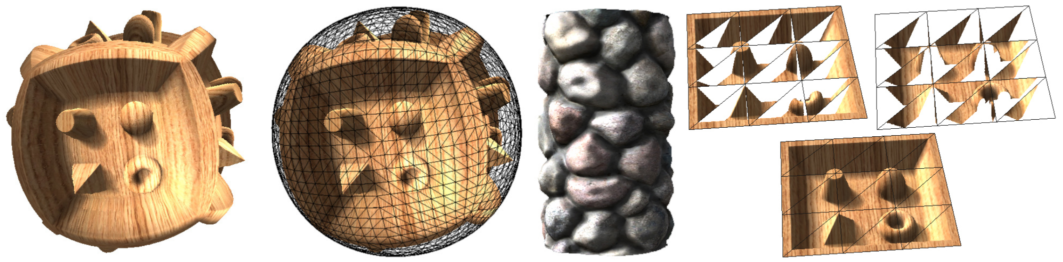
# Prism Parallax Occlusion Mapping with Accurate Silhouette Generation

Carsten Dachsbacher, Natalya Tatarchuk

# Prism Parallax Occlusion Mapping with Accurate Silhouette Generation

Carsten Dachsbacher / REVES/INRIA Sophia-Antipolis [*]    Natalya Tatarchuk / 3D Application Research Group/AMD [†]



Per-pixel displacement mapping algorithms such as [Policarpo et al. 2005; Tatarchuk 2006] became very popular recently as they can take advantage of the parallel nature of programmable GPU pipelines and render detailed surfaces at highly interactive rates. These approaches exhibit pleasing visual quality and render motion parallax effects, however, most of them suffer from lack of correct silhouettes. We perform ray-surface intersection in a volume given by prisms extruded from the input mesh triangles in the direction of the normal. The displaced surface is embedded in the volume of these prisms, bounded by a top and a bottom triangle and three bilinear patches (*slabs*). [Hirche et al. 2004] propose to triangulate the slabs and split the prisms into three tetrahedra. A consistent triangulation of adjacent prisms ensures that no gaps between tetrahedra exist and no tetrahedra overlap. Ray marching through tetrahedra is then straightforward as texture gradients (for marching along the ray) can be computed per tetrahedron.

## 1 Prism Parallax Occlusion Mapping

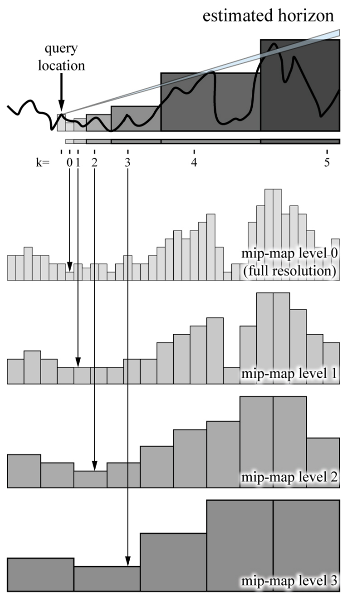
In contrast to previous work, we propose to directly operate on the prisms: The key observation on which we base our method is that we achieve visually pleasing results if we compute correct texture coordinates on the prism surface and rely on a constant texture gradient thereof for each intersecting view ray. To this end, we compute intersections of view rays and prisms and the corresponding texture coordinates in pixel shaders directly. Slabs are rasterized as two triangles; the split diagonal has to be chosen such that the resulting triangular surface is pointing outwards and true intersections are computed per-pixel [Ramsey et al. 2004]. To avoid superfluous intersection tests, we use early rejection tests during geometry processing.

## 2 Shadows and Ambient Occlusion

We use a fast approximation to the horizon angle on height fields to compute ambient occlusion and shadows for local surface features at render time: Close elevations have greater impact on the horizon angle, whereas small scale detail at greater distance can be neglected. For the height map, we generate mip-map levels, but instead of averaging pixels when going to the next level, we store their maximum value. For our estimation we then use $k$ height samples along the view direction, with a distance of $2^k$, $k \geq 0$ texels to the point in question to estimate the horizon slope. To omit the small scale detail, we use the mip-map level $k$ respectively.

We also compute per-pixel ambient occlusion by estimating the the horizon approximation for several directions (4 to 8 in our tests) splitting the hemisphere above the query point into equal sized sectors; per-pixel rotated directions conceal regular structures.

[*]e-mail: Carsten.Dachsbacher@sophia.inria.fr
[†]e-mail:natalya.tatarchuk@amd.com



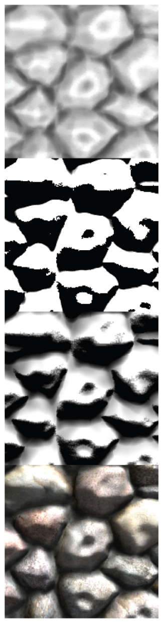


Figure 1: Left: to approximate the horizon we sample distant elevation features at coarser resolution. Right (top to bottom): ambient occlusion (4 directions), hard and soft shadows and final rendering with our horizon approximation.

## 3 Results and Discussion

We implemented our ray-casting method for DirectX®9 class hardware by preprocessing the input meshes to extrude the triangles, and then rendering them on the GPU achieving frame rates of 79 fps for a 500 triangle sphere model (top figure) and 59 fps for a 7K triangle cylinder model on a ATI®Radeon®X1950 graphics card.

Our shadow by horizon approximation technique runs at comparable speed as the POM shadowing technique. Computing per-pixel ambient occlusion causes a 35% drop in performance for 4 static directions, 60% for 8 randomized directions.

## References

HIRCHE, J., EHLERT, A., GUTHE, S., AND DOGGETT, M. 2004. Hardware accelerated per-pixel displacement mapping. In *GI '04: Proceedings of the 2004 conference on Graphics interface*.

POLICARPO, F., OLIVEIRA, M. M., AND COMBA, J. L. D. 2005. Real-time relief mapping on arbitrary polygonal surfaces. In *Symposium on Interactive 3D Graphics and Games 2005*, ACM Press, 155–162.

RAMSEY, S. D., POTTER, K., AND HANSEN, C. 2004. Ray bilinear patch intersections. *Journal of Graphics Tools 9*, 3, 41–47.

TATARCHUK, N. 2006. Dynamic parallax occlusion mapping with approximate soft shadows. In *Symposium on Interactive 3D Graphics and Games 2006*, ACM Press, 63–69.