3D GEOMETRY REPRESENTATION USING MULTIVIEW CODING OF IMAGE TILES

Yu Gao [*], *Gene Cheung* [#], *Thomas Maugey* [$], *Pascal Frossard* [$] *and Jie Liang* [*]

[*] Simon Fraser University, [#] National Institute of Informatics,
[$] Ecole Polytechnique Fédérale de Lausanne (EPFL)

ABSTRACT

Compression of dynamic 3D geometry obtained from depth sensors is challenging, because noise and temporal inconsistency inherent in acquisition of depth data means there is no one-to-one correspondence between sets of 3D points in consecutive time instants. In this paper, instead of coding 3D points (or meshes) directly, we propose to represent an object's 3D geometry as a collection of tile images. Specifically, we first place a set of image tiles around an object. Then, we project the object's 3D geometry onto the tiles that are interpreted as 2D depth images, which we subsequently encode using a modified multiview image codec tuned for piecewise smooth signals. The crux of the tile image framework is the "optimal" placement of image tiles—one that yields the best tradeoff in rate and distortion. We show that if only planar and cylindrical tiles are considered, then the optimal placement problem for K tiles can be mapped to a tractable piecewise linear approximation problem. We propose an efficient dynamic programming algorithm to find an optimal solution to the piecewise linear approximation problem. Experimental results show that optimal tiling outperforms naïve tiling by up to 35% in rate reduction, and graph transform can further exploit the smoothness of the tile images for coding gain.

Index Terms— 3D geometry compression, multiview image coding

1. INTRODUCTION

The advent of depth sensing technologies like Microsoft Kinect[1] means (partial) 3D geometries of objects in a dynamic scene can now be captured with relative ease. If an object's geometrical information can be accurately and compactly represented at the sender for network transmission, then at the receiver it can enable a spectrum of 3D imaging applications, such as virtual image synthesis of the object from any freely chosen viewpoint. Thus compact representation of an object's 3D geometry is an important research problem; this is the focus of this paper.

Unlike computer-generated objects, dynamic 3D geometry periodically captured by depth sensors is subject to acquisition noise and temporal inconsistencies, which means there is no one-to-one correspondence in 3D points and edges between neighboring frames in time. Called *time-varying meshes* (TVM) in [1, 2, 3], the authors proposed to code TVM directly using predictive techniques similar to ones used in video compression algorithms like H.264 [4]. This is a difficult proposition, because: i) 3D meshes typically undergo more complicated transformations over time than the simple translational motion model assumed in motion prediction in video coding; and ii) lack of one-to-one correspondence in 3D points across frames means exact match at patch-level might not exist at all even if more complex motion models (which entail significant computation costs) are introduced.

In this paper, instead of coding 3D points (or meshes) directly, we propose an alternative to compactly represent the geometry of 3D objects using the concept of *image tiling*. Specifically, we first project the object's 3D geometry as images onto a set of carefully placed tiles surrounding the object, then encode the *tile images* using a multiview image / video codec like [5] tuned for piecewise smooth signals (*e.g.*, incorporating tools like *graph transform* (GT) [6, 7]). At the receiver, the decoded tile images are projected back into 3D space to reconstruct the object's geometry. The key to a compact yet accurate representation in our image tiling framework is the appropriate selection of image tiles for a given object. We show that by restricting the tile types considered to be planar and cylindrical tiles only, the optimal selection of tiles—the best-fitting tile combination given a representative cross section of the object—maps to a tractable piecewise linear approximation problem. We propose an efficient dynamic programming algorithm to solve the piecewise linear approximation problem. To the best of our knowledge, *we are the first in the literature to address the optimal image tiling problem* and provide a computation-efficient solution. Experimental results show that optimized image tiling can outperform naïve tiling by up to 35% in rate reduction, and GT can further exploit the smoothness of the tile images for coding gain.

The outline of the paper is as follows. We discuss related works in Section 2. We then formulate the image tile selection problem in Section 3. Finally, we present experimental results and conclusion in Section 4 and 5, respectively.

[1] http://www.microsoft.com/en-us/kinectforwindows/

2. RELATED WORK

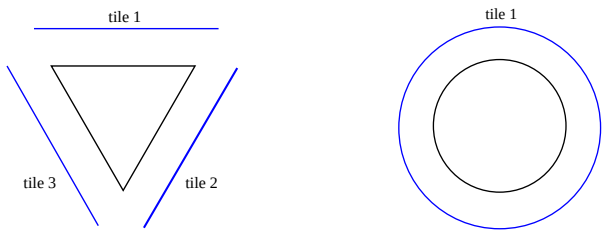
Previous work on compression of *static* 3D geometry is extensive. [8] proposed to compress triangular meshes directly through prediction of vertices and connectivity. [9] proposed a progressive coding scheme based on over-complete expansions by first projecting a 3D object onto a 3D sphere. However, as discussed in the Introduction, the time-varying aspect of TVM brings new challenges that are not addressed in these works.

In [1, 2, 3], TVM in each frame is first divided into patches, then a patch-based matching procedure is used to exploit inter-frame correlation. The residual information is coded using scalar or vector quantization. In contrast, we pursue an *image-based* approach, where 3D geometry is first projected to tiles before the tile images are coded using a multiview image codec. An image-based approach means motion compensation well understood in video coding can be directly applied to reduce temporal redundancy[2].

Depth map compression has been studied in the context of *texture-plus-depth* representation of a 3D scene [10], where texture maps (color images) and depth maps (per-pixel distance between objects in the 3D scene and capturing camera) from multiple closely spaced viewpoints are captured and encoded at the sender for virtual view synthesis via *depth-image-based rendering* (DIBR) [11] at the receiver. Because depth maps have unique signal characteristics such as sharp edges and smooth interior surface, coding tools such as *graph transform* (GT) [6, 7] (for coding of smooth surfaces) and *arithmetic edge coding* (AEC) [12] (for lossless coding of sharp object boundaries) have been proposed. In our codec implementation for coding of multiple tile images, we incorporated both GT and AEC for optimal coding performance, albeit the focus of this paper is the formulation and algorithm development for the optimal image tile selection problem.

3. IMAGE TILES SELECTION

3.1. System Overview



(a) linear tile (b) circular tile

Fig. 1. Examples of linear and circular tiles.

We first describe the overall framework to represent an object using multiple depth images that are projections of the object's 3D geometry on selected tiles. The actual projected images on tiles can be coded using either DCT-based or GT-based multiview image codec.

Given the geometry of an object in 3D mesh, we first place a set of K tiles surrounding the object, onto which the object's 3D geometry is projected as 2D images. By projection, we mean that for each pixel location on the tile, we trace an orthogonal line from the tile surface until we hit the surface of the object, upon which we record the distance (depth) as the pixel value. Each resulting depth image on a tile is a 2D impression of the 3D geometry; the goal is to identify the optimal set of 2D impressions for a given object.

Intuitively, the best placements of tiles—in terms of accuracy in representing the object's geometry—are the ones that are parallel to the surface of the object. As simple examples, consider only the *cross sections* of two 3D objects in Fig. 1, which are triangle and circle in (a) and (b), respectively. For Fig. 1(a), if we place three tiles parallel to the three sides of the triangle, regularly sampled voxels on each side of the object can potentially be captured by the same number of evenly spaced pixels on the corresponding tile. (Evenly spaced pixels in a 2D grid on the tile becomes an image for coding.) If the three tile pixel lines are then losslessly coded, then the exact same triangle cross section (in terms of the original voxel samples) can be reproduced at the decoder, resulting in zero distortion. Similarly, in Fig. 1(b) we see a circular tile that is ideal for an object's circular cross section. Given this developed intuition, we next describe how we find K tiles that best match the object's surface.

3.2. Cross Section Selection

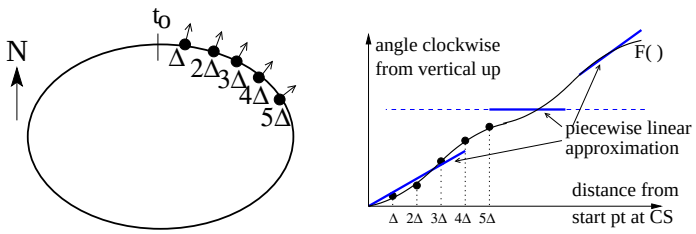
To simplify the tile selection problem, we first select a *representative 2D cross section* from the object's 3D geometry, on which we perform a best-fitting procedure given K tiles. We identify the representative cross section as follows. For a given 3D geometric object, we first identify an axis in 3D space where the object is longest; *i.e.*, if the object's 3D points are projected onto this dimension, the range from maximum value to minimum is the largest. As an example, if the 3D object is a person standing up, the longest dimension would be the vertical axis (range will be his height). We denote this axis as the *principal axis* or simply z*-axis*.

Given the chosen principal axis, we next identify a 2D plane of the object orthogonal to the axis that has the *largest* cross section. In the previous example, this could be a cross section of the person's belly. We perform our 2D optimization for tiling on this representative cross section, as described in the next section.

3.3. Tile Selection

Given a 2D representative cross section, we first define a *surface normal function* (SNF), which is the degree (in radian) of

[2]For brevity, in this paper we demonstrate the efficacy of image tiling only for static 3D objects. Nonetheless, we expect the extension of our image tiling approach to TVM to be relatively straight-forward.



(a) cross section (b) surface normal function

Fig. 2. Example of cross section and surface normal function.

the surface normal, computed clockwise from the vertical up direction, as one moves clockwise around the representative cross section. An example of a cross section and its corresponding SNF $F(\,)$ is shown in Fig. 2.

As discussed earlier, our goal is to select tiles to approximate the representative cross section as closely as possible. However, if too many tiles or tiles of too complicated shapes are deployed, then the descriptions of the tiles themselves would require too many coding bits, leading to large representation size. *Thus the goal is to select a limited number of tiles, each of which required few bits for description, to approximate the cross section as closely as possible.*

Given the above considerations, we will use only two kinds of 2D tiles to approximate the representative cross section in this paper: i) *linear tiles*, ii) *circular tiles*. When considering 3D space, linear and circular tiles extends to *planar* and *cylindrical* tiles respectively along the z-axis. Assuming a change of coordinate is performed so that x and y are axes orthogonal to the principal axis z, the location of these two kinds of tiles can be described using simple equations:

$$ax + y = b \quad (1)$$

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \quad (2)$$

In particular, each tile can be described by at most three parameters: (a, b) for a planar tile, and (x_0, y_0, r) for a cylindrical tile. Thus, the cost of describing each tile is negligible. Further, one can map the pixels in each tile image easily onto a 2D grid for conventional image / video coding. Note that though in general a tile can be located any distance from the object's surface, for simplicity we will assume in this paper that a tile is at average distance ξ away from the surface it is approximating, for suitably chosen ξ.

When a linear or circular 2D tile is mapped to the surface normal (SN) domain, they become a *constant* or *linear* function, respectively. Given only linear or circular tiles are used for approximating the representative cross section, the tile selection problem becomes essentially a *piecewise linear approximation* problem in the SN domain, where we constrain the number of pieces used to be K. We assume that the value K—*i.e.* the number of tiles we used—affects directly the overall rate of the coding system. Hence, for a given K, we find the best possible piecewise linear function to approximate the cross section's SNF—a proxy for distortion. By adjusting K we can induce a rate-distortion (RD) tradeoff.

3.3.1. *Objective Function*

Mathematically, given K we can define our objective function as follows. First, let the representative cross section's SNF be $F(\,)$, and the voxel sampling period on its surface be Δ (with N total samples). Let the circular tile k be represented as a linear function in SN domain as $C_k(t) = m_k t + h_k$, where m_k and h_k are the slope and y-intercept, respectively. For linear tile, its representation in SN domain is simply $L_k(t) = h_k$. Let the boundary between neighboring pieces k and $k+1$ be $\gamma_k \Delta$. The distortion function—l_2-norm between $F(\,)$ and its piecewise linear approximation—can then be written as:

$$D = \sum_{k=1}^{K} \sum_{i=\gamma_{k-1}+1}^{\gamma_k} |F(i\Delta) - \alpha_k C_k(i\Delta) - (1 - \alpha_k) L_k(i\Delta)|^2 \quad (3)$$

where $\gamma_0 = 0$ and $\gamma_K = N - 1$ is the last voxel sample on the cross section, and α_k is a binary variable to denote if a circular or linear tile is used for the kth piece.

3.3.2. *Problem Constraints*

We consider the following constraints for our optimization problem. First, the right boundary $\gamma_k \Delta$ for the kth piece must come before the next boundary $\gamma_{k+1}\Delta$ in the approximation function, hence

$$\gamma_k < \gamma_{k+1}, \qquad 1 \le k \le K - 1 \quad (4)$$

Second, though in theory any radius r can be used to describe the circular tile in (2), a very large radius will create numerical instability when mapping tile pixels to the object surface. Thus, we will lower-bound the choice of slope m_k for a circular tile as follows:

$$m_k \ge m_{\min}, \qquad 1 \le k \le K \quad (5)$$

Finally, as described earlier, α_k is constrained to be a binary variable and can take on only one of two values:

$$\alpha_k \in \{0, 1\} \quad (6)$$

3.3.3. *Dynamic Programming Algorithm*

The optimization is now well defined:

$$\min_{\{\alpha_k, m_k, h_k, \gamma_k\}} D \quad (7)$$

such that constraints (4), (5) and (6) are satisfied.

The optimization problem (7) can be solved optimally and efficiently using *dynamic programming* (DP). Let $D(i, k)$ be the minimum distortion for voxel samples i to N, if k constant / linear pieces can be used for approximating SNF $F(\,)$. If a single piece is used to approximate samples i till j (resulting in distortion $d(i, j)$), then there will be one fewer pieces $k - 1$

for the remaining samples $j+1$ to N. Mathematically, we can write:

$$D(i,k) = \begin{cases} \min\limits_{j=i,\ldots,N} d(i,j) + D(j+1,k-1) & \text{if } k \geq 2 \\ d(i,N) & \text{o.w.} \end{cases} \quad (8)$$

$d(i,j)$ is the distortion if a piece k is used to approximate voxel samples i to j. We can solve for the optimal tile variables α_k, m_k and h_k using *linear regression* [13]. Specifically, we first solve for the best-fit m_k^* and h_k^*:

$$\begin{aligned} m_k^* &= \frac{\overline{\theta_k t_k} - \bar{\theta}_k \bar{t}_k}{\overline{t_k^2} - (\bar{t}_k)^2} \\ h_k^* &= \bar{\theta}_k - m_k^* - \bar{t}_k \end{aligned} \quad (9)$$

where $\bar{t}_k$ and $\bar{\theta}_k$ are the average surface distance and surface normal angle respectively, $\overline{t_k^2}$ is the average of the square distance, and $\overline{\theta_k t_k}$ is the average of the product of distance and angle, for voxel samples i to j.

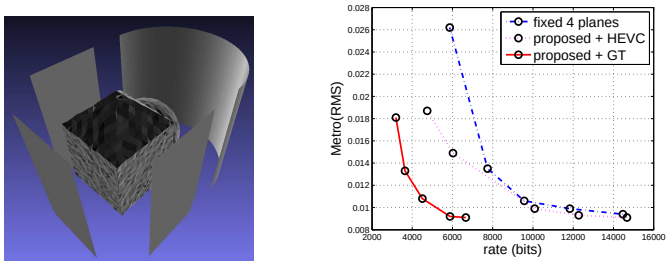
If the best-fit slope m_k^* is smaller than $m_{\min}$, we then compare the solutions when slope is $m_{\min}$ (circular tile) and when slope is 0 (linear tile). The solution with the smaller square error with respect to $F(\,)$ will be chosen.

The complexity of the DP algorithm (8), with initial call $D(1,K)$, can be analyzed as follows. The size of of the DP table to contain solutions to sub-problems $D(i,k)$ is $O(NK)$. To compute each entry in the DP table, the operation in (8) is $O(N)$. Hence the complexity of the algorithm is $O(N^2K)$.

4. EXPERIMENTATION

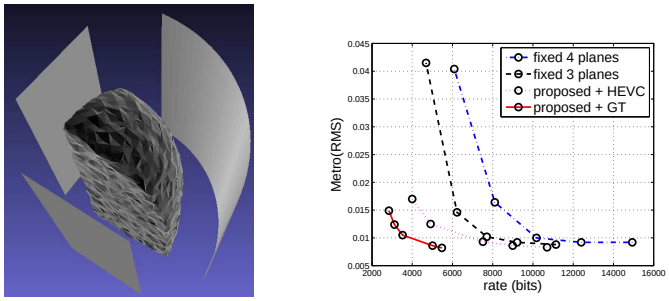
To demonstrate the value of optimizing tile placement, we conducted the following experiment. To find the optimal tile placement for a given static geometry, we use our proposed DP algorithm discussed in Section 3.3.3 for a given budget of tiles K. Fig. 3(a) illustrates the optimal tile placement for the 3D mesh when $K=4$, while a naïve scheme uses four planar tiles. The prediction structure is IPPP and the quantization parameters are 5, 10, 15, 20, and 25. The generated depth maps are coded using either DCT based encoder (HEVC) or GT based encoder[3]. Fig. 3(b) shows the RD performance of the two schemes, where maximum root mean square (RMS) of the metro distance [14] is used to measure 3D reconstruction distortion. The performance curve of our proposed method is the convex hull of operational points obtained by varying QP and the number pieces K used for approximation of SNF. We see that our proposed method outperformed the naïve method at all bitrates.

Fig. 4 shows the performance of another 3D mesh. The 3-tile competing scheme is obtained by replacing cylindrical tile with a planar tile. The 4-tile competing scheme replaces the cylinder tile by two orthogonal planar tiles, *i.e.*, the four tiles face North, East, South and West direction, respectively. From both experiments, we found that our proposed tile selection scheme outperforms the naïve tile placements. The gain is most pronounced at low rate, where up to 35% bit rate reduction can be observed. We observe also that more gain is achieved by using GT, where piecewise smooth signals can be represented more sparsely compared to DCT.



(a) selected tiles (b) RD performance

Fig. 3. Result for the first 3D mesh.



(a) selected tiles (b) RD performance

Fig. 4. Result for the second 3D mesh.

5. CONCLUSION

We presented an image tiling framework for representation of an object's dynamic 3D geometry, where the object's 3D mesh is first projected as 2D images to a set of carefully placed image tiles, before a multiview image codec tuned for piecewise smooth signals is deployed for coding of tile images. The key to a compact yet accurate representation is the selection and placement of image tiles. We showed that if only planar and cylindrical tiles are considered, the optimal tile placement problem (a finite tile set that best matches the object's representative 2D cross section) can be mapped to a piecewise linear approximation problem. The approximation problem can be subsequently solved efficiently using a dynamic programming algorithm. Experimental results show that optimal tile placements can outperform naïve tile placements by up to 35% in rate reduction.

[3]Implementation of GT used was found here: http://biron.usc.edu/∼kumarsun/Codes/GraphTransform_Matlab.zip

6. REFERENCES

[1] S. Nakagawa, T. Yamasaki, and K. Aizawa, "Deformation-based compression of time-varying meshes for displaying on mobile terminals," in *3DTV-Conference*, Tampere, Finland, June 2010.

[2] T. Yamasaki and K. Aizawa, "Patch-based compression for time-varying meshes," in *IEEE International Conference on Image Processing*, Hong Kong, September 2010.

[3] T. Yamasaki and K. Aizawa, "Bit allocation of vertices and colors for patch-based coding in time-varying meshes," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

[4] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003, vol. 13, no.7, pp. 560–576.

[5] I. Daribo, G. Cheung, T. Maugey, and P. Frossard, "RD optimized auxiliary information for inpainting-based view synthesis," in *3DTV-Conference*, Zurich, Switzerland, October 2012.

[6] G. Shen, W.-S. Kim, S.K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

[7] W. Hu, G. Cheung, X. Li, and O. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[8] G. Taubin and J. Rossingnac, "Geometric compression through topological surgery," in *ACM Transactions on Graphics*, 1998, vol. 17, pp. 84–115.

[9] I. Tosic, P. Frossard, and P. Vandergheynst, "Progressive coding of 3-D objects based on overcomplete decomposition," in *IEEE Transactions on Circuits and Systems for Video Technology*, November 2006, vol. 16, no.11, pp. 1338–1349.

[10] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *IEEE International Conference on Image Processing*, San Antonio, TX, October 2007.

[11] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila, "View synthesis techniques for 3D video," in *Applications of Digital Image Processing XXXII, Proceedings of the SPIE*, 2009, vol. 7443 (2009), pp. 74430T–74430T–11.

[12] I. Daribo, G. Cheung, and D. Florencio, "Arithmetic edge coding for arbitrarily shaped sub-block motion prediction in depth video coding," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[13] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[14] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno, "Metro: Measuring error on simplified surfaces," in *Computer Graphics Forum*, 1998, vol. 17, pp. 167–174.