w_s	size	surfels	fps
2	0	250k	20
2	0.34M	250k	17
4	1.1M	119k	37
8	4.1M	70k	59
16	16.2M	49k	85

Figure 10: Framerate results for using normal mapping vs. no normal mapping (first row) when rendering the Igea model in 1024x1024. The surfel size is limited to $r_{inner} = w_s$ (e.g. $r_{sil} = 2$). The final images for the first row is shown in 2(a) and for the last row in 2(c).

can see we benefit from less surfels, which yields to a significant increase in framerate especially when using larger normal maps.

This table also shows the normal mapping overhead caused by additional vertex/fragment shader operations used for $\mathbf{S}'^{-1}$ and texturing (first and second row). On the one hand calculating $\mathbf{S}'^{-1}$ needs many operations in the vertex shader. On the other hand the number of vertices is very decimated, which more than compensates these overhead. On the other side in the fragment stage not much more fragments have to be shaded when using fewer large splats than many small. In addition, texturing surfels only needs a few additional operations (see 3.3) in the fragment shader. This is important, because fragment shading is still a bottleneck especially of point-based rendering. We can conclude that normal mapping only causes an appreciable overhead by the required texture memory for the normal map.

7 Conclusion and Future Work

In this paper we proposed an approach to decrease the number of surfels in the interior of the object surface without visual disadvantages by using normal mapping and silhouette refinement. This results in a significant increases in framerate. We also shown how to create normal maps for several types of point hierarchies using a ray-casting approach. This is accelerated by using the recursive space subdivision provided by the point hierarchy.

Although our framework supports nearly all kind of point-based surface descriptions there are opinions for future works. When needing more than one normal map (e.g. in the case of many surfels in the hierarchy) an intelligent grouping of surfels to be rendered is necessary to avoid many graphic library procedure calls to select the proper map. Another working topic is to save texture memory. One way to achieve this is to estimate surfels with nearly the same normal map or with a normal map that can be tiled. Such surfels can mostly be found on a surface part with no or low curvature.

REFERENCES

[BE05] G. Barequet and G. Elber. Optimal bounding cones of vectors in three dimensions. *Inf. Process. Lett.*, 93(2):83–89, 2005.

[BK03] M. Botsch and L. Kobbelt. High-quality point-based rendering on modern gpus. In *PG'03 conf.proc.*, page 335. IEEE Computer Society, 2003.

[CAZ01] J. D. Cohen, D. G. Aliaga, and W. Zhang. Hybrid simplification: combining multi-resolution polygon and point rendering. In *Vis'01 conf.proc.*, pages 37–44. IEEE Computer Society, 2001.

[CN01] B. Chen and M. X. Nguyen. Pop: a hybrid point and polygon rendering system for large data. In *Vis'01 conf.proc.*, pages 45–52. IEEE Computer Society, 2001.

[DVS03] C. Dachsbacher, C. Vogelgsang, and M. Stamminger. Sequential point trees. *ACM Trans. Graph.*, 22(3):657–662, 2003.

[Her92] G.T. Herman. Discrete multidimensional jordan surfaces. *CVGIP: Graph. Models Image Process.*, 54(6):507–515, 1992.

[LH01] D. Luebke and B. Hallen. Perceptually driven interactive rendering. Technical Report #CS-2001-01, University of Virginia, 2001.

[LW85] M. Levoy and T. Whitted. The use of points as a display primitive. Technical Report 85-022, Computer Science Department, University of North Carolina at Chapel Hill, 1985.

[PGK02] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02 conf.proc.*, pages 163–170, Washington, DC, USA, 2002.

[PZvBG00] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: surface elements as rendering primitives. In *SIGGRAPH'00 conf.proc.*, pages 335–342, New York, NY, USA, 2000.

[RL00] S. Rusinkiewicz and M. Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH'00 conf.proc.*, pages 343–352. ACM Press/Addison-Wesley Publishing Co., 2000.

[RPZ02] L. Ren, H. Pfister, and M. Zwicker. Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering, 2002.

[ZPvBG01] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *SIGGRAPH'01 conf.proc.*, pages 371–378. ACM Press, 2001.

[ZRB$^+$04] M. Zwicker, J. Räsänen, M. Botsch, C. Dachsbacher, and M. Pauly. Perspective accurate splatting. In *GI'04 conf.proc.*, pages 247–254, University of Waterloo, 2004.