JAIST Repository
https://dspace.jaist.ac.jp/

Title	Mosaic Image Generation using 3D Models
Author(s)	Noda, Takahiko; Miyata, Kazunori; Kimura, Eriko; Kawai, Naoki
Citation	ASIAGRAPH 2009 Proceedings: 110-115
Issue Date	2009-10-23
Type	Conference Paper
Text version	author
URL	http://hdl.handle.net/10119/11440
Rights	It is posted here by permission of the Virtual Reality Society of Japan. Takahiko Noda, Kazunori Miyata, Eriko Kimura, Naoki Kawai, ASIAGRAPH 2009 Proceedings, 2009, pp.110-115.
Description	

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

ASIAGRAPH 2009 PROCEEDINGS

3 次元モデルからのモザイク画生成法

Mosaic Image Generation using 3D Models

野田貴彦/北陸先端科学技術大学院大学,宮田一乗/北陸先端科学技術大学院大学,木村絵里子/大日本印刷(株),河合直樹/大日本印刷(株)
Takahiko Noda/JAIST, Kazunori Miyata[1]/JAIST, Eriko Kimura/DNP Co., Ltd., Naoki Kawai/DNP Co., Ltd.
*[1]miyata@jaist.ac.jp

Abstract: Mosaic is the art of creating images by arranging small pieces of stone, tile or other material. Many conventional methods are used to generate mosaic images from 2D images. In contrast, this paper proposes a method of mosaic image generation using 3D models. This method consists of the following steps. First, the method calculates normal vectors of surface and depth data from 3D models. Then the feature data of the 3D model, such as edges and ridges, are calculated from these data. Next, the guidelines to control the arrangement of tiling are calculated. Finally, tiles are placed along the guide lines. This method has two advantages; 1) It is possible to render a mosaic image from multiple viewpoints. 2) This method can generate a mosaic image which represents three-dimensional features, such as surface ridges and wrinkles, accurately.

Keywords: Mosaic Image, Three-Dimensional Graphics, Ridge

1. Introduction

We usually catch sight of mosaics in architecture, pavements, book covers, and so on, in our daily life. A mosaic is a picture or pattern produced by arranging together small pieces of stone, tile or other material. It can readily be imagined that it takes a lot of time and requires highly developed skills to make an actual mosaic.

The mosaics that we create almost always go through the following process; 1) Design: Make a sketch of the idea to be created, and draw guide lines to place the pieces. 2) Material Preparation: Collect pieces of tile or glass to fill the area. You can also create pieces from a larger piece by using nippers or a hammer. 3) Arrange the pieces in the correct order referring to the guide lines so that together they make the intended image. Once they are arranged correctly, they are attached to the base, such as a wall or pavement. Figure 1 shows examples of mosaic in progress.

(a) sample 1[Tumminello] (b) sample 2[Humby]

Figure 1: Samples of mosaic in progress

2. Related Work

There are many computational methods to generate mosaic images from 2D images, including photomosaic methods [Silvers97, Finkelstein98, Kim02]. Most photo retouching software, e.g. Adobe Photoshop and GIMP, also have the function of creating mosaic images.

The main issue in computational mosaics is how to fill a specified area with small patches.

Hausner proposed a method for simulating decorative tile mosaics using centroidal voronoi diagram (CVD) [Hausner01]. CVD can locate each site of a diagram at the mass-center of its region. The method is adapted to place tiles along the specified direction field, consequently the mosaic images created using this method represent fine flows of tiling. Elber et al. reported a mosaic creation method which lays rows of tiles along the feature curves given for a source image [Elber03]. Parallel curves to a given curve are calculated as offsets of the given curve, then the tiles are placed so that they are tangent to the curve along one edge of the tile, while packed as tightly as possible.

Kojima et al. proposed an image mosaicing method by partitioning the input image with quadrilateral patches [Kojima06]. This method first generates a mosaic respecting global features such as image gradation, then incorporates local features into each underlying region referring to the existing global features. Mould presented a method of converting an image into a stained-glass version of that image [Mould03]. This method tessellates an input image, then smoothes the boundaries of the initial image segments by means of erosion and dilation operators.

Dobashi et al. devised a method for creating mosaic images using Voronoi diagrams [Dobashi02]. The diagrams are optimized so that the difference between the original image and Voronoi image is minimized by moving the sites in the diagrams. They also accelerate the process using graphics hardware.

Most of these conventional methods generate mosaic images from 2D images. Therefore if a user wants to change the viewing angle of the objects or composition of an image, it is necessary to redesign and reproduce the source image. In contrast, this paper generates a mosaic image from 3D models. This approach has two advantages; 1) It is possible to render a mosaic image from multiple viewpoints. 2) The resulting image represents three-dimensional features, such

as surface ridges and wrinkles, accurately. These features are sometimes not depicted in 2D images, and it is hard to extract them from the source image.

3. Algorithm

This chapter describes the overview of the algorithm, and then explains each process in detail.

Our algorithm for generating a mosaic image consists of the following five steps;

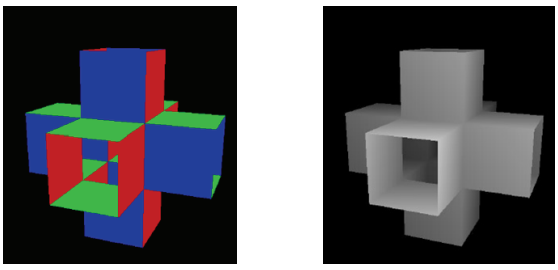
(1) Primary data calculation
(2) Feature line extraction
(3) Guideline generation for tiling
(4) Tile arrangement
(5) Rendering

The procedure first computes primary data, such as normal vectors of surface and depth data, from 3D models. Then the procedure extracts features such as edges and ridges of the object. These features are used to represent the three-dimensional features of the model accurately. Next, the guidelines to control the arrangement of tiling are calculated using extracted feature data. Concurrently, the flow vector field which specifies the direction of each tile is also generated. Finally, tiles are placed along the guidelines referring to the flow vector field, and then the intended mosaic image is obtained.

3.1. Primary Data Calculation

The procedure first calculates primary data, which are used in the next step for feature data extraction, from a 3D model.

The system loads a 3D model and renders it onto a virtual screen, that is a frame buffer, by means of conventional 3D API, such as Microsoft DirectX. The normal vectors of surface and depth data of the object are then calculated using graphics hardware. Figure 2 shows an example of calculated primary data. Figure 2(a) shows a sample image, called a normal map in this paper, that represents the normal unit vectors *(x,y,z)* in color value *(R, G, B)*, and Figure 2(b) shows the depth data which is represented in gray scale image, the brighter, the closer. Here, the range of the gray scale image is normalized with the depths of near and far clipping planes in a perspective projection.



(a) normal map (b) depth map

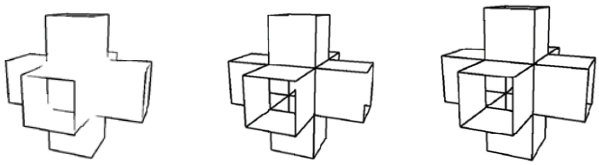
Figure 2: Primary Data of 3D Model

3.2. Feature Line Extraction

In general, edges appear at the boundaries between foreground objects and background space, or form where objects overlap. Our method extracts these edges by applying the Laplacian filter to the depth map. Figure 3(a) shows an example which is generated from Figure 2(b). A crease is not detected in this procedure, theoretically, as shown in this example. The calculated edges are used for separating the foreground and background in a mosaic image.

Ridges appear where the surface curvature is high. This procedure computes the ridges by checking changes of surface normal vector. This process is also performed by applying the Laplacian filter to the normal map. Figure 3(b) shows an example generated from Figure 2(a). The procedure does not detect a line as a ridge where the normal of foreground and background surface are the same, as shown in this example. The obtained ridges are used to represent surface creases and bumpiness.

These two kinds of lines are combined and used as generating guidelines to control the arrangement of tiling. Figure 3(c) shows the result obtained by combining Figure 3(a) and (b).



(a) edges (b) ridges (c) combination

Figure 3: Edge and Ridge Extraction

3.3. Guideline Generation for Tiling

Next, the procedure generates the guidelines for tiling using the extracted feature lines. In this method, all tiles are square and the same size.

The guidelines are a set of lines which are parallel to the feature lines. Each tile is placed so that the center of gravity is on a placement line, the midway between two guidelines, and the angle of its placement corresponds to the flow vector field described in 3.4, as shown in Figure 4.

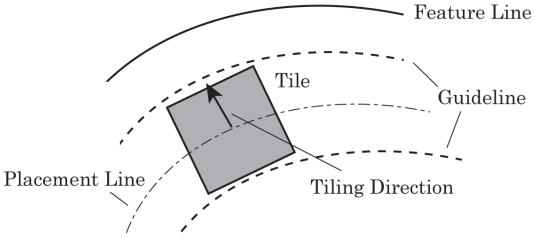


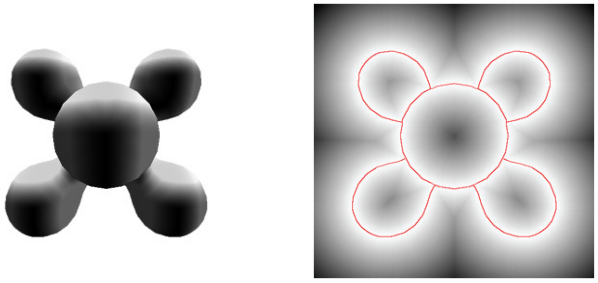
Figure 4: Guidelines for Tiling

3.3.1. Distance Map

For calculating guidelines, the procedure generates a distance map from feature lines. In a distance map, each

pixel value represents the distance from the nearest feature line.

The distance map is obtained as follows; First, the nearest feature line from a point is searched. Here, the nearest feature line is found by scanning the surrounding space in an expanding circle shape. Then, the distance between a point and the found feature line is stored in the distance map. Figure 5 shows an image which represents the distance map as a gray scale image. Here, the red lines are feature lines, and the pixel becomes brighter as it approaches the feature line.

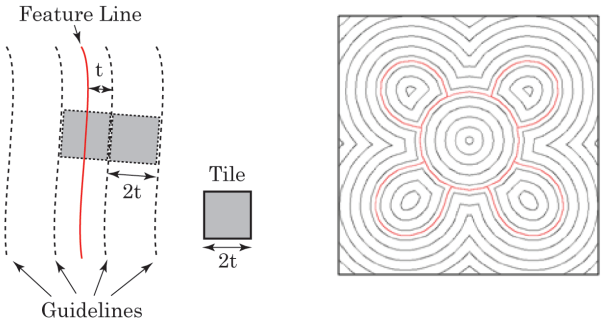


(a) 3D model (b) distance map

Figure 5: Distance Map

3.3.2. Guideline Generation

The guidelines are generated by connecting adjacent points on the distance map which have the same brightness value. The interval between each pair of adjacent guidelines is set at the size of a tile, *2t*, as shown in Figure 6(a). Figure 6(b) shows an example of generated guide lines from Figure 5(b); Here, the red lines are feature lines, and the black lines are guidelines.



(a) guidelines (b) example

Figure 6: Guideline Generation

3.4. Tile Arrangement

Each tile is placed into the position so that the center of gravity of the tile is on the placement line. Besides that, each tile should avoid overlapping each other, or making broad gaps between tiles. The computation cost for placing tiles while avoiding overlaps and gaps is high, therefore an approximated calculation for placing tiles is required. To reduce the computation cost, the procedure calculates locations of tiles by using a representative circle as follows.

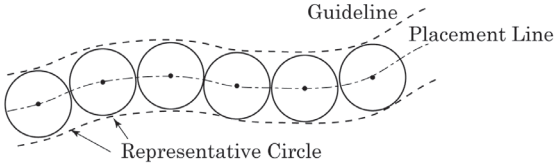
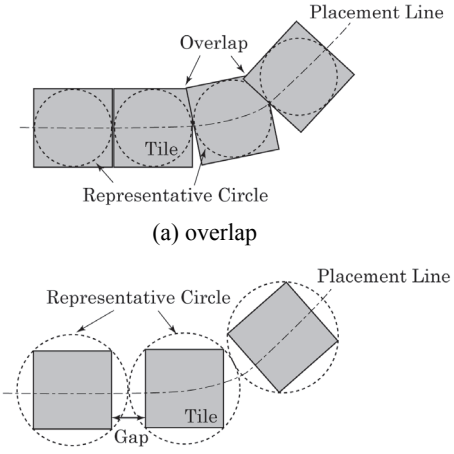


Figure 7: Location of Representative Circles

First, a representative circle, a circle representing a tile, is placed at a certain position on the placement line. Then, the next one is placed as shown in Figure 7. This process is repeated until no more representative circles can be placed on the placement line. The radius of a representative circle, *r*, is specified by the user within the range from *t* to $\sqrt{2}t$, where *t* is half the width of a tile. Some tile may overlap the others if *r* is set at *t*, as shown in Figure 8(a), or make a broad gap if *r* is set at $\sqrt{2}t$, as shown in Figure 8(b).

Then, each representative circle is replaced with a tile. Each tile is placed so that one of its sides is orthogonal to the placement line. This placement angle is calculated in the process of generating a distance map, already described in 3.3.1; When the length of the perpendicular from a point to the guideline is found in that process, the procedure calculates the orthogonal vector to the placement line. Therefore, the perpendicular vector at each point is stored as a flow vector in the process of generating a distance map, and is used for placing the tiles.



(a) overlap

(b) gap

Figure 8: Overlap and Gap

3.5. Rendering

The procedure renders the mosaic tiles, which are arranged using the obtained guidelines and flow vectors. Each representative circle is replaced with a square. The precise placement of each square is determined by the flow vector which is located at the center of the representative circle.

4. Implementation and Results

Our system converts a 3D model into a mosaic image in the following five steps, illustrated in Figure 9. We use texture

image data to store the result and to transfer data from GPU to CPU. Here, each color channel of texture data, *R*, *G*, *B*, *A*, has 32-bit floating-point data.

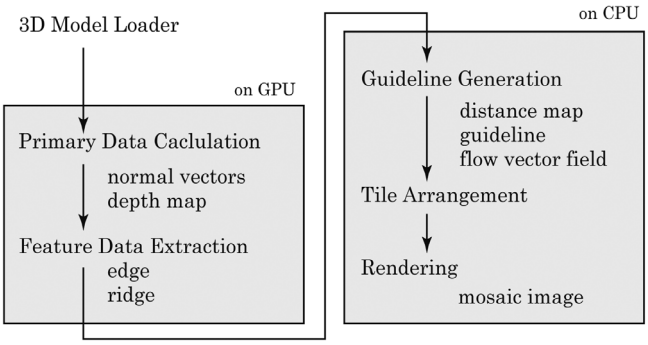


Figure 9: Process Flow

Primary Data Calculation: The system first loads the 3D model, and calculates primary data, normal vectors and depth map, using graphics hardware. Each element of obtained normal vector, *(x, y, z)*, is stored in a color channel, *(R, G, B)*, respectively, and the depth value is stored in an alpha channel, *A*, of texture data.

Feature Data Extraction: The system extracts feature data, edges and ridges, from primary data using a pixel shader on DirectX API. The extracted data are stored as texture data, then the texture data is transferred to a CPU.

Guide Line Generation: The system generates a distance map from the feature data. Then, the guidelines are generated from the distance map. Concurrently with this process, flow vectors are calculated and stored to a certain buffer.

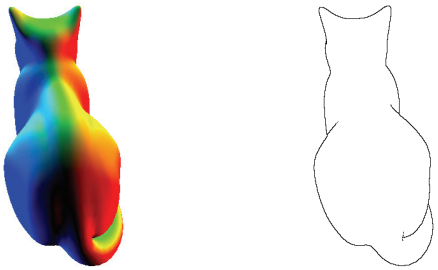
Tile Arrangement: The system determines the position and exact placement angle for each tile by referring to the guideline and flow vector.

Rendering: A mosaic image is rendered in accordance with the obtained tile arrangements.

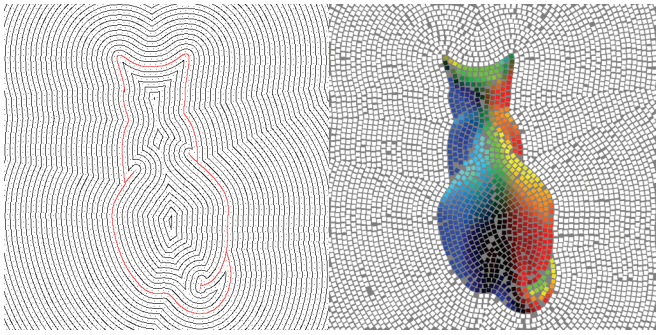
Figure 10 shows the result; Figure 10(a) is generated by conventional shading method. The surface of the object is colored with its surface normal vector. Figure 10(b) shows the extracted feature data, and Figure 10(c) represents the generated guidelines. Figure 10(d) is the obtained mosaic image. The color of each tile is determined by referring to the color values in Figure 10(a).

Figure 13 shows other results. Here, the color of each tile is determined in the same manner as in Figure 10. The 3D model of Figure 13(1a) has smooth and monotonic curved surface, and both of the rendered mosaic images represent these features accurately. In the results for the 3D model with complex curved surfaces, tiles are also well arranged according to the features of the surface. However, there are some noticeable gaps between tiles.

This implementation requires about 3 minutes to calculate a 512 x 512 pixel mosaic image with Intel Core2 Duo 2.2 GHz processor and 2GB RAM. Most of the processing time, about 90% of total processing time, is for the process of guideline generation, especially for searching for the nearest guideline from a point.



(a) normal (b) feature data



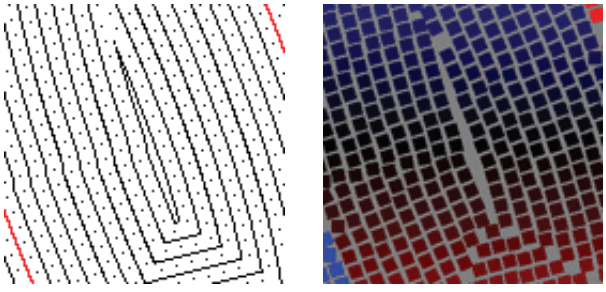
(c) guidelines (d) rendered mosaic

Figure 10: Result

5. Discussion

The mosaic images generated by our algorithm can represent the features of a 3D model, and we can observe three-dimensional surface features in the obtained image.

The main unsolved problem is that there are some noticeable gaps between tiles as shown in Figure 11. Here, in Figure 11(a) a black line is a guideline and a small black dot indicates the center of gravity of a tile to be placed. In some cases, a narrow or small island, whose width is less than the width of a tile, remains after the process of guideline generation, as shown in Figure 11(a). No tile can be placed in this island region, therefore a noticeable gap is observed in that area.



(a) guidelines (b) mosaic image

Figure 11: Noticeable Gap

Sometimes, a tile is not placed near a sharp corner of the guideline, because there is no representative circle on the placement line, as shown in Figure 12.

A relaxation algorithm, such as particle simulation for representative circles, or adaptive size fitting for tile, is one possible solution for this problem, but the flow of tile might be broken after the relaxation process. We need to incorporate a number of additional heuristics to solve this problem, which we plan to address in the future.

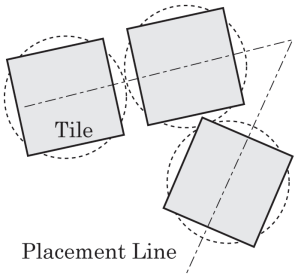


Figure 12: Gap near a Corner

6. Conclusion

We presented an automatic method for producing mosaic images from a 3D model. Our system first calculates feature data of the model, and then the guidelines and directions for tiling are generated. The system finally arranges the tiles by referring to the guidelines and the flow vectors.

This work demonstrates the effectiveness of generating mosaic images directly from a 3D model, to preserve its three-dimensional features.

In the future, we would like to solve the problem of noticeable gaps, and also would like to apply this approach to represent surface attributes, such as texture, to generate more attractive mosaic images.

References

[Tumminello] Tumminello, S., Descrizione e tecnica utilizzata nei mosaici del Duomo di Monreale, http://www.parrocchie.it/monreale/sscrocifisso/italia/mosaici.htm

[Humby] Humby, R., The joy of shards mosaics resource: The reverse method of making mosaics, http://www.thejoyofshards.co.uk/projects/mermaid/reverse.shtml

[Silvers97] Silvers, R. and Hawley, M., 1997. Photomosaics, *New York: Henry Holt*.

[Finkelstein98] Finkelstein, A. and Range, M., 1998. Image Mosaics, *Artistic Imaging and Digital Typography, LNCS, No. 1375, Heidelberg:Springer-Verlag*.

[Kim02] Kim, J. and Pellacini, F., 2002. Jigsaw Image Mosaics, *Proc. of SIGGRAPH 2002*, 657–664.

[Hausner01] Hausner, A., 2001. Simulating Decorative Mosaic, *Proc. of SIGGRAPH 2001*, 573–580.

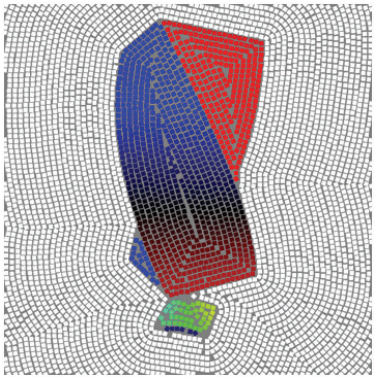
[Elber03] Elber, G. and Wolberg G., 2003. Rendering traditional mosaics, *The Visual Computer*, 19, 1, 67–78.

[Kojima06] Kojima K., Takahashi S., and Nishita T., 2006. Creating quadrilateral mosaics from image topographic features, *APGV '06: Proc. of the 3rd Symposium on Applied Perception in Graphics and Visualization* , 144.

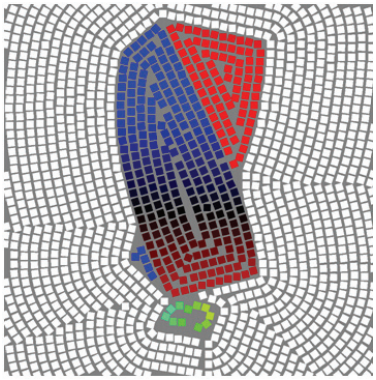
[Mould03] Mould D., 2003. A Stained Glass Image Filter, *Proc. of the 14th Eurographics Workshop on Rendering* , 20–25.

[Dobashi02] Dobashi Y., Haga T., Johan H., and Nishita T., 2002. A Method for Creating Mosaic Images Using Voronoi Diagrams, *Proc. of Eurographics 2002* , 341–348.

(1a) Shading Image

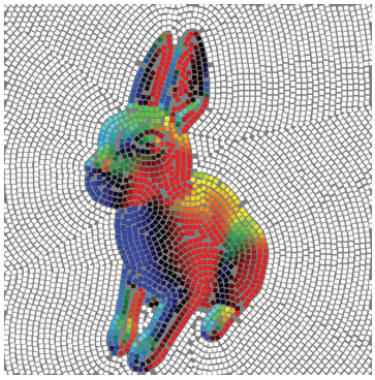


(1b) Fine Mosaic

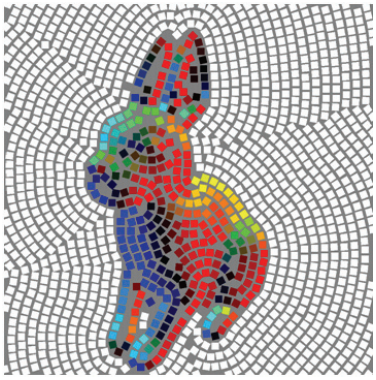


(1c) Coarse Mosaic

(2a) Shading Image

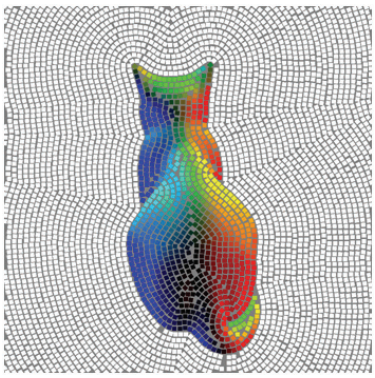


(2b) Fine Mosaic

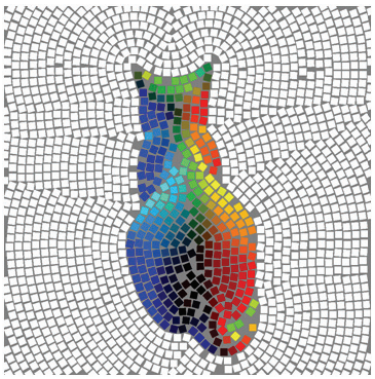


(2c) Coarse Mosaic

(3a) Shading Image

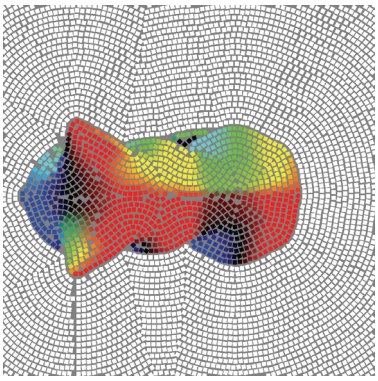


(3b) Fine Mosaic

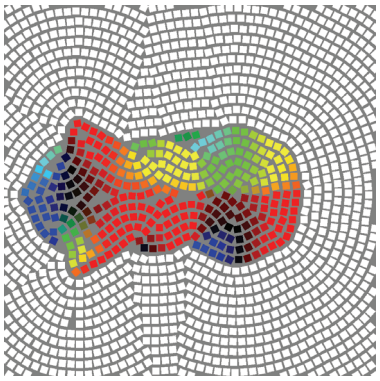


(3c) Coarse Mosaic

(4a) Shading Image



(4b) Fine Mosaic



(4c) Coarse Mosaic

Figure 13: Results