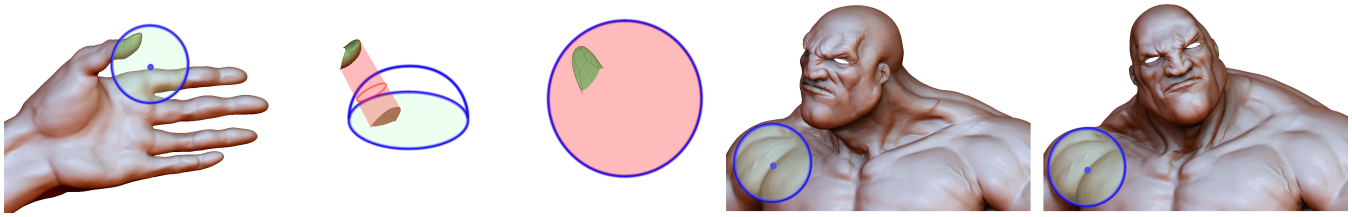
Approximate Ambient Occlusion for Dynamic Scenes using the GPU

Shailen Agrawal*
UBC

Subodh Kumar†
IIT Delhi



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

Figure 1: *(a) Nearest Neighbor query for a single point on hand (b) All neighboring triangles are returned, a part of which is the tip of the thumb. All returned neighbors cause occlusion on the query point. Returned triangles are projected onto the view plane at the occlusion vertex (c) Use a ratio of green and red areas to determine occlusion caused at the query point (d), (e) Some of the occlusion values can be reused between two frames in a sequence if their neighborhood remains the same. For example the local neighborhood of the shown query points in blue doesn't change due to the head turn, so we must avoid reprojecting all the neighbors and recomputing occlusion for such.*

1 Introduction

Ambient occlusion has been tackled in many different ways to inculcate realism into renderings. Ambient occlusion is a crude approximation to global illumination. But performing a full global illumination in real-time has turned out to be computationally expensive. Combined with local rendering models, ambient occlusion can produce renderings which have increased realism.

In this work an improvement over existing occlusion computation techniques is proposed for use in rendering shadows for dynamic scenes. Using modern GPU hardware, ambient occlusion in dynamic scenes can be rendered in real-time. Clever updation and pre-processing techniques are required to accomplish this feat. The main contributions of this technique are towards the development of an approximate occlusion computation via projections onto the view plane at the query vertex and a scheme which utilizes coherence in both spatial and temporal domain to reuse already computed occlusion values.

2 Our Approach

Occlusion is computed at each query point in the scene by determining the local neighborhood of the query point. The local neighborhood search is accelerated using a kd-tree data structure which is built on the GPU in real time using the technique mentioned in [Zhou et al. 2008]. Since the scene is dynamic and we want to maintain the kd-tree structure at each frame, it is essential that we can construct the kd-tree quickly.

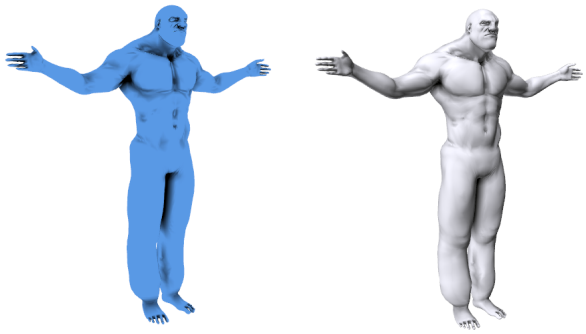
Once all the triangles in the local neighborhood are determined for a query point, we project them onto the query point and determine the occlusion. Using an approach like this, an occlusion computation will have to be performed for all query points for every frame. We develop a scheme which utilizes spatial and temporal coherence for reusing already computed occlusion values. This results in reduction of large number of new occlusion computations for query points which can take advantage of coherence in space or time. The query points can be per-vertex in which case the occlusion values are passed onto a vertex shader for modulating local lighting.

*e-mail: shailen@cs.ubc.ca
†e-mail: subodh@cse.iitd.ac.in

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007

Firstly we can use coherence in space. If the local neighborhood for two query points close-by is similar then the same occlusion value can be used for them instead of recomputing it for every such query point. A neighborhood matching scheme is being developed where fast neighborhood matches can be performed on the GPU. This utilizes coherence in the recipient side. Only the local neighborhood is used, so the model could change at a distance without changing the occlusion at the query point. While projecting details we use a simplified version of the scene which utilizes coherence in occluder side.

Secondly we develop a scheme for utilizing coherence in time. As shown in Figure 1 (d) and (e), there are points for which the local neighborhood remains same across two frames in time. Hence for points like these we can reuse the occlusion values instead of recomputing it for each frame.



(a) Color coded image depicting ambient occlusion.
Blue : no occlusion.
Black : full occlusion

(b) Direct lighting modulated with ambient occlusion

Figure 2: *Current state of rendering from the system*

References

ZHOU, K., HOU, Q., WANG, R., AND GUO, B. 2008. Real-time kd-tree construction on graphics hardware. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, ACM, 1–11.