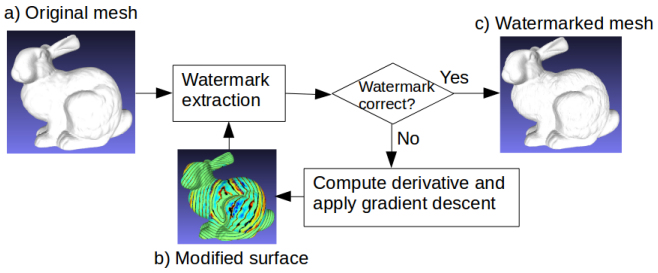


Fig. 3: Watermark embedding algorithm

digital data with mesh structure, not for print-scan scenario. It has great advantage due to the invariant to translation, rotation, rescaling and vertex connectivity, but it can also have problems when resampling is applied, which is the case in 3d print-scan. In this section, we first quickly explain the original method, then explain our contributions to make it suitable for 3d print-scan. Figures 2 and 3 give an overview of the watermarking extraction and embedding process, and sec. III describes the implementation details.

A. Vertex Norms Watermarking [6], [7]

Each vertex can be represented in spherical coordinates (p_i, θ_i, ϕ_i) where p_i is the norm of the vertex, which is the distance to the center, and (θ_i, ϕ_i) are the direction angles. By taking all the vertex norms, we can make an histogram of the norm distribution as shown in Fig. 4a. The vertical lines represent the separations between the different bins. Inside each bin, we embed one bit of information by modifying the norm distribution such that the mean of the distribution is respectively on the left or the right of the center of the bin, to encode a 0 or 1, as shown in Fig. 5. To modify the norm distribution, each vertex norm p_i will be increased or decreased, without exceeding the bounds of their bin. The directions (θ_i, ϕ_i) stay unchanged. Figure 4b shows an example of histogram of watermarked object.

B. Increasing robustness against resampling

The original method uses the vertices to encode the watermark. The print-scan process produces a complete resampling of the mesh, and may cause decoding errors because the vertex positions in which watermark has been encoded are lost, and new vertices are generated. Using a uniform sampling with high vertex density could reduce the effect of the resampling because the newly generated vertex would be close to the original ones and in same density, but the watermarking process does not preserve this uniformity so it would still be a problem.

In order to avoid the resampling problem, we compute a probability density function of the surface norm, which is the distance between the surface and the center of gravity, over the complete surface instead of a set of vertex. This corresponds to a continuous approach instead of a discrete one, and is not sensitive to resampling if the shape is not degraded. For the computation of the mean of each bin, each triangle is subdivided such that each sub-triangle is included only in one bin, then we take the mean of the norm of the three vertices of each triangle inside a bin, multiplied by the area of these triangles, and we divide by the total area of all the triangles inside a bin. Note that this is an approximation of the real value, but if the triangle is far enough from the center of gravity relatively to its size, the error is very small. Additionally, for the watermark encoding, instead of modifying a vertex position by changing its norm, we move it along the normal of the surface. This prevent the vertex to move along the tangent of the surface, which would have no effect after resampling, and would have a risk of producing invalid mesh, as shown in Fig. 7.

C. Combination of multiple non-consecutive bins

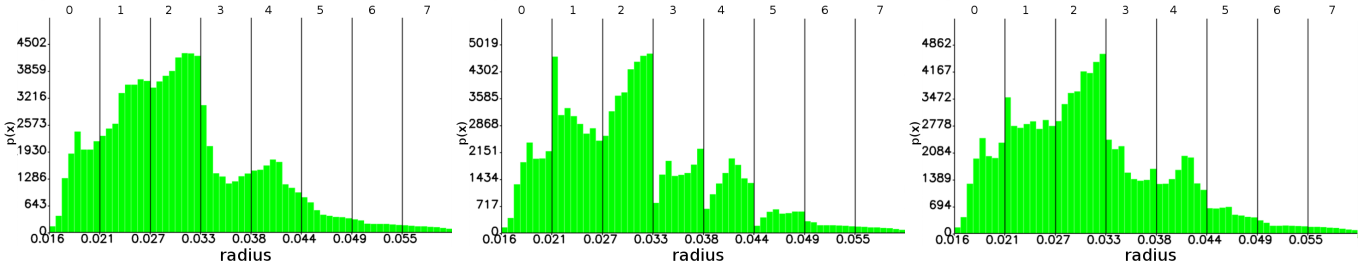
Because the bin segmentation is done regularly by distance to the center of gravity, some bins are filled by a large surface of the model, others by only a small surface. For example, in Fig. 2.b, the surface of a bin on the ear of the bunny is much smaller than a bin on the central part of the body. Fig. 4a shows that the bin 5,6,7 contain much less surface than the other bins and are thus more sensitive to local printing or scanning errors. Additionally, the norm distribution inside a bin is not always uniform, it is especially visible in Fig. 4a in bin 3. If the bin would be originally biased likely to have a bit, and the bit to encode is the opposite value, it requires a large modification in the distribution. To get a more uniform surface distribution, reducing the impact of regions difficult to watermark where is parallel to the vector to the center, we subdivide each bin and combine equally spaced sub-bins to form a new bin. Figure 8 illustrates this combination, we can observe that the resulting histogram is more uniform than the original one.

D. Increasing robustness of bin localization

The limits of each bin are calculated based on the center position, minimal and maximal distance to the center. If any of these values is slightly changed, it has a big impact on all the extraction process so we must try to make it robust to errors.

1) Center preservation : As explained in [8], if the center position has been changed, the norms will be modified too and the watermark will not be extracted correctly. Because the mean of the vertices is not resistant to resampling, we use a moment-based center estimation that has been proven to be more resistant in 3d printing context [5].

$$c = (\overline{x}, \overline{y}, \overline{z}) = \frac{(M_{100}, M_{010}, M_{001})}{M_{000}}, \quad (1)$$



(a) original distribution (b) after the watermarking process (c) after the print simulation

Fig. 4: Distribution of vertex norms from the bunny model, each vertical line represent the border of a bin.

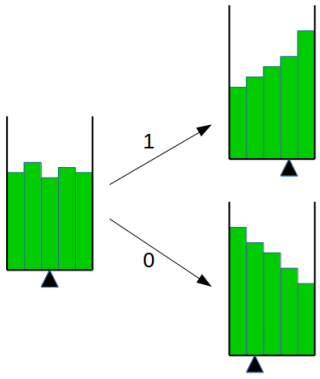


Fig. 5: Encoding 1 bit of data by shifting the mean to the left or right side of the bin.

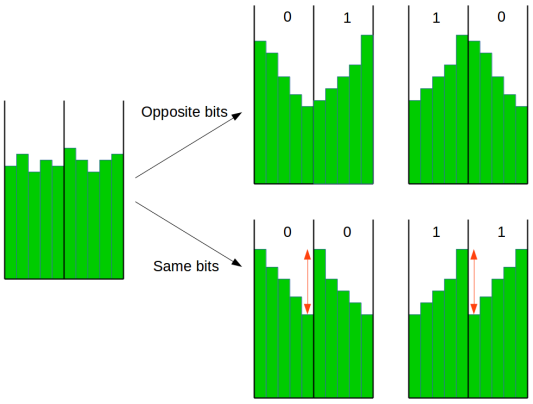


Fig. 6: Example of watermarking result for 2 bins. Encoding opposite successive bits is easier than same bits because it avoid a huge gap at the border between the bits

where M_{pqr} denotes the p, q, r-th order volume moment of the mesh. The watermarking process has generally very small effect on the center position because the shape is only slightly changed, but for safety, we include the conservation of the center in the list of parameters to optimize.

2) Preservation of the minimal and maximal norm : If we modify the surface around the minimal or maximal norm, all the bins will be shifted and it will result in decoding errors. We keep a border distance around the minimal and maximal norm where we make no modifications on the surface. In case

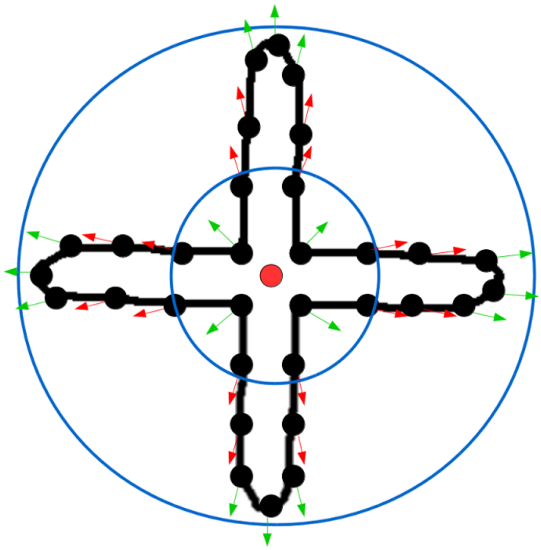


Fig. 7: Example of vertex transformation during the watermarking process. Vertices are modified along the vector from the center. When the vector is nearly parallel to the surface (red arrows), the shape will not be affected by the vertex displacements, and the modification will be lost after remeshing. Blue circles are the border of bins, red point is the center and black points are the vertices

the surface around the minimal or maximal norm corresponds to a spike or any very thin volume in the model, it may be difficult to print or scan and produce errors. We can increase the resistance by slightly rounding the border.

3) Margin between the bins : As shown in Fig. 6, vertices on the border of two bins get a huge change of density if the two bins encode the same bit value. New vertices will be generated in this gap after the resampling, reducing the strength of the watermark. For example, on Fig. 4b, the transition between the bin 2 and 3 is a bin maximum to bin minimum transition. After the printing simulation, on Fig. 4c, this gap has been partially filled, reversing the value of bin 3. Additionally, a small shift of the bin borders may also have a really strong effect on the mean. To reduce this problem, we included a small margin in which surface is not used for computation of the mean of the bin. This allows a softer

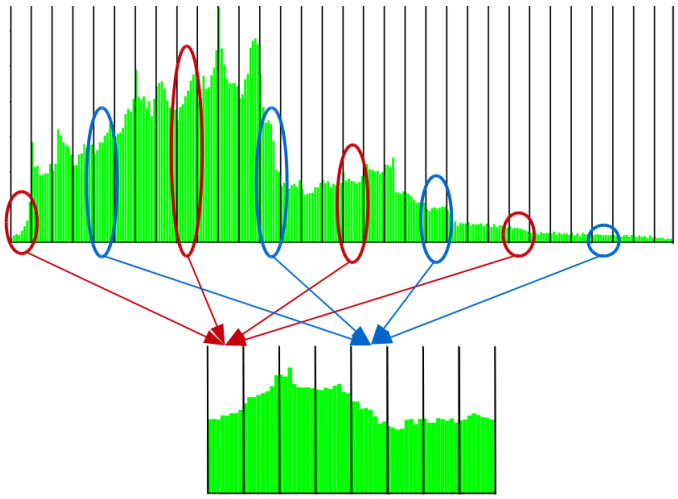


Fig. 8: Fusion of multiple non-consecutive bins to get a more uniform histogram

change of density and reduce the effect of misalignments.

III. Watermark embedding and extraction algorithm implementation

We will now describe more precisely the embedding and extraction algorithm. We first describe the input parameters for the watermarking, we need to use the same set of parameters for embedding and extracting the watermark. Then we describe the extraction algorithm and finally the embedding algorithm.

A. Algorithm parameters

Here are the parameters of the algorithm:

$minNorm$	minimal norm of the mesh
$maxNorm$	maximal norm of the mesh
N	number of watermark bits
N_d	number of sub-bins per bits (see sec. II-C)
m	ratio of margin in the bin (generally set to 5%, see sec. II-D3)
b	number of sub-bins that we keep constant around $minNorm$ and $maxNorm$ (generally set to 1, see sec. II-D2)
$\#subBin$	number of sub-bins
$subBinSize$	size of a sub-bin
$minBound[j]$	lower bound of the sub-bin j
$maxBound[j]$	upper bound of the sub-bin j
$S[j]$	total area of the triangles inside sub-bin j
$\mu[j]$	mean norm inside sub-bin j
$S'[i]$	total area of the triangles inside bin i
$\mu'[i]$	mean norm inside bin i
$w[i]$	i^{th} bit of the watermark signal
$strength[i]$	strength of the i^{th} bit in the range [-1,1], negative value if the bit is wrong

B. Watermark extraction algorithm

As shown in Fig. 2, the extraction algorithm consists of computing the value in each sub-bin, then combine the sub-bins together and computing the mean of each bin. We then output 1 if the mean is greater than 0.5, or 0 otherwise.

More concretely, we begin by centering the model with the moment method described in sec. II-D1. Then we subdivide each triangle such that it is fully included in a single sub-bin. The boundaries of the sub-bins are computed by

$$\begin{cases} \#subBin & = N * N_d + 2 * b, \\ subBinSize & = \frac{maxNorm - minNorm}{\#subBin}, \\ minBound[j] & = minNorm + (j+m) * subBinSize, \\ maxBound[j] & = minNorm + (j+1-m) * subBinSize, \end{cases} \tag{2}$$

where j is the index of the sub-bin. To compute the area of a triangle, we use the formula

$$Surf(A,B,C) = \frac{|\overrightarrow{AB} \times \overrightarrow{AC}|}{2} \tag{3}$$

where A,B,C are the three vertex coordinates of the triangle. We also define the function

$$f_j(X) = \frac{X - minBound[j]}{maxBound[j] - minBound[j]} \tag{4}$$

that remap the norm X, which is inside the sub-bin j, to the range [0,1]. Then we compute the total area $S[j]$ and mean norm $\mu[j]$ in each sub-bin :

$$S[j] = \sum_{(A,B,C) \in Tri[j]} Surf(A,B,C) \tag{5}$$

$$\mu[j] = \sum_{(A,B,C) \in Tri[j]} f_j\left(\frac{||A|| + ||B|| + ||C||}{3}\right) \frac{Surf(A,B,C)}{S[j]} \tag{6}$$

where A,B,C are the coordinates of a vertex, $Tri[j]$ is the list of triangles included in the range $[minBound[j], maxBound[j]]$. We merge the sub-bins into the bins:

$$S'[i] = \sum_{j=1}^{N_d} S[i + j * N + b] \tag{7}$$

$$\mu'[i] = \frac{1}{S'[i]} \sum_{j=1}^{N_d} \mu[i + j * N + b] * S[i + j * N + b] \tag{8}$$

And finally get the watermark signal by $w[i] = (\mu'[i] > 0.5)$. We define the strength of a watermark bit by the distance to the mean : $strength[i] = 2|\mu'[i] - 0.5|$

C. Watermark embedding algorithm

As shown in Fig. 3, the embedding algorithm is just using a gradient descent to move the mean of the bins. It computes the derivative of the function from the extraction algorithm for each vertex and loop until the watermark is encoded with enough strength. Algorithm 1 describe this process.

The first and last b sub-bins are not used for computation of the bins value so will remain unaffected by the algorithm. We can also reinforce these points by encoding 0 in the first sub-bin and 1 in the last sub-bin, which will have the effect

Algorithm 1: Embedding algorithm

Result: Watermarked mesh
Center the object with the moment method described in sec. II-D1;
Subdivide the triangles to be smaller than the sub-bin size, but bigger than the layer height.;
Compute the normal of the surface at each vertex.;
while *encoded strength <target strength* **do**
  Compute the mean of each bin (similar to the extraction algorithm);
  Compute the derivative of the mean for each vertex along the precomputed normal;
  Compute the derivative of the object center for each vertex;
  Apply one step of gradient descent;
end

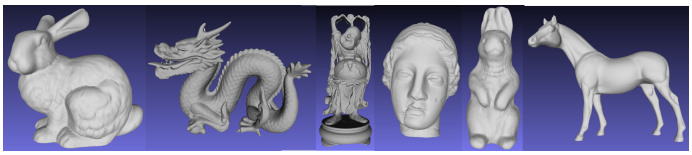


Fig. 9: 3d objects used for evaluation. From left to right : bunny, dragon, happy, venus, rabbit, horse

of increasing the concentration of surface of norm close to $minNorm$ and $maxNorm$ without exceeding it.

IV. Experiment results

Our goal in this paper is to have a watermarking algorithm resistant to the print-scan process, so our experiments focus on this process. We did our experiments both in simulation and with real print-scan. Real experiment is very time consuming, so we could only do a few ones for validation.

A. Simulation Results

For the simulated result, we introduce a new method to evaluate watermark algorithm for the print-scan context. Our goal is to simulate the printing process and generate a new mesh that has a shape similar to a real printed object, and try to extract the watermark from generated mesh. We take as input a 3d mesh and a layer thickness, and generate the "gcode", which corresponds to the list of the commands sent to the printer during the printing process, via a slicing software called "slic3r" [2]. Based on these printer commands, we generate each 2D slice as a Truncated Signed Distance Function (TSDF), fill the holes inside the object because we only need the external shape. Finally, we generate the new mesh with Marching Cube Algorithm, assuming the layers have an ideal shape with round borders. This simulation does not cover all the errors that occur in a real print-scan process, but it lets us evaluate the effect of the layering. It simulates the result of an ideal FDM printer for a given layer thickness.

[2]http://slic3r.org/

TABLE I: Simulation results by original method (encoding strength 10%, layer height 0.2mm)

N	Model	Volume (cm^3)	Decoding errors	Decoding strength min	Decoding strength mean
8	bunny	250	1.6	-10.69%	6.16%
8	rabbit	250	2.4	-6.58%	3.03%
8	venus	250	0.8	-7.56%	8.06%
8	horse	250	2.4	-19.29%	2.55%
16	bunny	250	4	-8.54%	6.32%
16	rabbit	250	2.6	-10.13%	3.79%
16	venus	250	0.8	-8.56%	7.13%
16	horse	250	3.8	-9.76%	3.32%
24	bunny	250	4.2	-7.54%	4.00%
24	rabbit	250	2.6	-10.16%	3.97%
24	venus	250	1	-8.23%	5.54%
24	horse	250	6	-9.87%	2.26%
32	bunny	250	4.6	-7.24%	4.05%
32	rabbit	250	3.6	-10.02%	3.69%
32	venus	250	1.4	-7.89%	5.27%
32	horse	250	2.4	-11.77%	3.84%

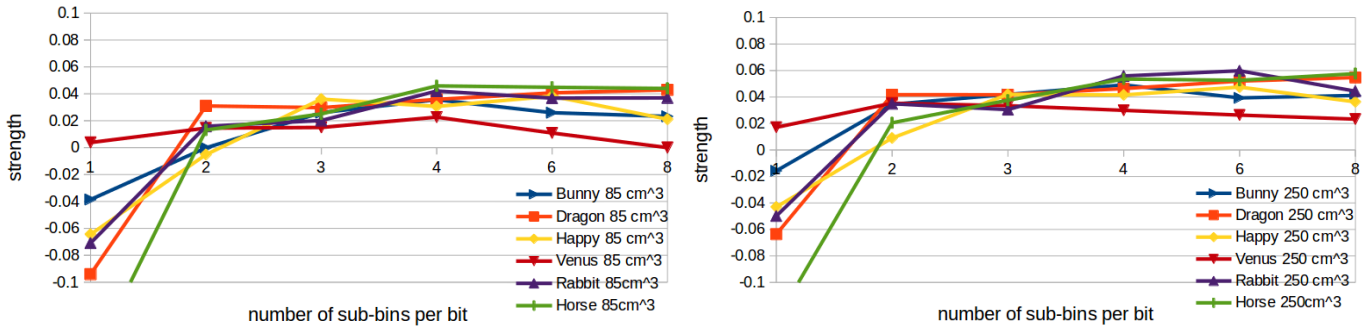
We did multiple simulations by embedding a watermark of 8 to 32 bits in the 3d models from the dataset represented in Fig. 9, simulating the printing process and testing if we could retrieve the watermark. We compared results with the original vertex norm watermarking algorithm [6], [7].

We need a metric that evaluates the probability of having a decoding error because having even a single bit error can be a problem in a lot of watermarking scenarios, so we chose the minimal detection strength (as defined in section III-B) among the bins, and we take the minimal value out of five simulations with different watermark values. Figure 10 shows the strength for 8 bits watermark. We can observe that the ideal number of sub-bins is between 3 and 6, and that a bigger volume gives better results, which is normal because the shape is proportionally less degraded by the layering effect. Figure 11 shows the strength for 16 bits watermarking. We now observe that the ideal number of bins is between 2 and 4. Having more sub-bins gives the advantage that the initial norm distribution is more uniform and average the difficulty to watermark each bin, but also give the disadvantage to be more sensitive to misalignment and measurement errors because the bins are smaller. Based on our experiments, using a number of sub-bins per bit $N_d = 48/N$ gives good results.

Figure 12 gives more detailed results. Mean decoding strength is the mean value across all the bins and across the five simulations, minimum decoding strength is still the same metric as in fig. 10 and 11. Table I is the result of the original method. We observe decoding errors in most of the simulations.

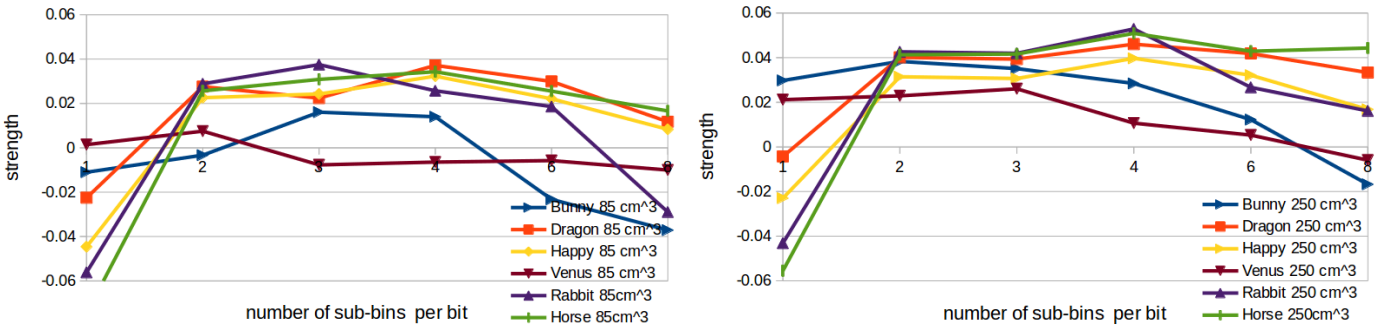
B. Print-Scan Results

For 3d printing, we used a "DaVinci Color" 3d printer with PLA filament. We did not use the color ink. This printer does not support soluble support materials, it is often difficult to totally remove the support material, and this manual removal often degrades the faces on contact. This is visible on the second bunny of Fig. 13, which has been printed on the side instead of the normal orientation. The printing quality is



(a) volume : 85 cm^3 (b) volume : 250 cm^3

Fig. 10: Minimal detection strength for 8 bits watermark, layer height : 0.2mm



(a) volume : 85 cm^3 (b) volume : 250 cm^3

Fig. 11: Minimal detection strength for 16 bits watermark, layer height : 0.2mm

similar to most of the consumer 3d printers. For 3d scanning, we used "HP 3D Structured Light Scanner Pro S3 Single Camera" previously known as "David SLS-3".

The 3d print-scan process is very time-consuming, so we could only do a few experiments to verify that our method could really be used in real condition. The table II contains the results of our 3d print-scan experiments. "decoding errors" is the number of bits wrongly decoded, "encoding strength" is the strength set in each bin as defined in sec. III-B, and "decoding strength" is the remaining strength after print-scan process. The "min decoding strength" is the most important metric because it is the smallest strength of all bins, and indicate how close we are to decode wrongly a bit. Figure 13 shows a few 3d printed watermarked models.

We tested models with different number of bits, but also tried to *reprint* a model: this consist of scanning a watermarked 3d printed object, and printing the scanned model. This increases the degradation on the mesh because we have two times the digital and physical conversions. Additionally, we also used another printing axis. The angle following "reprint" in the result table II indicates the rotation compared to the original printing axis. We got one wrong bit on our test with 24 bits which may be due to the fact that we need to reduce the number of sub-bins when we increase the number of bits to encode, and thus making more sensitive to local errors.

TABLE II: print-scan results (layer height 0.2mm, volume 85.75 cm^3)

N	N_d	Model	Decoding errors	Encoding strength	Decoding strength min	mean	stddev
8	6	bunny	0	10%	3.17%	4.35%	0.99%
8	6	bunny	0	reprint 10°	1.25%	2.96%	0.91%
8	6	bunny	0	reprint 90°	0.54%	1.74%	0.85%
8	6	rabbit	0	10%	2.75%	3.85%	0.77%
16	3	bunny	0	10%	3.32%	4.35%	0.76%
24	2	bunny	1	10%	-0.34%	3.57%	2.01%

V. Discussion and Future Work

We presented a blind watermarking algorithm for 3d printed objects, with low visibility and resisting to the print-scan process even in presence of some printing and scanning errors. Our method has less constraint than the previous work, so our method supports reprint in any orientation or higher quality printer without slicing artifacts. We used a rotation invariant method to avoid orientation synchronization step. Our main improvements were to embed the watermark signal in the histogram of the shape instead of individual vertex to be robust to resampling, subdivide and shuffle the bins to equalize the robustness of the signal in the different bins, and increased the robustness of the bin localization.

We presented a new evaluation method that allows to simulate the layering effect which is one of the main sources of errors for the print-scan scenario, and is much faster than a real print-scan evaluation.

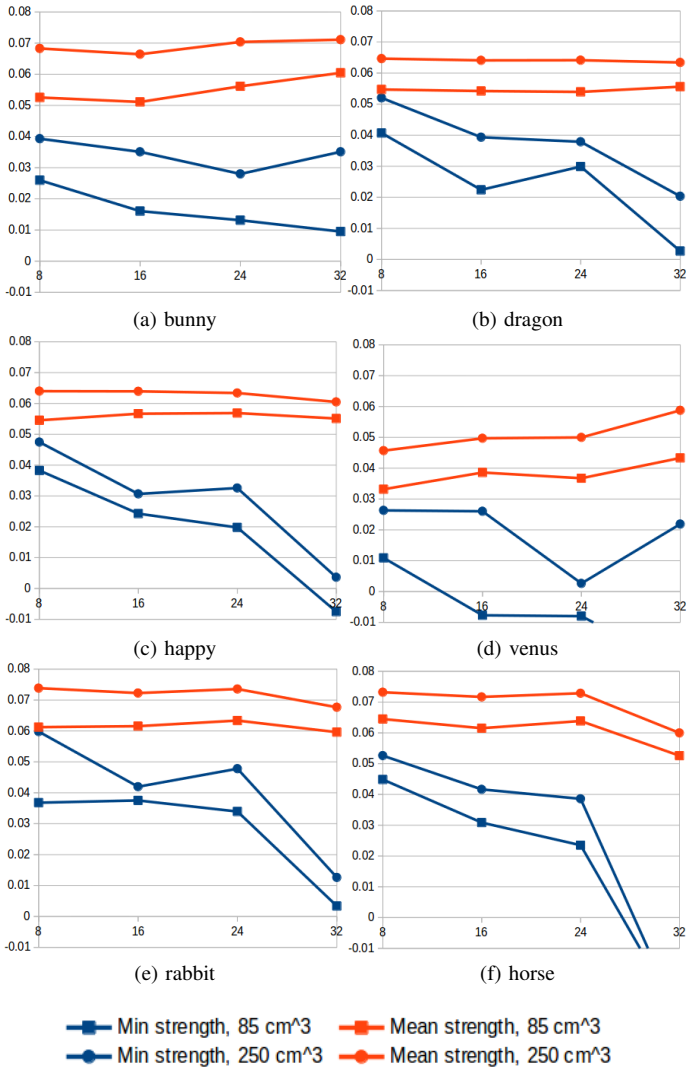


Fig. 12: Simulation result for the different objects, using $N_d = 48/N$. X axis is the number of bits, Y axis is the detection strength.

Fig. 13: Real print of watermarked models. From left to right : bunny 8 bits, reprint 90°of bunny 8 bits, rabbit 8 bits, bunny 16 bits. For the second bunny, we used the first bunny, scanned it, then printed the scanned model with a 90°angle, some degradations are visible because we printed it on the side, and the surface was on contact to the support material.

We successfully validated our method on a set of 3d models on simulation and with real print-scan experiments. The number of bits usable in practice will depend on the context, the size of the objects, and the quality of the printer and scanner used. We focused our experiments on natural shapes. CAD files are more difficult to watermark with our method because flat surfaces make the watermark very visible, allowing to use less strength if we want to maintain imperceptibility.

Our 3d printer only supports PLA so we did not try other materials such as ABS, resin or powder. Our method is not dependent on the type of material or the printing and scanning technique used, and could theoretically be used with any materials as long as the printing and scanning quality is good enough. For ABS, it is generally considered a little bit more difficult to print than PLA due to warping and shrinking problems, but with a good temperature control during the print with a heated bed and enclosure, it is still possible to get high quality print. Dark or translucent materials make the scanning process more difficult with standard techniques such as laser scan or structured light, resulting in poor scanning quality and affecting the decoding process.

For future work, we plan to improve the encoding algorithm to better handle different resolutions instead of having to resample the model to a chosen resolution, and to improve the imperceptibility by weighting the area to encode by the roughness [9]. We also plan to do more experiments including different printers like the high quality ones from online 3d printing services to see if we can embed more than 16 bits with a better printer. We will also evaluate some attacks such as surface treatments like smoothing.

References

[1] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking three-dimensional polygonal models through geometric and topological modifications," *IEEE Journal on selected areas in communications*, vol. 16, no. 4, pp. 551–560, 1998.

[2] S. Yamazaki, S. Kagami, and M. Mochimaru, "Extracting watermark from 3d prints," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 4576–4581, IEEE, 2014.

[3] B. Macq, P. R. Alface, and M. Montanola, "Applicability of watermarking for intellectual property rights protection in a 3d printing scenario," in *Proceedings of the 20th International Conference on 3D Web Technology*, pp. 89–95, ACM, 2015.

[4] D. Li, A. S. Nair, S. K. Nayar, and C. Zheng, "Aircode: Unobtrusive physical tags for digital fabrication," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pp. 449–460, ACM, 2017.

[5] J.-U. Hou, D.-G. Kim, and H.-K. Lee, "Blind 3d mesh watermarking for 3d printed model by analyzing layering artifact," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2712–2725, 2017.

[6] J.-W. Cho, M.-S. Kim, R. Prost, H.-Y. Chung, and H.-Y. Jung, "Robust watermarking on polygonal meshes using distribution of vertex norms," in *International Workshop on Digital Watermarking*, pp. 283–293, Springer, 2004.

[7] J.-W. Cho, R. Prost, and H.-Y. Jung, "An oblivious watermarking for 3-d polygonal meshes using distribution of vertex norms," *IEEE Transactions on Signal Processing*, vol. 55, no. 1, pp. 142–155, 2007.

[8] R. Hu, P. Rondao-Alface, and B. Macq, "Constrained optimisation of 3d polygonal mesh watermarking by quadratic programming," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pp. 1501–1504, IEEE, 2009.

[9] G. Lavoué, "A local roughness measure for 3d meshes and its application to visual masking," *ACM Transactions on Applied perception (TAP)*, vol. 5, no. 4, p. 21, 2009.