# Displacement Mapping Techniques: Analysis and Comparisons

Gabriel Nobrega de Lima, Harlen C. Batagelo, João Paulo Gois
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Santo André, Brazil
joao.gois@ufabc.edu.br

***Abstract*—Polygonal meshes are commonly used to represent 3D objects. In order to simulate detailed surfaces, a large number of polygons is required. However, the greater the number of polygons, the greater the processing required on the Graphics Processing Unit (GPU), which ends up limiting the quality of the images that can be generated in real-time applications. To overcome such an issue, techniques for real-time displacement mapping allows for simulating realistic 3D objects from a very coarse polygonal mesh. Understanding the importance and the particularities of the displacement mapping methods proposed in the recent years, this work conducts an study that compares the most popular per-pixel displacement mapping techniques, finds out their advantages, disadvantages and evaluates the environments where each one is best applied.**

***Keywords*-Displacement Mapping, GPU, texture mapping, GLSL, mesostructure, normal map, tangent space.**

## I. Introduction

Real-time displacement mapping techniques exploit the flexibility of programmable shading [1], [2] and texturing to map surface details on a coarse surface mesh, in a per-pixel level [3]. Per-pixel displacement mapping techniques work on three levels of structure details: macrostructure, mesostructure and microstructure. The macrostructure stands for the most abrupt details, where they are referenced to be the polygonal meshes. The mesostructure is defined as the geometrical details that are slightly small and only distinguishable according to the viewer position. They are frequently represented by small deformations on the surface. Finally, the microstructure stands for not perceived surface details from distinct viewer positions, such as the microscopic features that define the surface albedo. In general, microstructures are modeled by classic texture mapping [4].

Macrostrucutures, which are used to represent 3D objects with complex details, require a large number of polygons to capture small features. On the other hand, details at the mesostructure level can be represented without the requirement of refined meshes. Per-pixel displacement mapping techniques can model such details by using heightmaps (Fig. 1) and/or normal vector maps (Fig. 2) besides the diffuse map used to model microstructures. They allow to model the appearance of small surface deformations in real-time during fragment shader processing, with significant reduction of computational cost when compared to modelling these details through mesh refinement.

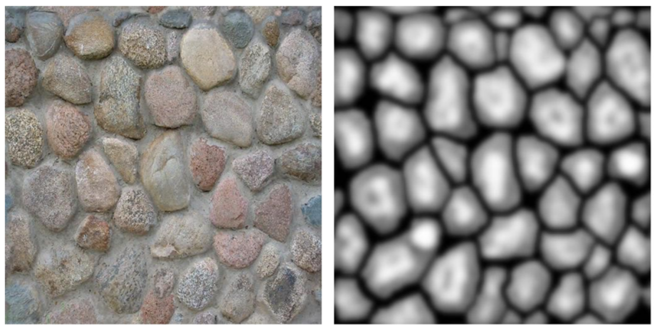


Fig. 1. Diffuse map of a stone texture (left) and a corresponding heightmap (right).

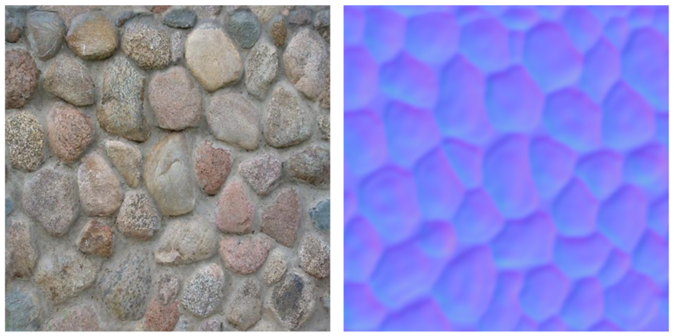


Fig. 2. Diffuse map of a stone texture (left) and a corresponding normal map (right).

An example of a per-pixel displacement mapping technique is depicted in Figure 3. On (a) we have a refined mesh onto which a stone texture is mapped, generating (b). On (c) we have a very coarse mesh (actually, only two triangles), but by using a per-pixel displacement mapping technique we can depict not only the surface colors, but also the geometry details, as shown in (d).

We cite four main abilities that are expected from a good per-pixel displacement mapping technique. They are the ability to simulate: silhouette effects, motion parallax, self-occlusion and self-shadowing. Silhouette effects shows the expected contours that are seen on finer meshes. Motion parallax allows that details close to the viewer move faster than far ones.

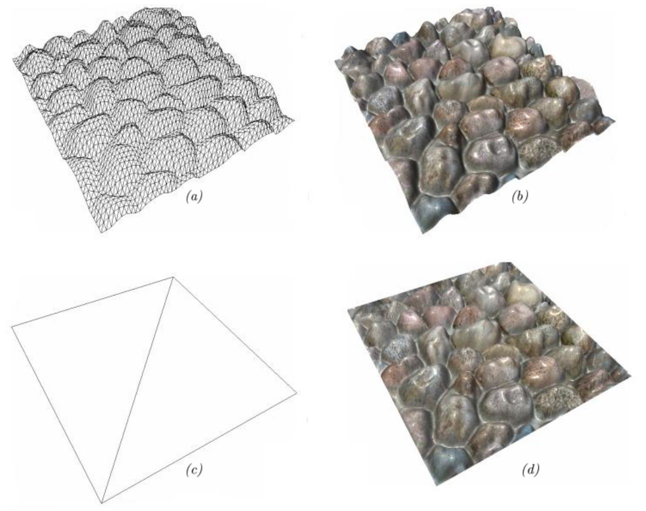


Fig. 3. The power of the per-pixel displacement mapping: on *(a)* a mesh with a large number of polygons receives the straightforward diffuse mapping using a stone texture *(b)*. The mesh on *(c)* has only two triangles, but a per-pixel displacement mapping technique achieves a similar result on *(d)*.

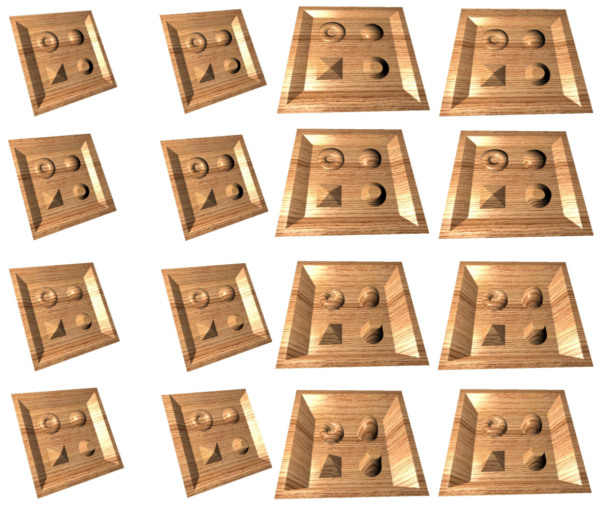


Fig. 4. Test 1 (a) and Test 2 (b): On (a) small angle test whereas on (b) medium angle test. From top to bottom and from left to right: Normal Mapping, Parallax Mapping, Parallax Mapping with Offset Limiting, Iterative Parallax Mapping, Linear Search, Relief Mapping, Parallax Occlusion Mapping and Cone Step Mapping.

Self-occluding allows surface details close to the viewer to occlude far details. Finally, self-shadowing allows that the mesostructure generates shadows on itself.

## II. Methodology

Since each of the per-pixel displacement mapping techniques analyzed here have their own characteristics, they were individually tuned in order to achieve their best results. Thus, the visual quality of the images are compared based on two features: $(i)$ the presence of artifacts or image distortions and $(ii)$ the quality of level of detail of the mesostructure. Details about the techniques are omitted. We suggest to the reader not only the original references along this work but also the survey [3] for further details. Also, a very detailed description of the methods, further comparisons and shader source codes are available in the student's monograph which was written in Portuguese [5].

In our work, the generation of the images was performed with the software Render Monkey[1]. All shaders were written by us in OpenGL Shader Language (GLSL) [6].

The per-pixel displacement mapping methods compared were: Normal Mapping [4], Parallax Mapping [7], Parallax Mapping with Offset Limiting [8], Iterative Parallax Mapping [9], Linear Search [10], Relief Mapping [11], Parallax Occlusion Mapping and [12] Cone Step Mapping [13].

It is worth mentioning that the techniques of Mesh-based Displacement Mapping [14], Bump Mapping [15], [16], Parallax Mapping with Slope Information Search [3] and Binary Search [10] were excluded from our comparisons. Bump Mapping and Mesh-based Displacement Mapping, in this work, are treated only as historical methods. Parallax Mapping with Slope Information was not included in the comparisons as its iterate-based version – Iterative Parallax Mapping – was included. Binary Search was not effective when applied directly, generating artifacts and distortions from most viewpoints. It was disregarded because of its low visual quality.

## III. Analysis & Comparisons

The methods were compared on three types of mesostructure (Figs. 4-6) mapped onto a square defined by two triangles, as in Fig. 3-(c). For each mesostructure, images were taken in small, medium and large angles between the view vector and the surface normal. All techniques were tested on a CPU with 2.4 GHz, 2 GB of RAM and a graphics card NVIDIA GeForce GTX 275 with 898 MB of RAM.

In Tests 1, 2 (Fig. 4) and 8 (Fig. 8) we used a $512 \times 512$ diffuse texture map and $256 \times 256$ normal maps and heightmaps. For Test 1 (Fig. 4-(a)), the view vector makes a small angle with the surface normal. The methods compared here were calibrated so as to present the fewest artifacts or distortions, thus showing their best quality on details of the mesostructure. Linear Search [10] was configured to run up to 80 steps, Iterative Parallax Mapping [9] used 4 iterations, Relief Mapping [10] performed up to 40 steps for linear search and up to 10 steps for binary search, Parallax Occlusion Mapping [12] performed a minimum of 30 and maximum of 60 steps for linear search and Cone Step Mapping [13] performed 20 iterations. For Test 2, shown in Figure 4-(b), the view vector makes a medium angle with the surface normal. In this case, Linear Search was configured to run up to 90 steps, Iterative Parallax Mapping used 4 iterations, Relief Mapping was configured to run up to 20 steps with linear search and 6 with binary search, Parallax Mapping Occlusion performed a minimum of 30 and maximum of 60 steps for linear search and Cone step Mapping performed 20 iterations. For Test 8, shown

[1]http://developer.amd.com/archive/gpu/rendermonkey/pages/default.aspx

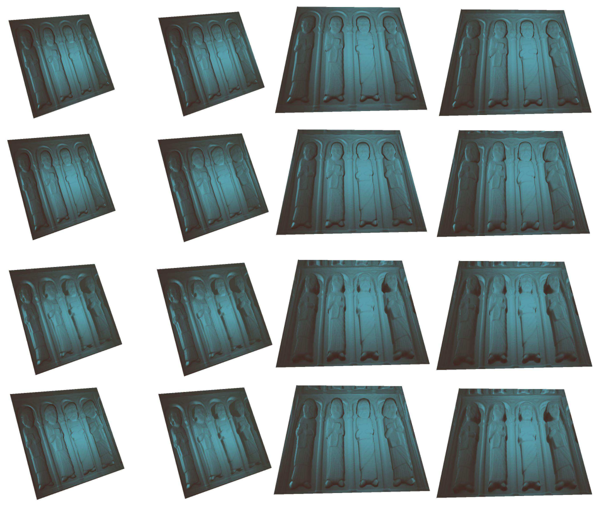


Fig. 5. Test 3 (a) and Test 4 (b): On (a) small angle test whereas on (b) medium angle test. From top to bottom and from left to right: Normal Mapping, Parallax Mapping, Parallax Mapping with Offset Limiting, Iterative Parallax Mapping, Linear Search, Relief Mapping, Parallax Occlusiong Mapping and Cone Step Mapping.

in Figure 8, the view vector makes a large angle with the surface normal. Iterative Parallax Mapping used 6 iterations, Relief Mapping was configured to run up 40 steps with linear search and 6 with binary search, Parallax Occlusion Mapping performed a minimum of 50 and maximum of 200 steps for linear search and Cone Step Mapping performed 20 iterations.

In Tests 3 and 4 (Fig. 5) we used a $512 \times 512$ diffuse map and $256 \times 256$ normal maps and heightmaps. For Test 3 (Fig. 5-(a)), the view vector makes a small angle with the surface normal. Linear Search was configured to run up to 70 steps, Iterative Parallax Mapping was applied to 6 iterations, Relief Mapping run up to 30 steps with linear search and 5 with binary search, Parallax Mapping Occlusion performed a minimum of 30 and a maximum of 60 steps for linear search and Cone step Mapping performed 12 iterations. For Test 4 (Fig. 5-(b)), the view vector makes a medium angle with the surface normal. Linear Search was configured to run up to 70 steps, Iterative Parallax Mapping was applied to 6 iterations, Relief Mapping was configured to run up to 30 steps with linear search and 5 steps with binary search, Parallax Mapping Occlusion performed a minimum of 30 and maximum of 60 steps for linear search and Cone step Mapping performed 12 iterations.

On Tests 5 and 6 (Fig. 6) and 7 (Fig. 7) all textures have a resolution of $1024 \times 1024$. For Test 5 (Fig. 6-(a)), the view vector makes a small angle with the surface normal. Linear Search was configured to run up to 60 steps, Iterative Parallax Mapping was applied to 6 iterations, Relief Mapping was configured to run up to 15 steps with linear search and 6 steps with binary search, Parallax Mapping Occlusion performed a minimum of 30 and maximum of 40 steps for linear search and Cone step Mapping performed 10 iterations. For Test 6 (Fig. 6-(b)), the view vector makes a medium angle with the surface normal. Linear Search was configured to run up to 64 steps, Iterative Parallax Mapping was applied to 6 iterations, Relief Mapping was configured to run up to 26 steps with linear search and 6 steps with binary search, Parallax Mapping Occlusion performed a minimum of 10 and a maximum of 60 steps for linear search and Cone step Mapping performed 20 iterations. For Test 7 (Fig. 6-(b)), the view vector makes a large angle with the surface normal. Linear Search was configured to run up to 64 steps, Iterative Parallax Mapping was applied to 6 iterations, Relief Mapping was set to run up to 26 steps with linear search and 6 with binary search, Parallax Mapping Occlusion performed a minimum of 10 and a maximum of 60 steps for linear search and Cone step Mapping performed 20 iterations. For Test 7, shown in Figure 7, the view vector makes a small angle with the surface normal. Iterative Parallax Mapping used 6 iterations, Relief Mapping was configured to run up to 40 steps with linear search and 6 steps with binary search, Parallax Occlusion Mapping performed a minimum of 50 and a maximum of 200 steps for linear search and Cone Step Mapping performed 20 iterations.

For completeness, the frame rates for Tests 1-6 are found in Table I.

Normal Mapping had the poorest level of details in all tests. This is because it simulates the detailing on the surface by only changing the value of the per-pixel normal used in the evaluation of the lighting equation. As it does not allow for shifting texels or traversing a ray as in ray casting, it cannot simulate any of the four abilities that we have cited as expected capabilities for a good per-pixel displacement mapping technique. However, its efficiency was the best among all techniques, with the highest frame rate as shown in Table I.

Parallax Mapping and Parallax Mapping with Offset Limiting generated images with quality and efficiency very similar for all comparisons. In Tests 2, 4 and 6 their results are similar to Normal Mapping, as the viewpoint avoids a large number of surface overlaps in screen space. However, Tests 1, 3 and 5 show that Parallax Mapping and Parallax Mapping with Offset Limiting obtain superior results than Normal Mapping. This is because both techniques simulate the effect of motion parallax [3], which is more evident as the view vector makes large angles with respect to the surface normal. The efficiency of both techniques was quite similar. The only exception was Normal Mapping. One should take into account that Parallax Mapping with Offset Limiting eliminates distortions that often occur for large angles between the view vector and the surface normal.

For all tests, Iterative Parallax Mapping generated results with extrusion emphasis greater than Normal Mapping, Parallax Mapping and Parallax Mapping with Offset Limiting. This is due to the fact that it takes into account the normal map to simulate the effect of motion parallax. Further, Tests 2, 4 and 5 show that the technique generates some distortions, evidenced in Figures 4-(b) and 5-(b). Their rendering efficiency was lower than the previously discussed methods.

For all tests, Linear Search, Relief Mapping, Parallax Oc-

TABLE I
FRAME RATES.

| **Methods** | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 | Test 6 |
|---|---|---|---|---|---|---|
| Normal Mapping | 2890 | 2893 | 2890 | 2850 | 2540 | 2550 |
| Parallax Mapping (PM) | 2888 | 2680 | 2800 | 2525 | 2510 | 2543 |
| PM with Offset Limmiting | 2884 | 2680 | 2804 | 2538 | 2500 | 2530 |
| Iterative PM | 2900 | 2600 | 2790 | 2522 | 2130 | 1163 |
| Linear Search | 1220 | 700 | 1558 | 1031 | 812 | 477 |
| Relief Mapping | 2000 | 2238 | 2516 | 1850 | 1839 | 800 |
| Parallax Occlusion Mapping | 2600 | 2200 | 2306 | 1378 | 1700 | 1000 |
| Cone Step Mapping | 2700 | 2350 | 2580 | 2300 | 2517 | 1200 |

clusion Mapping and Cone Step Mapping presented similar results, but better than Normal Mapping, Parallax Mapping and Parallax Mapping with Offset Limiting. This is due to the fact that these techniques simulate both the effects of motion parallax and self-occlusion. However, the requirement of a large number of iterations to achieve their best visual results made them the most inneficient methods, as shown by the frame rates listed in Table I. Among all methods, Linear Search obtained the worst frame rate because its ray march requires a high number of iterations for removing all the distortions and artifacts. Parallax Occlusion Mapping produces superior images than Linear Search for all tests. Relief Mapping, on the other hand, yielded better frame rates than Parallax Occlusion Mapping for Tests 2, 3, 4 and 5 as they required on average a smaller number of iterations. Cone Step Mapping had the best frame rates when compared to Linear Search, Relief Mapping and Parallax Occlusion Mapping. This is because such technique makes use of a texture that maps void regions through cones, allowing the intersection between the field of view and the elevation of the heighmap to be found quickly.

For each one of tests we can see that Iterative Parallax Mapping, Linear Search, Relief Mapping, Parallax Occlusion Mapping and Cone Step Mapping showed the best image quality results. Normal Mapping, Parallax Mapping and Parallax Mapping with Offset Limiting can generate some details of the mesostructure, but the result is worse than the other techniques. For all tests, the number of iterations used by each technique was carefully tuned in order to generate images with the best quality possible. This meant that most of the distortions and artifacts disappeared during the tests, making it difficult to elect the best method. Therefore, we selected the techniques that have the best trade-off between image quality and efficiency for the execution of Tests 7 and 8 (large angles between the view vector and the surface normal). Such techniques are Iterative Parallax Mapping, Relief Mapping, Parallax Occlusion Mapping and Cone Step Mapping (but excluding its linear search step due to its poor performance).

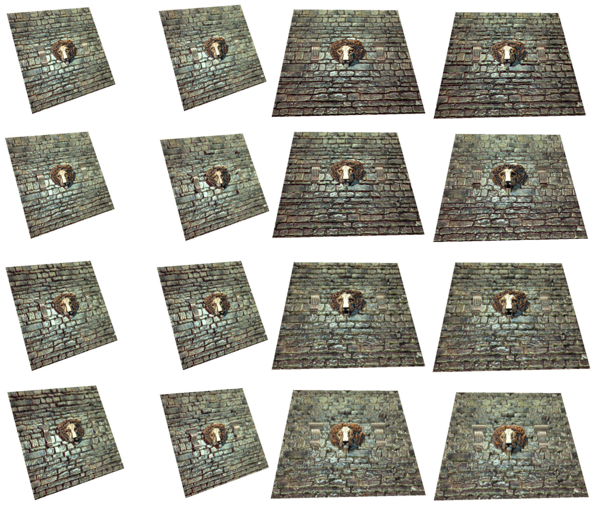
Test 7, shown in Fig. 7, compares Cone Step Mapping, Parallax Occlusion Mapping, Relief Mapping and Parallax Mapping Iterative about large angles between the view vector and the surface normal. All the techniques presented distortions and artifacts. However, they also generated different levels of extrusion without distortion. Iterative Parallax Mapping was the method that allowed the lower extrusion with higher levels of distortion. Relief Mapping achieved similar levels of extrusions to those achieved for Iterative Parallax Mapping, but with less distortion and artifacts. Cone Step Mapping and Parallax Occlusion Mapping produced better and larger extrusions than other techniques. Cone Step Mapping exhibited a lower level of distortion when compared to Parallax Occlusion Mapping. On Test 8 (Fig. 8), all techniques present small distortions and similar extrusions. The only exception was Iterative Parallax Mapping (Fig. 8-(d)).

TABLE II
FRAME RATES.

| Methods | Test 7 | Test 8 |
|---|---|---|
| Iterative Parallax Mapping | 2134 | 2305 |
| Relief Mapping | 775 | 688 |
| Parallax Occlusion Mapping | 355 | 507 |
| Cone Step Mapping | 1120 | 1400 |

Table II shows the frame rates for Tests 7 and 8. In this case, we find that the Iterative Parallax Mapping was the most efficient. However, the visual quality obtained in these tests ended up being the worst. Relief Mapping and Parallax Occlusion Mapping had the lowest frame rates. Relief Mapping obtained good results for Test 8, similar to Cone Step Mapping and Parallax Occlusion Mapping, but in Test 7, Relief Mapping did not allow to generate large extrusions, though the results had not significant distortions. It also resulted in high frame rates, as shown in Table II.

All eight tests as well as their frame rates presented in Tables I and II allowed us to infer which are the best methods. To perform this evaluation, we took into account that, in the trade-off between quality and efficiency, image quality is preferred. In this sense, Cone Step Mapping seems to be the best method implemented in this work, producing images with relatively high quality level along with high frame rates. Secondly, Relief Mapping and Parallax Occlusion Mapping tied for producing approximate images at very distinctly frame rates in some tests. It makes difficult to estimate which approach is the best. Although Iterative Parallax Mapping achieves high frame rates, the image quality was not so good in Tests 2, 4, 5, 7 and 8. Linear Search, though producing high levels of quality, had the worst frame rate among all tests



(a) (b)

Fig. 6. Test 5 (a) and Test 6 (b): On (a) small angle test whereas on (b) medium angle test. From top to bottom and from left to right: Normal Mapping, Parallax Mapping, Parallax Mapping with Offset Limiting, Iterative Parallax Mapping, Linear Search, Relief Mapping, Parallax Occlusiong Mapping and Cone Step Mapping.

TABLE III
RANKING FOR PER-PIXEL DISPLACEMENT MAPPING TECHNIQUES.

| |
|---|
| 1.Cone Step Mapping |
| 2. Parallax Occlusion Mapping e Relief Mapping |
| 3. Iterative Parallax Mapping |
| 4. Linear Search |
| 5. Parallax Mapping with Offset Limiting |
| 6. Parallax Mapping |
| 7. Normal Mapping |

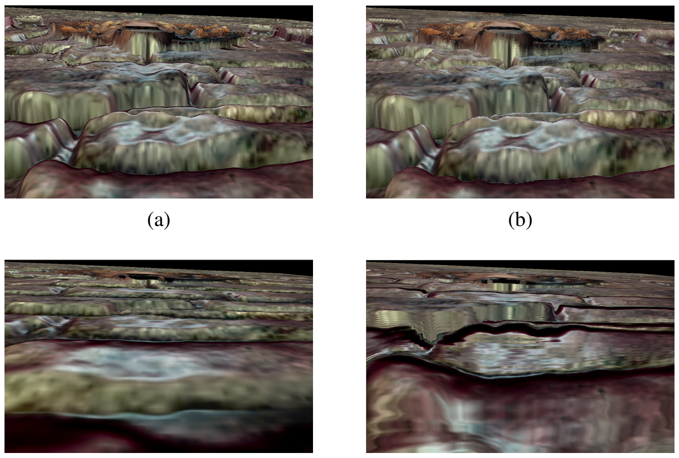
and was positioned in fourth place. Parallax Mapping with Offset Limiting and Parallax Mapping achieved very similar levels of graphics quality for all tests, however, they produced strong distortions for surfaces observed at large angles between the view vector and the surface normal. Normal Mapping produced the images with the worst relative quality, and the lowest perception of relief. In contrast, its frame rate was the largest among all tests.

A summary of the supported effects of motion parallax, self-occlusion, self-shadowing and silhouettes is presented in Table IV.

We also classified the per-pixel displacement mapping techniques according to the frameworks they make use. In particular, the Phong shading model, the displacement of textures [4], the linear search, the binary search, the numerical method for finding root (False Position) [21] and the void space mapping [12]. Figure 9 shows which each of these schemes are used by displacement mapping techniques.

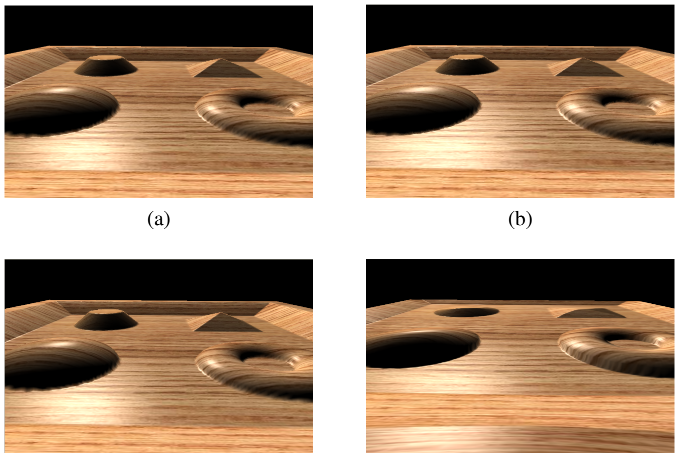
## IV. Final Considerations

This study analyzed and compared per-pixel displacement mapping techniques broadly employed in real-time applications such as games. All compared techniques were implemented in OpenGL Shader Language.



(a) (b) (c) (d)

Fig. 7. Test 7: large angles between the view vector and the surface normal: (a) Cone Step Mapping, (b) Parallax Occlusion Mapping, (c) Relief Mapping and (d) Iterative Parallax Mapping.



(a) (b) (c) (d)

Fig. 8. Test 8: large angles between the view vector and the surface normal: (a) Cone Step Mapping, (b) Parallax Occlusion Mapping, (c) Relief Mapping and (d) Iterative Parallax Mapping.

As future work, we intend not only to analyze further displacement mapping methods, e.g., [22] that explores tessellation shaders on current GPU [23], but also to propose novel schemes for real-time displacement mapping.

## Acknowledgment

The third author thanks to Universidade Federal do ABC, FAPESP and CNPq for supporting this research.

## References

[1] R. Fosner, *Real-Time Shader Programming. San Francisco: Publishers. 2003.* Morgan Kaufmann, 2003.
[2] D. S.-C. Crespo Dalmau, *Core Techniques and Algorithms in Game Programming*, Indianopolis, Ed. New Riders Games, 2004.
[3] T. U. Lszl Szirmay-Kalos, "Displacement mapping on the gpu state of the art," in *Computer Graphics Forum*, 2008.

TABLE IV
SUPPORTED EFFECTS FOR DISPLACEMENT MAPPING TECHNIQUES.

| | **Motion parallax** | **Self-occlusion** | **Self-shadow** | **Silhouettes** |
|---|---|---|---|---|
| Displacement Mapping (DM) | Yes | Yes | No | Yes |
| Bump Mapping | No | No | No | No |
| Normal Mapping | No | No | No | No |
| Parallax Mapping (PM) | Yes | No | No | No |
| PM with Offset Limiting | Yes | No | No | No |
| PM with Slope Information | Yes | No | No | No |
| Iterative PM | Yes | No | No | No |
| Linear Search | Yes | Yes | Yes | No |
| Binary Search | Yes | Yes | No | No |
| Relief Mapping (RM) | Yes | Yes | Yes | No |
| Parallax Occlusion Mapping | Yes | Yes | Yes | No |
| Cone Step Mapping | Yes | Yes | Yes | No |
| DM with Distance Functions | Yes | Yes | Yes | No |
| Relaxed Cone Stepping for RM | Yes | Yes | Yes | No |
| RM Of Non-Height-Field Surface Details | Yes | Yes | Yes | Yes |

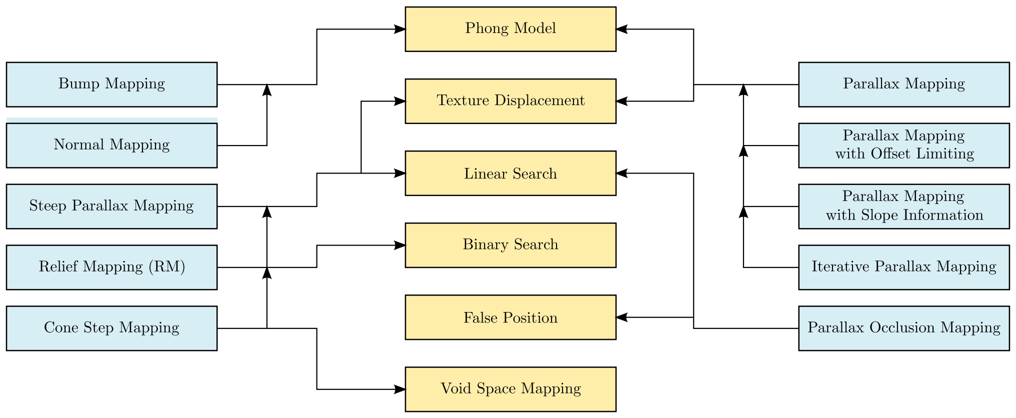


Fig. 9. Frameworks used on per-pixel displacement mapping techniques. *Bump Mapping* [15], Normal Mapping [4], *Steep Parallax Mapping* [17], Parallax Mapping [7], *Parallax Mapping with Offset Limmiting* [8], Iterative Parallax Mapping [9], *Interval Mapping* [18], *Parallax Occlusion Mapping* [12], Relief Mapping [11], Cone Step Mapping [13], *Parallax Mapping with Slope Information* [9], *Relaxed Cone Stepping* [19] and *Distance Functions* [20].

[4] M. P. B. Donald D. Hearn, *Computer Graphics with OpenGL*, 3rd ed. Prentice Hall, 2004.

[5] G. N. de Lima, "Mapeamento de geometrias a partir de texturas em tempo real," Monografia de Trabalho de Conclusão de Curso; Universidade Federal do ABC;2011. [Online]. Available: http://professor.ufabc.edu.br/~joao.gois/gabriel_monograph_pt.pdf

[6] R. J. Rost, *OpenGL Shading Language*. Addison-Wesley Professional, 2006.

[7] T. Kaneko, T. Takahei, M. Inami, N. Kawakami, Y. Yanagida, T. Maeda, and S. Tachi, "Detailed shape representation with parallax mapping," in *Proceedings of ICAT*, 2001, pp. 205–208.

[8] T. Welsh, "Parallax mapping with offset limiting: A per pixel approximation of uneven surfaces," Infiscape, Tech. Rep., 2004.

[9] M. Premecz, "Iterative parallax mapping with slope information," in *Central European Seminar on Computer Graphics*, 2006. [Online]. Available: http://www.cescg.org/CESCG-2006/papers/TUBudapest-Premecz-Matyas.pdf

[10] F. P. Manuel M. Oliveira, "An efficient representation for surface details," UFRGS, Tech. Rep. RP-35, 2005.

[11] F. Policarpo, M. M. Oliveira, and J. a. L. D. Comba, "Real-time relief mapping on arbitrary polygonal surfaces," in *Symposium on Interactive 3D Graphics and Games*, 2005, pp. 155–162.

[12] N. Tatarchuk, "Dynamic parallax occlusion mapping with approximate soft shadows," in *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, 2006, pp. 63–69.

[13] J. Dummer, "Cone step mapping: An iterative ray-height field intersection algorithm," , Tech. Rep., 2006. [Online]. Available: http://www.lonesock.net/files/ConeStepMapping.pdf

[14] R. L. Cook, "Shade trees," in *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '84, 1984, pp. 223–231.

[15] J. F. Blinn, "Simulation of wrinkled surfaces," in *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '78, 1978, pp. 286–292.

[16] M. Peercy, J. Airey, and B. Cabral, "Efficient bump mapping hardware," *Computer Graphics*, vol. 31, pp. 303,306, 1997.

[17] M. McGuire and M. McGuire, "Steep parallax mapping," *I3D Poster*, 2005.

[18] M. A. S. Eric A. Risser and S. Pattanaik, "Interval mapping," School of E. and C. Science, University of Central Florida, Tech. Rep., 2005. [Online]. Available: http://graphics.cs.ucf.edu/IntervalMapping/

[19] H. Nguyen, *GPU Gems 3*. Addison-Wesley Professional, 2007.

[20] M. Pharr and R. Fernando, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005.

[21] R. L. Burden and J. D. Faires, *Numerical Analysis*, 4th ed. Boston, MA, USA: PWS Publishing Co., 1989.

[22] H. Jang and J. Han, "Feature-preserving displacement mapping with graphics processing unit (gpu) tessellation," *Computer Graphics Forum*, 2012. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2012.03068.x

[23] D. Wolff, *OpenGL 4.0 Shading Language Cookbook*. Packt Publishing, 2011.