June 03, 2019

Distortion-Guided Mesh Processing

Ondrej Stava

Igor Vytyaz

Distortion-Guided Mesh Processing

Abstract

Mesh processing algorithms have the potential to perform better when being guided by a distortion metric. Some mesh processing algorithms can take advantage of such guidance and produce better results, while maintaining high visual quality. Mesh geometry decimation and texture coordinate generation are candidate algorithms to be guided by the distortion metric. An additional motivation is to provide a convenient way to visualize the most distorted areas on mesh surface in 3D space.

Detailed Description

Distortion-guided algorithms require the mesh to be augmented with local distortion information that quantifies degradation of mesh visual quality introduced by mesh processing. Mesh distortion can be made available at all points on the mesh surface. Surface properties, such as color, can be described in a number of ways such as the following approaches:

- Face attribute
- Vertex attribute
- Texture map

The mesh distortion can be represented using any of these approaches as well. An approach can be chosen based on the properties of the mesh and/or the nature of the processing algorithm that uses distortion information for guidance.

Mesh properties can influence how distortion is represented. For example, distortion can be represented in a way that reflects the expression of other mesh properties. Specifically, consider a mesh with sparse geometry where surface details are mostly captured in texture maps. For such mesh, representing distortion in terms of a vertex attribute may not be sufficiently descriptive. Instead, the distortion can be represented in terms of a texture map with resolution comparable to resolutions of other mesh textures. On the other hand, when a mesh has low-resolution textures and most of the surface details are captured with geometry described by vertex positions, the distortion can be represented in terms of a vertex or face attribute.

The nature of the processing algorithm may affect the choice of the distortion representation as well. For example, when an edge collapse algorithm makes a decision to collapse an edge, it may be convenient for it to access distortion at vertices connected by this edge, or from faces that share the edge. On the other hand, for parameterization algorithms that generate texture coordinates, it may be more convenient to work with distortion in the form of a texture map. Therefore, all three distortion storage methods listed above may be needed.

While the color of a texture pixel, vertex, or face is traditionally stored as three RGB or four RGBA components, the distortion at a single point on mesh surface could be described as a scalar. For distortion visualization purposes, an 8-bit integer is sufficient, e.g., stored in a grayscale texture. When distortion is used to guide a processing algorithm, a more precise representation may be needed, such as a single precision floating-point, which can be justified by the floating-point data type.

It may also be desirable to represent distortion at a mesh surface point as a 3D vector, with magnitude representing the amount of distortion, and direction representing a camera viewing angle that would see most degradation in visual quality at that surface point. Similar to surface normals, the distortion vector could be represented in model space or tangent space.

During the computation of an overall mesh distortion metric, snapshot images of reference and processed meshes are taken from dozens of angles. Figure 1 illustrates an example pair of snapshot images of reference and processed meshes taken from the same angle. In this example, the visual appearance of the processed mesh has been distorted as the color of one mesh face has changed.

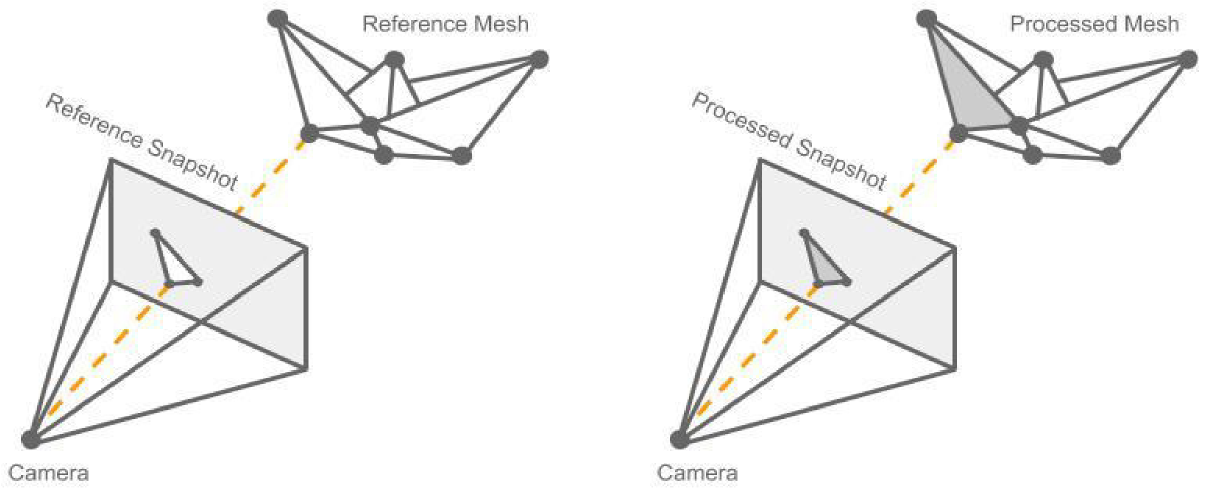

Figure 1. Snapshot images of reference and processed mesh taken from the same angle.

The pair of snapshot images is then compared using an image-based distortion metric, such as MS-SSIM. An example distortion heat map image is computed as shown in Figure 2.

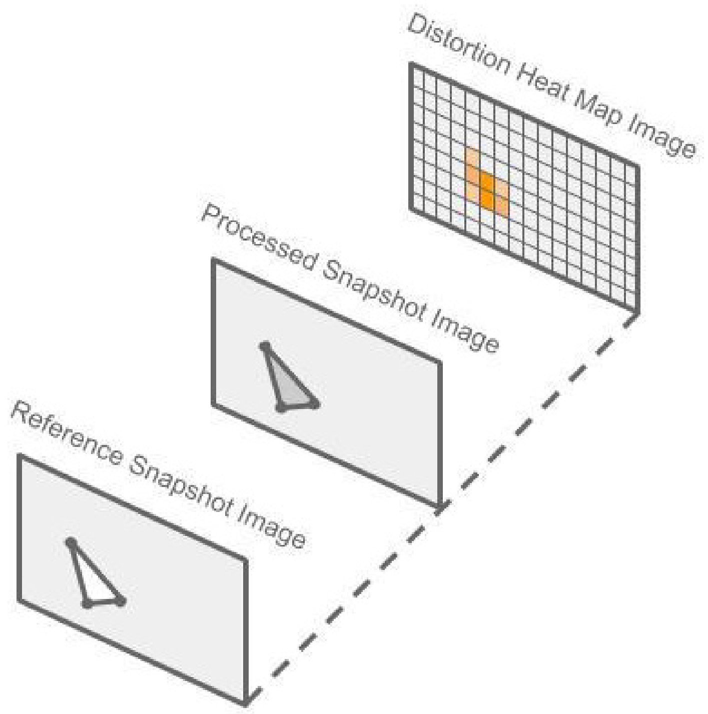

Figure 2. Distortion heat map is computed by comparing a pair of snapshot images.

Pairs of snapshot images are taken from various camera angles, resulting in dozens of heat map images that describe mesh visual distortion when viewed from various camera angles. The next step is to transfer distortion information from a collection of 2D heat map images onto the surface of a 3D mesh, such that the information can be used to guide mesh processing.

For every available viewing angle, a rectangular canvas with heat map image is placed in 3D space between the camera and the mesh, as shown in Figure 3. The size and position of the canvas is selected, such that heat map pixels are aligned with corresponding mesh surface points, form the camera's point of view. Depending on the desired type of distortion storage, i.e., face attribute, vertex attribute, texture map, appropriate ray tracing procedures are used to transfer distortion information from heat map pixels to mesh surface, as described next.

An example process of transferring distortion from heat map to face attribute is illustrated in Figure 3. For each heat map pixel, a ray originating from camera location and passing through the center of the pixel is traced, until it intersects a mesh face, or misses the mesh. In some implementations, there are a plurality of rays traced for each face of the mesh rather than one ray per pixel. The distortion at the heat map pixel contributes to the distortion attribute of the face that has been intersected by the ray. When distortion attribute is represented as a 3D vector, ray direction contributes to distortion vector direction.

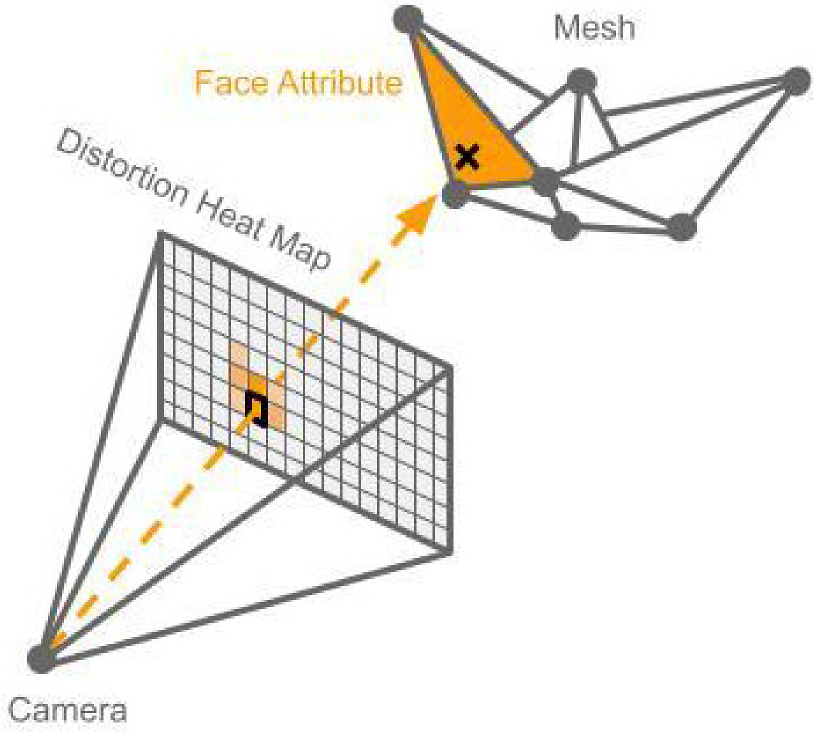

Figure 3. Ray tracing is used to transfer distortion from heat map pixel to mesh face attribute.

The same face may be intersected by rays from a number of pixels of a heat map, as well as by rays from other heat maps at other camera angles. All ray intersections are considered in the computation of the final distortion value for this face attribute.

There may be mesh faces that are never intersected by rays. Such faces are either not visible from any of the available camera angles or are too small and below the resolution of the heat maps. The distortion attribute value is then set to zero or to a value indicating unknown distortion. Tracing multiple rays for each face of the mesh, as described above in some implementations, may alleviate this problem.

An example process of transferring distortion from heat map to vertex attribute is illustrated in Figure 4. The ray tracing procedure is similar to the above-described procedure for the face attribute, in which a ray is traced from camera location, through a heat map pixel, until it intersects a mesh face. The pixel distortion contributes to vertex distortion attribute of the three face vertices. Barycentric coordinates of the location where the ray intersects a face (marked with a cross in Figure 4) is used to split the distortion between face vertices. The split is such that vertices that are closer to the location of the intersection receive larger amounts of distortion. Distortion could also be associated with face corner points, rather than with geometry vertices.

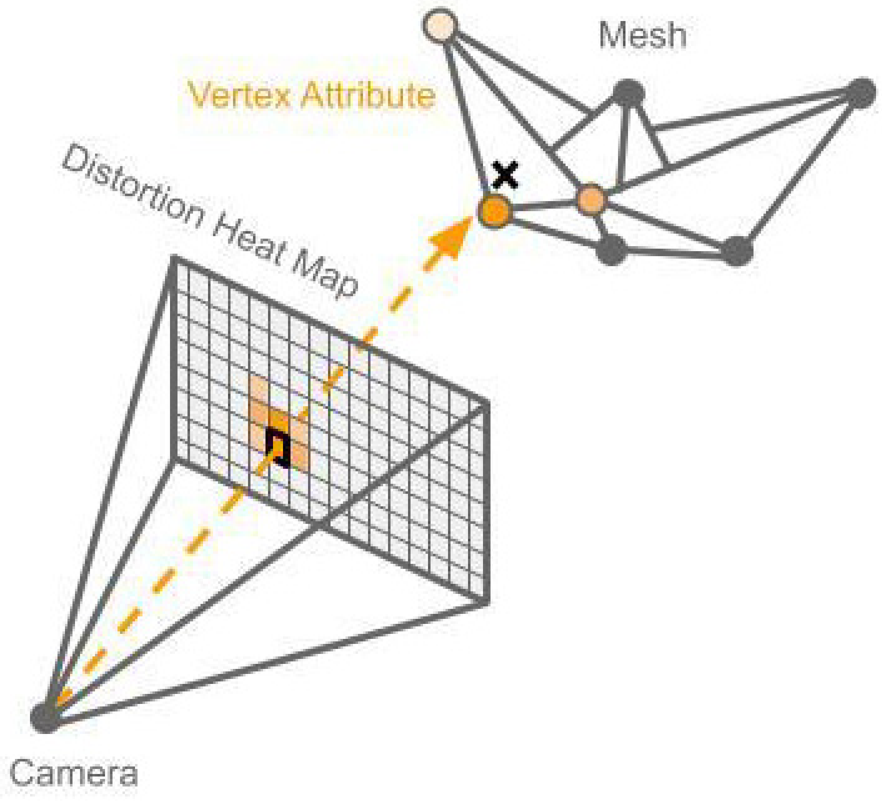

Figure 4. Ray tracing is used to transfer distortion from heat map pixel to mesh vertex attribute.

Ray intersections from all pixels of all heat maps can be considered when computing the final value of vertex distortion attribute. Contributions from all ray intersections can be considered while computing the final distortion value.

The process of transferring distortion from heat map to mesh texture is illustrated in Figure 5. Again, ray tracing is used, however, the ray direction is now opposite, originating at the mesh and tracing towards the camera. Each pixel of the mesh distortion texture is mapped onto the surface of the mesh using mesh texture coordinates. One of such mapped pixels is shown as a grayscale square on the mesh surface in Figure 5. A ray is traced from the center of the pixel in the direction of the camera, until it intersects the canvas, or is blocked by an opaque mesh.

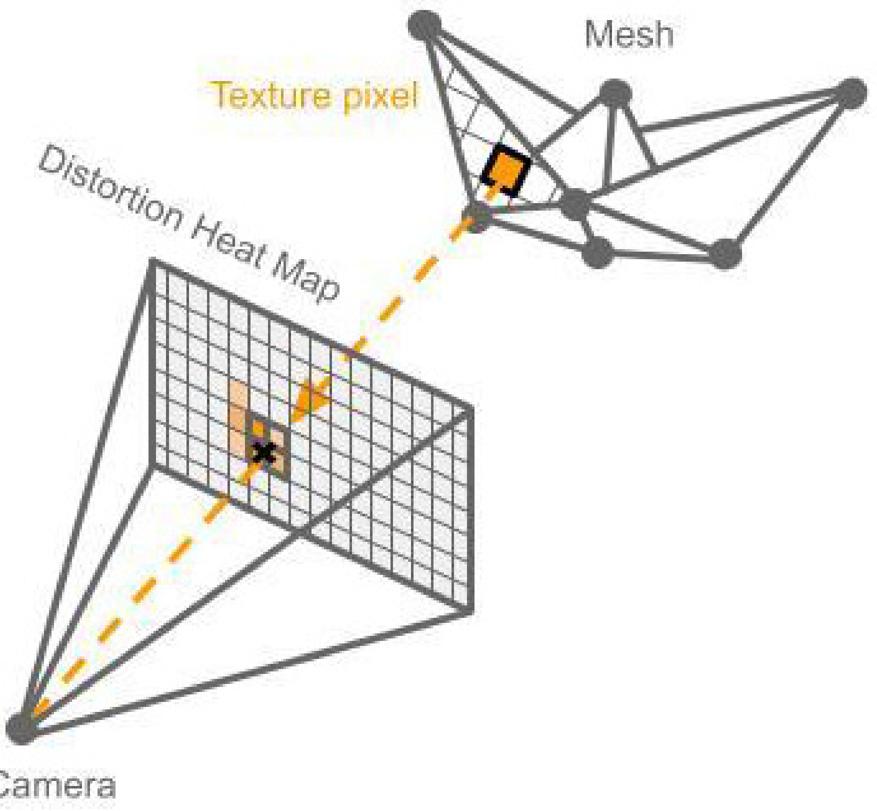

Figure 5. Ray tracing is used to sample heat map and transfer distortion to mesh texture pixel.

Distortion is sampled from the heat map at the point where the ray hits the canvas. In some implementations, bilinear interpolation is used where distortion is computed from four nearby heat map pixels. The sampled distortion contributes to the mesh distortion texture pixel from which the ray was traced. Distortion sampled from other heat maps at other camera angles can also contribute to this mesh texture pixel. This ray tracing process can be repeated for all mesh texture pixels.

Rays from some of the texture pixels may never hit any of the canvases. Such pixels are not visible from any of the available camera angles. Distortion for such pixels may be assumed to be zero, or have a special value indicating that the distortion is unknown.

During the ray tracing, it is common for multiple rays to pass through one mesh face or through one mesh texture pixel. Generally, these rays transfer distortion information from multiple heat map pixels and from multiple heat maps at various camera angles. Distortion values transferred by all rays can be combined into a final distortion value of an individual face, vertex, or texture map pixel.

Some mesh surface regions may be visible from a few camera angles and other regions may be visible from a large number of camera angles. Also, some mesh faces may have different surface areas. These differences in mesh surface properties will potentially lead to an order of magnitude difference in the number of rays that hit for individual faces and texture map pixels. Distortion values can be computed in a way that makes the final distortion values consistent (in some sense) with each other across the mesh surface.

For distortion visualization purposes, distortion values transferred by all rays that intersect a face or a texture pixel can be averaged. With averaging, faces and map pixels of different areas and visibility to camera can appear to have similar distortion if corresponding mesh regions have similar distortion in heat maps.

For guided mesh processing, the maximum distortion of all intersections can be considered. In some implementations, the distortions from all intersections for one viewing angle are accumulated and averaged across all relevant viewing angles. Such max and accumulated metrics can provide a better guidance to processing algorithms that make decisions per individual face, edge, vertex, or pixel, because such metric represents a total cost of making a decision. For efficiency, ray tracing can be skipped for heat map pixels with zero or acceptably low distortion.

Other ways of combining distortion values may be valid. For example, one could justify accumulating distortion without averaging across viewing angles when it is more acceptable to introduce distortion visible from one viewing angle, rather than from many viewing angles, as would be the case for the inside surface of an opaque bottle.