Simplification of Moving 3D Scene Data on GPU

Rajesh Chenchu, Nick Michiels, Sammy Rogmans and Philippe Bekaert
Hasselt University - iMinds, Expertise Centre for Digital Media, Wetenschapspark 2, 3590, Diepenbeek, Belgium

Keywords: Billboard, Scene Simplification, GPU, Real-time Rendering.

Abstract: Real-time large scale continuous image and geometry based data visualization, with an uninterrupted content delivery, quality and rendering, on home and mobile devices is difficult or even mostly impossible because of the low processing capabilities of these hardware devices. However, a gracefully simplified version of the same data can enable us to view the content without significant quality degradation. To do this in a graceful manner, we extended a well-known concept - called 'billboard cloud' - for animated scene data and implemented this technique using the capabilities of the GPU to generate the simplified versions of large scale data sets.

1 INTRODUCTION

Scene data is a combination of both image and geometry. Visualizing the same scene data in real-time on mobile devices is cumbersome because most detailed scene data is complex and occupies huge storage and high bandwidth on the network. Simplification methods create a simpler version of the scene data targeted for mobile platforms. The main goal of scene simplification is to transform an input representation into an output representation that can be stored, distributed and rendered more easily and precisely on (light-weight) home and mobile platforms.

As each frame of the scene data is a combination of both image and geometry, creating a simpler version with an uninterrupted content delivery, quality and rendering in real-time is a great challenge for the coming years. Mesh decimation has dramatically progressed, and techniques such as edge collapse permit efficient and accurate simplification, e.g., [(Garland and Heckbert, 1997), (Puppo and Scopigno, 1997), (Luebke, 2001)]. However, these methods work best on finely tessellated smooth manifolds, and often require mesh connectivity. Moreover the quality of the simplified model often becomes unacceptable and doesn't work for large sets of data combining different disconnected objects, subjects and scenes. On the other hand, image-based acceleration is very efficient for distant scenery, as it can naturally fuse multiple objects, but offers only limited parallax effects.

A billboard cloud (Décoret et al., 2003) method is used for simplification in this paper. This method simplifies 3D models into a set of planes with texture and transparency maps and results in better output compared to other techniques. This method was proposed for static objects, we extended current method to support moving scenes by using the capabilities of the GPU. Our method generates a simplified version of each frame in the scene data and render on the target device.

2 PROPOSED METHOD

There are different methods to implement billboard cloud techniques. Hough transform, Stochastic method (Lacewell et al., 2006) and K-means clustering algorithm (Huang, 2004) are well known methods for identifying planes for billboard clouds. The original method introduced by Décoret (Décoret et al., 2003) uses Hough transform and greedy method. The greedy method uses recursive sub-division approach for finding the best-plane. The existing Hough transform method takes a lot of time for processing even one million triangular meshes and a possibility to hit an infinite loop during recursion. The proposed method eliminates the recursive sub-division approach and uses Hough transform, mipmap techniques and the capabilities of a compute shader unit to generate billboard clouds for scene data. These results were realized in the context of the SCENE EU FP7 project (SCENE, 2014).

2.1 Hough Transform

The Hough transform for planes is a voting procedure where each feature (triangle) in a mesh votes for all possible planes passing through that feature. All votes are stored in the so called Hough space, which is three dimensional for the Hough transform for planes. The

Chenchu, R., Michiels, N., Rogmans, S. and Bekaert, P.
Simplification of Moving 3D Scene Data on GPU.
DOI: 10.5220/0005962000950098
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 5: SIGMAP*, pages 95-98
ISBN: 978-989-758-196-0

size of the Hough space is determined by the size of the mesh and the required accuracy for the parameterization of the planes. Scene frame triangles are used as input and the output consists of parameterized planes. Planes are commonly represented by a signed distance d to the origin of the coordinate system and the slopes m_x and m_y in the direction of the x- and y-axis, respectively:

$$z = m_x x + m_y y + d \qquad (1)$$

To avoid problems due to infinite slopes when trying to represent vertical planes, another usual definition of a plane is the Hesse normal form. A plane is thereby given by a point p on the plane, the normal vector n that is perpendicular to the plane and the distance d to the origin

$$p.n = p_x n_x + p_y n_y + p_z n_z = d \qquad (2)$$

Considering the angles between the normal vector and the coordinate system, the coordinates of n are factorized to

$$p_x.cos\theta.sin\phi + p_y.cos\phi + p_z.sin\phi.sin\theta = d \qquad (3)$$

Given a feature p in Cartesian coordinates, Our method will find all possible planes that the feature lies on, i.e., find all planes that satisfy $Eq.(3)$. Marking each point in a triangle in the Hough space leads to a 3D sinusoid curve and the intersection of three curves in Hough space corresponds to the polar coordinates defining the plane spanned by the three points. Given a set P of features (point or triangle) in Cartesian coordinates, one transforms all features $p_i \in P$ into Hough space. The more curves intersect $h_j \in (\theta, \phi, d)$, the more features (point or triangle) lie on the plane represented by h_j and the higher is the probability that h_j is actually extracted from P.

3 IMPLEMENTATION

A view-independent billboard cloud generation method is implemented in three modules i.e., plane generation, texture generation and renderer. Texture generation method takes the input data from the plane generation module and the renderer method takes the input data from texture generation. These modules are executed independently.

3.1 Plane Generation

Plane identification for billboard cloud using Hough transform is implemented using compute shaders. The input for the plane generation module is a normalized scene data frame. Once the plane information is generated for one frame in the scene data then it takes the next frame and continues for the entire scene video. Each frame in the scene data needs to be normalized and centered w.r.t (0, 0, 0) before processing to avoid misalignments. Usually scene mesh data resolution is high during 3D-reconstruction, for example more than one million triangles for each frame, so our method generates different sets of plane information for mobile and TV, based on the target device resolution. The pseudo code for the selection of planes for a single frame is shown in Figure 1.

```
Plane Generation (SCENE video frame, threshold ε)
  set of faces F = SCENE video frame
  planes P = ∅
  Dispatch_CS1 : compute density in plane space with threshold ε
  while F ≠ ∅
      Dispatch_CS2 : pick bin B with highest density
      $p_i$ = compute plane normal parameters (B)
      compute validfaces ($p_i$)
      Dispatch_CS1 : updateDensity(validfaces($p_i$))
      F = F \ validfaces($p_i$)
      $P = P \cup p_i$
```

Figure 1: Pseudocode for Plane Generation.

$Dispatch\_CS1$ and $Dispatch\_CS2$ code blocks are executed on the GPU computer shader unit. $Dispatch\_CS1$ block is implemented to update histogram table when adding or removing the faces to/from the accumulator and the $Dispatch\_CS2$ block is implemented with a mipmap technique to find the highest density bin in the 3D plane space. $Dispatch\_CS1$ shader can be invoked for all faces at once, if the GPU is capable to process within the specified timeout. Otherwise mesh data needs to be split into batches and submit to the GPU by invoking the $Dispatch\_CS1$ shader every time. There are options to change/disable the GPU timeout to process all faces at once but this is not always recommended.

3.2 Validity

The plane-space grid uses a simple validity. The Euclidean distance between a point and its simplification should be less than a small threshold ε. This means that a point is allowed to move only in a sphere of radius $epsilon$, i.e., a plane p yields a valid representation of a point v if there exists a point p on P such that $||vp|| < \varepsilon$. The validity domain of v as the set of planes that can represent it, which is denoted by $valid_\varepsilon(v)$. It is the set of planes that intersect the sphere of radius ε centred on v. A plane is valid for a polygon if it is valid for all its vertices. The validity domain $valid_\varepsilon(f)$ of a face f is the intersection of the validity domain of its vertices. Geometrically, this corresponds to the set of planes intersecting a set of spheres. We will interchangeably say that a face is valid for a plane, and that the plane is valid for the face.

3.3 Cost Function

The cost function for updating the density value is 1 at the time of adding a face to the histogram and -1 for removing a face from the histogram. We can also consider the traingle area as a cost function weight for adding and removing the faces from the histogram.

3.4 Crack Reduction

A general solution to minimize cracks between adjacent billboards is to render the faces onto the textures of all planes for which they are valid. This operation is easily performed by rendering the entire scene in the texture, using extra clipping planes to restrict imaging to the valid region around the plane.

3.5 Texture Generation

Billboard cloud textures are generated by finding the minimum bounding rectangle of the projection of the faces in a texture plane and shooting a texture by rendering all valid faces using an orthographic projection. Texture resolution is automatically selected according to the object-space size of the bounding rectangle. If the normal maps are computed together with the textures, a billboard cloud can be relit in real-time. Texture and normal map packing in a single or multiple texture atlas improves the performance during run-time. Our method uses a simple greedy packing algorithm (Coffman et al., 1997) to store all the textures of an entire billboard-cloud into a single/multiple texture atlas which optimizes the memory usage and minimizes texture switching.

3.6 Renderer

For rendering textured billboards, our method will find the correct mapping between the texture and plane coordinates. For proper alpha-compositing and mip-mapping, our method uses black background and pre-composited alpha (Porter and Duff, 1984). The rendering of a billboard cloud is penalized mainly by the fill rate: when 100 interleaved billboards are rendered, lots of pixels are drawn many times (the worst case being parallel billboards viewed from the front). Therefore, a billboard cloud is useful when its projected screen size is not too large. When an object is far away, using a billboard cloud simplification gives higher quality results than a geometrically simplified approach.

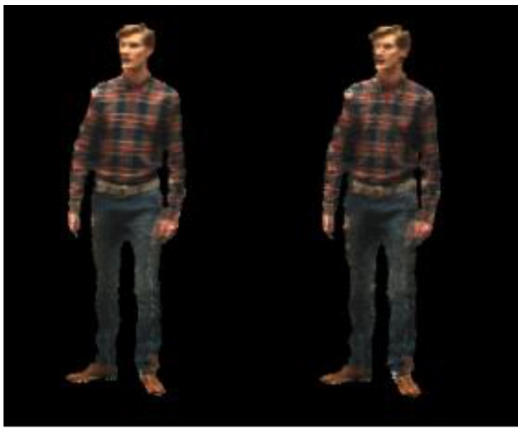
Figure 2: Scene frame data: original(left) vs simplified(right).

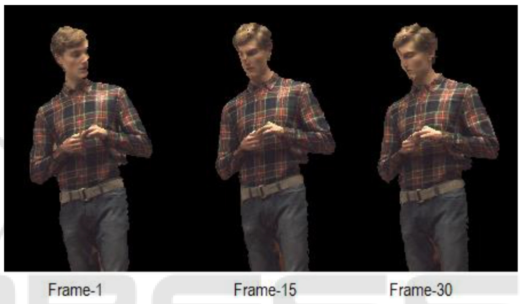

Figure 3: Frames in the scene video file.

4 RESULTS

The result of the proposed technique is shown in Figure 2 for a single scene frame. The 3D-reconstructed frame is shown on the left and the simplified version of the billboard cloud is shown on the right. Our method focuses on the visual quality of the results, without optimizing for resource usage. The technique always converges and yields a billboard cloud where all original faces have been collapsed on at least one billboard, while enforcing the error bound. A few frames of the scene video file are shown in Figure 3. A billboard cloud is generated for each frame and stored in the scene format. The renderer module on the target device reads the billboard cloud data from the scene format and renders each frame continuously. Normal maps are used to relight the scene if required. Usually the number of billboards are less for simpler objects but complex irregular objects generate more billboards to retain all features in the scene. Figure 4 shows the correlation between the number of triangles in the scene frame and the generated billboards. More billboards were identified when the scene frame contains more triangles. Normally

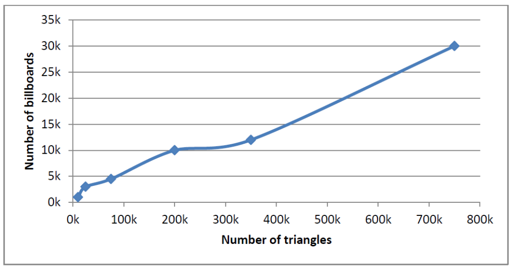

Figure 4: Number of triangles in the scene frame data vs generated billboards.

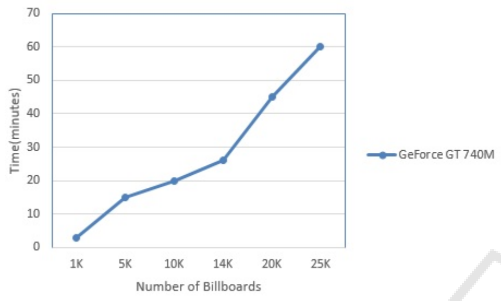

Figure 5: Number of billboards vs processing time.

a complex mesh will generate more billboards than a simpler mesh to preserve the same visual quality. Figure 5 shows the correlation between the generated billboards and GPU processing time.

5 CONCLUSIONS

Our method generates a simpler version of the scene data that can be stored, distributed and rendered more easily and precisely on home and mobile platforms. Our simplification approach of scene frame data using billboard clouds gives promosing results. The extreme case of a single billboard would obviously look flat, to retain the same visual quality with original frame, the proposed method generates a sufficient number of billboards for each frame and also shows a significant performance improvement on high-end GPUs. Our method is not recursive and completely works with independent parallel threads on GPU, so there is no chance to get into an infinite loop at any stage, in contrast to existing methods. We also rendered a few scene frames on mobile devices for testing purposes and the presented results look good. A few remaining performance bottlenecks will be addressed in the future work.

ACKNOWLEDGEMENTS

The authors acknowledge financial support from iMinds and the SCENE EU STREP project (grant nr 287639).

REFERENCES

Coffman, Jr., E. G., Garey, M. R., and Johnson, D. S. (1997). Approximation algorithms for np-hard problems. chapter Approximation Algorithms for Bin Packing: A Survey, pages 46–93. PWS Publishing Co., Boston, MA, USA.

Décoret, X., Durand, F., Sillion, F. X., and Dorsey, J. (2003). Billboard clouds for extreme model simplification. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 689–696, New York, NY, USA. ACM.

Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Huang, I. (2004). *Improved Billboard Clouds for Extreme Model Simplification*. Computer Science)–University of Auckland.

Lacewell, J. D., Edwards, D., Shirley, P., and Thompson, W. B. (2006). Stochastic billboard clouds for interactive foliage rendering. *J. Graphics Tools*, 11(1):1–12.

Luebke, D. P. (2001). A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.*, 21(3):24–35.

Porter, T. and Duff, T. (1984). Compositing digital images. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 253–259, New York, NY, USA. ACM.

Puppo, E. and Scopigno, R. (1997). Simplification, lod and multiresolution - principles and applications.

SCENE (2014). Scene. http://3d-scene.eu/. SCENE is co-funded by the European Commission under the Seventh Framework Programme FP7 ICT.