Colored Stochastic Shadow Mapping for Direct Volume Rendering

Johannes Weidner and Lars Linsen

Westfälische Wilhelms-Universität Münster, Germany

Abstract

Creating direct volume renderings with shadows can lead to a better perception of the rendered images. Deep shadow maps is an effective and efficient approach to compute shadows at interactive rates. We propose an alternative approach based on colored stochastic shadow maps (CSSM), which can produce higher-quality colored shadows while using less texture memory. CSSM is based on a stochastic generation of the shadow map, which is queried multiple times for denoising during rendering.

CCS Concepts

•***Computing methodologies*** → ***Rendering;***

1. Introduction

Direct volume rendering (DVR) has become the standard in scientific and medical applications for visualizing volumetric data sets with opaque and translucent objects. Standard approaches implement the discrete volume rendering integral based on the emission-absorption model [Max95], which neglects light scattering. Illumination methods are used to compensate for this, for renderings that are better to perceive and easier to interpret. Shadow computations are one of the central parts for realistic illumination results. Many computer graphics methods exist to compute shadows, which are directly applicable to isosurface renderings for volumetric data visualization. However, when dealing with translucent objects, more effort has to be taken to properly capture shadows in DVR.

When compared to other shadow computation methods, see Section 2, deep shadow mapping is an effective and efficient, generally applicable method for shadow computation in interactive DVR applications [HKSB06, RKH08]. Deep shadow maps when applied to DVR would theoretically capture the accumulated light absorption along rays sent from a light source through each pixel of the shadow map in a preprocessing step, which then would allow for perfect shadow computations during the rendering phase. Storing the accumulated light absorption along rays precisely requires an unpredictable amount of time and space. Due to limited amount of available texture memory when implemented on the GPU, the accumulated light absorption needs to be approximated in practice. This approximation can be improved using a stochastic model. McGuire and Enderton [ME11] introduced the concept of colored stochastic shadow maps (CSSM) for surface renderings, which can be regarded as a stochastic equivalent of deep shadow maps. We present an approach to adopt the CSSM approach for DVR, see Section 3. We apply our approach to synthetic and real data sets and compare it to deep shadow maps in terms of efficiency and quality, see Section 4. We discuss advantages and limitations of our method and demonstrate that the DVR quality can, indeed, benefit from using CSSM without severe efficiency drawbacks.

2. Related Work

Many approaches exist for illuminating direct volume rendered images for interactive applications. Jönsson et al. [JSYR14] categorize them as: (1) Local methods, which only consider local illumination models [Lev88], possibly enhanced with ambient occlusion [HLY10]. They do not support shadow computations. (2) Slice-based methods, which propagate illumination from one slice to the next and, thus, are limited to be used in slice-based techniques [CCF94]. (3) Light space methods, which compute illumination as seen from the light source as in our approach. Light space methods are independent of the underlying DVR approach. (4) Lattice-based techniques, which compute the illumination on the underlying grid as lattice. They require a larger amount of texture memory to store the shadow volume [BR98]. (5) Methods using basis function representations of the illumination, where light source radiance and transparency are computed from the border of the volume using basis functions such as spherical harmonics [Rit07]. Storing the precomputed information leads to an increased memory footprint. Jönsson et al. [JSYR14] also mention ray tracing-based methods, which do not support interactive usage though, and methods only applicable to isosurface rendering, which do not support general DVR applications.

In comparison to the other categories, light space methods are generally applicable methods with low memory footprint and highly interactive rendering rates. The most prominent representative of light space methods is the deep shadow mapping approach. It has been introduced to illuminate complex scenes with multiple semi-transparent structures [LV00], which is not supported by the original shadow mapping approach [Wil78]. Opac-

DOI: 10.2312/eurovisshort.20181072

ity shadow maps support semi-transparent structures by using a stack of shadow maps [KN01]. Deep shadow maps use a more compact representation by storing a piecewise linear approximation of the shadow function. This approach can be adapted for DVR, e.g., by storing the absorption in the form of the accumulated opacity value [HKSB06, RKH08]. Deep shadow mapping typically uses a single point-light source located outside the rendered volume. Updates of the deep shadow map as well as applying it to the DVR is typically achieved at interactive rates. The memory footprint is proportional to the number of samples (or control points) used for the piecewise linear approximation of the shadow function. We introduce the concept of CSSM for DVR as a light space method and compare it in terms of rendering quality, rendering time, and memory footprint to the state of the art, i.e., a standard shadow mapping and a deep shadow mapping approach, see Section 4.

3. CSSM for Direct Volume Rendering

CSSM for semi-transparent surface rendering has been introduced by McGuire and Enderton [ME11]. It is a two-step rendering approach, where (1) the scene is rendered from the light source to generate the shadow map and (2) the scene is rendered from the viewpoint to generate the output image while considering the shadow map for illumination. For the shadow-map generation (1), it emits one photon of a certain wavelength from the light source through each pixel of the shadow map. The photon is traced through the scene and at each intersection with a surface the probability of the photon being absorbed or reflected (i.e., not transmitted) is computed and compared against a random threshold, cf. [ESSL11]. If the probability exceeds the threshold, the photon is not transmitted and the depth value is stored in the shadow map. This procedure is performed for each RGB color channel and the resulting depth values are stored in an RGB color texture, the so-called CSSM. During the rendering step with shadows (2), the distance of a visible surface to the light source at each fragment of the render buffer is compared against the value from the CSSM to decide, whether the surface is shadowed or not. Again, this is performed for each RGB channel. Since the CSSM only stores the result of one random test, an averaging over multiple tests is necessary to produce a denoised colored shadow, which is achieved by averaging the test results of multiple CSSM entries with local offsets of the CSSM fragments.

We adapt this procedure to DVR. For the CSSM generation step (1), the probabilistic events need to be modified to represent a photon traversing a volume with changing opacities. We adopt the computations of the discrete volume rendering integral. Hence, the ray that the photon follows is discretized and the segments of the discretized ray are traversed from the light source. For the i^{th} segment, we have the event V_i that the photon interacts with the volume in that segment, which depends on the opacity of the volume. We obtain the probability of event V_i by $P(V_i) = \alpha_i$. Moreover, if the photon interacts, we have the event $T_{i,\lambda}$ that the photon of wavelength λ is transmitted through the volume of the i^{th} segment, which depends on the color properties of the volume in that segment. We obtain the conditional probability $P(T_{i,\lambda}|V_i) = t_{i,\lambda}$. We can then compute the probability $\rho_{i,\lambda} = P(\overline{T_{i,\lambda}})$ that the photon of wavelength λ is not transmitted through the i^{th} volume segment as one minus the probabilities that the photon is not interacting and that the photon is interacting but transmitted, i.e.,

$$\begin{aligned}\rho_{i,\lambda} &= 1 - (P(\overline{V_i}) + P(V_i) \cdot P(T_{i,\lambda}|V_i)) \\ &= 1 - ((1-\alpha_i) + \alpha_i \cdot t_{i,\lambda}) \\ &= \alpha_i \cdot (1 - t_{i,\lambda})\end{aligned}$$

This probability is tested against a random threshold in each segment, until the probability exceeds the threshold, which indicates that the photon is not transmitted. Then, the depth value of that segment is stored in the shadow map for the given wavelength of the photon. Using the RGB color model, the computations are done for each of the RGB color channels and the CSSM is stored in an RGB color texture. Table 1 provides the pseudocode for the CSSM generation. The for-loop over the segments i can be terminated early when c_R, c_G, and c_B are smaller than 1.

Table 1: *Pseudocode for CSSM generation.*

```
for all fragments of CSSM:
    Initialize depths (c_R, c_G, c_B) := (1, 1, 1).
    Cast and discretize ray from light source through fragment.
    for i = 1 to n:
        Compute depth z_i ∈ [0, 1] of segment i.
        Compute random number r_i ∈ [0, 1].
        for λ = R, G, B:
            Compute ρ_{i,λ}.
            if (ρ_{i,λ} > r_i) then c_λ := min(c_λ, z_i)
    Store (c_R, c_G, c_B) in fragment.
```

During the DVR step (2), the color at each volumetric sample that is composited to generate the final volume rendered image is adjusted by multiplying with a shadow factor, where the shadow factor is obtained by querying the CSSM. The volumetric sample's distance to the light source is compared against the depth value stored in the CSSM. If the sample's distance is larger, then the sample is in the shadow and the shadow factor is 0. Otherwise, it is 1. This test is performed per wavelength, i.e., for each RGB color channel when using the RGB color model. However, since the CSSM is stochastic, we have to average over multiple tests. Multiple tests are obtained by not querying the CSSM only for the sample position but also for some surrounding values. Then, one averages the results of m queries with offsets $\Delta_1, \ldots, \Delta_m$. Consequently, the shadow factor is an averaged value $\frac{j}{m}$ with $j \in \{0, \ldots, m\}$ indicating the amount by which the current sample is shadowed. Table 2 provides the pseudocode for the CSSM usage to compute the shadow factor. This can be embedded into any DVR approach. In our implementation, we used a GPU-implementation of a ray casting approach.

4. Results and Discussion

We implemented our algorithm using the Voreen framework [Vor], which supports a GPU implementation of a ray casting approach. We tested our algorithm using a synthetic data set called

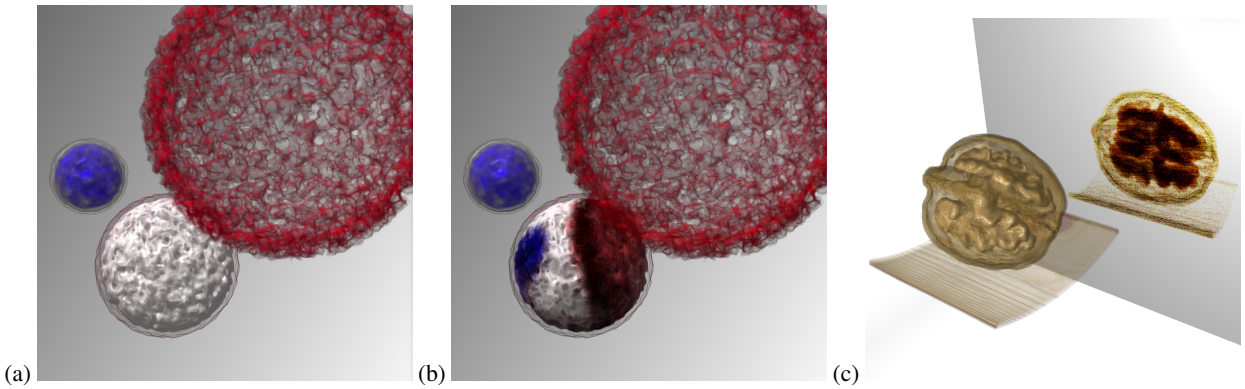

Figure 1: *Ray casting of synthetic data set without (a) and with (b) CSSM. (c) CSSM for combined volume and surface rendering when applied to Walnut data set in front of planar surface.*

Table 2: *Pseudocode for CSSM usage.*

for all samples of the DVR approach:
  Initialize shadow factors $(l_R, l_G, l_B) := (0,0,0)$.
  Compute sample position (s_x, s_y, s_z) in light coordinate system.
  for $j = 1$ **to** m:
    Read entries (c_R, c_G, c_B) from CSSM fragment $(s_x, s_y) + \Delta_j$.
    for $\lambda = R, G, B$:
      if $(c_\lambda > s_z)$ **then** $l_\lambda := l_\lambda + \frac{1}{m}$
  Use shadow factor (l_R, l_G, l_B) for illuminating sample.

Test Spheres at resolution 128^3 [Sph] and a CT scan of a walnut at resolution $400 \times 296 \times 352$ [Wal]. Figure 1 shows results for the synthetic data set without (a) and with (b) colored shadows generated using the stochastic approach. For generating the denoised shadows, we used $m = 13$ offsets $\Delta_j \in \{(0,0),(\pm3,\pm3),(\pm4,0),(0,\pm4),(\pm7,0),(0,\pm7)\}$. The resulting image exhibits the expected and desired features.

Similarly, we created a result for the Walnut data set, see Figure 1(c). Here, we inserted a plane, on which the colored shadows can be observed. We would like to point out that our approach is compatible with the CSSM approach for surface rendering [ME11]. Hence, we can use it to calculate shadows for scenes with volume and surface renderings. To do so, one would first generate the CSSM for surface renderings and subsequently apply our CSSM generation approach for DVR without clearing the CSSM buffers in between.

The main parameters used for the CSSM approach are the resolution of the shadow map and the number of offsets used for averaging. We experimented with different settings, see Figure 2. As expected, we observe that a higher resolution allows for a better resolved shadow with more details, thus reflecting a trade-off between memory and quality. For generating results with varying number of offsets, we restricted the computations to the 7 smallest offsets or added further larger offsets to have a total of 21. We observe that, for this example, 7 offsets already led to pleasing results with no noticeable noise. More offsets lead to softer shadows at the expense of larger computation times and more blurring. The number of offsets, however, depends on the complexity of the scene. For example, in Figure 1(c) the shadows on the plane exhibit a bit of noise, i.e., more offsets would have been helpful.

We compare our approach against traditional shadow mapping, where a thresholding at opacity 0.5 is applied, and against a deep shadow mapping approach using a GPU implementation [Sho11]. All approaches are implemented within the same ray casting framework [Vor]. Figure 3 shows a comparison for the Walnut data set using a shadow map with resolution 1024^2. The deep shadow map uses three nodes per color channel. The CSSM uses 13 offsets. Obviously, the traditional shadow map cannot handle translucent objects, which leads to dark shadows in the interior of the walnut and on the table, see Figure 3(a). The deep shadow map and the CSSM, instead, create partial shadows for the translucent shell in the interior of the walnut as well as on the table, see Figure 3(b,c). The deep shadow map's shadows appear to be almost grey, while the CSSM creates a clear color distinction leading to a better color representation of the shadows. Due to the averaging of the offsets, the CSSM shadows appear softer, which may or may not be desirable.

The image quality of the deep shadow map could be improved by using more than three nodes per color channel at the expense of higher memory and computational costs. The CSSM uses one color buffer for storage, which are three values per fragment. While the traditional shadow map only uses one value per fragment, the deep shadow map requires four values for each node per fragment. Using three nodes as in our example already requires storing 12 values, which is four times the storage needed for the CSSM. When introducing more nodes for the deep shadow map, the memory required increases linearly with four additional values per node.

Tables 3 and 4 list the computation times for generating shadow maps and rendering with shadow maps for the three approaches with shadow map resolutions 1024^2, 2048^2, and 4096^2. The presented numbers are the average for multiple settings of different data sets, where the findings were consistent among different data sets. The generation, of course, depends on the resolution of the

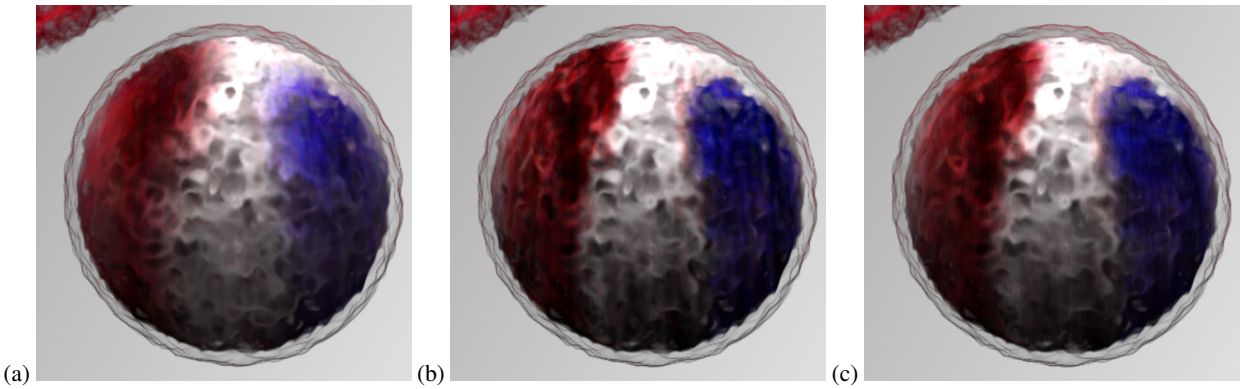

Figure 2: *Comparison of CSSM results on synthetic data set with different resolutions of the shadow map (256^2 for (a) and 1024^2 for (b,c)) and different number of samples (21 in (a), 7 in (b), and 13 in (c)).*

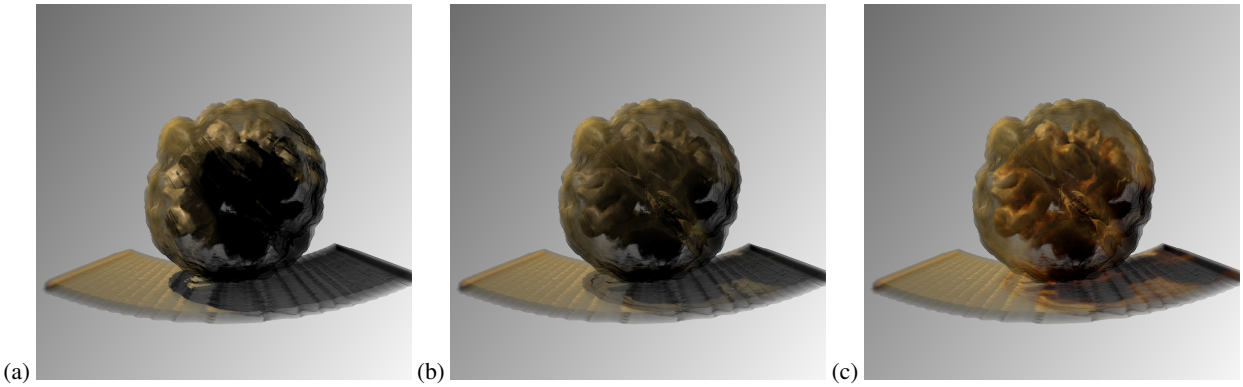

Figure 3: *Comparison of DVR with (a) traditional shadow map, (b) deep shadow map, and (c) CSSM.*

Table 3: *Computation times for generating shadow maps.*

Resolution	Shadow Map	Deep Shadow Map	CSSM
1024^2	1.0 ms	1.1 ms	1.2 ms
2048^2	2.9 ms	3.3 ms	3.6 ms
4096^2	8.5 ms	10.5 ms	11.1 ms

Table 4: *Computation times for rendering with shadow maps.*

Resolution	Shadow Map	Deep Shadow Map	CSSM
1024^2	3.2 ms	3.3 ms	4.9 ms
2048^2	3.2 ms	3.1 ms	4.7 ms
4096^2	3.3 ms	3.4 ms	4.9 ms

shadow map. All three approaches scale equally. We observe a slight increase of computation costs when using deep shadow maps or CSSMs. For the rendering step, the performance is not affected by the shadow map resolution, which is expected, as the performance depends on the resolution of the frame buffer to which the image is rendered. We observe that CSSM has a bit higher computation costs during rendering than the other two approaches. This is due to the increased amount of texture fetches for each query, which are one per offset for the CSSM (13 in our tests), while being one per node for the deep shadow map (3 in our tests) and always one for the traditional shadow map. All three approaches produce interactive frame rates.

For the creation of the renderings with CSSM in this paper, we used the same transfer function for CSSM generation and DVR, which allowed us to visually compare to other shadow mapping approaches. However, the CSSM approach is flexible and would allow us to use two different transfer functions in the two processing steps, which can be used to create novel effects, e.g., for highlighting or illustrative rendering.

One drawback of the presented approach is that it is limited to a single point light source, but this is generally true for all shadow mapping approaches. For additional light sources, additional shadow maps are needed. However, it is common to just use a single point light source for DVR.

5. Conclusions

We presented a novel shadow mapping technique for DVR, which is an alternative to using deep shadow maps. We demonstrated that CSSM can produce higher-quality colored shadows with less memory consumption and only slightly increased rendering times.

References

[BR98] BEHRENS U., RATERING R.: Adding shadows to a texture-based volume renderer. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization* (New York, NY, USA, 1998), VVS '98, ACM, pp. 39–46. URL: http://doi.acm.org/10.1145/288126.288149, doi:10.1145/288126.288149. 1

[CCF94] CABRAL B., CAM N., FORAN J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 Symposium on Volume Visualization* (New York, NY, USA, 1994), VVS '94, ACM, pp. 91–98. URL: http://doi.acm.org/10.1145/197938.197972, doi:10.1145/197938.197972. 1

[ESSL11] ENDERTON E., SINTORN E., SHIRLEY P., LUEBKE D.: Stochastic transparency. *IEEE Transactions on Visualization and Computer Graphics 17*, 8 (Aug 2011), 1036–1047. doi:10.1109/TVCG.2010.123. 2

[HKSB06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K.: Gpu-accelerated deep shadow maps for direct volume rendering. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware* (New York, NY, USA, 2006), GH '06, ACM, pp. 49–52. URL: http://doi.acm.org/10.1145/1283900.1283908, doi:10.1145/1283900.1283908. 1, 2

[HLY10] HERNELL F., LJUNG P., YNNERMAN A.: Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 16*, 4 (July 2010), 548–559. URL: http://dx.doi.org/10.1109/TVCG.2009.45, doi:10.1109/TVCG.2009.45. 1

[JSYR14] JÖNSSON D., SUNDÉN E., YNNERMAN A., ROPINSKI T.: A survey of volumetric illumination techniques for interactive volume rendering. *Comput. Graph. Forum 33*, 1 (Feb. 2014), 27–51. URL: http://dx.doi.org/10.1111/cgf.12252, doi:10.1111/cgf.12252. 1

[KN01] KIM T.-Y., NEUMANN U.: Opacity shadow maps. In *Proceedings of the 12th Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2001), EGWR'01, Eurographics Association, pp. 177–182. URL: http://dx.doi.org/10.2312/EGWR/EGWR01/177-182, doi:10.2312/EGWR/EGWR01/177-182. 2

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Comput. Graph. Appl. 8*, 3 (May 1988), 29–37. URL: http://dx.doi.org/10.1109/38.511, doi:10.1109/38.511. 1

[LV00] LOKOVIC T., VEACH E.: Deep shadow maps. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 385–392. URL: http://dx.doi.org/10.1145/344779.344958, doi:10.1145/344779.344958. 1

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (June 1995), 99–108. URL: https://doi.org/10.1109/2945.468400, doi:10.1109/2945.468400. 1

[ME11] MCGUIRE M., ENDERTON E.: Colored stochastic shadow maps. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), I3D '11, ACM, pp. 89–96. URL: http://doi.acm.org/10.1145/1944745.1944760, doi:10.1145/1944745.1944760. 1, 2, 3

[Rit07] RITSCHEL T.: Fast GPU-based Visibility Computation for Natural Illumination of Volume Data Sets. In *EG Short Papers* (2007), Cignoni P., Sochor J., (Eds.), The Eurographics Association. doi:10.2312/egs.20071033. 1

[RKH08] ROPINSKI T., KASTEN J., HINRICHS K. H.: Efficient shadows for gpu-based volume raycasting. In *Proceedings of International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2008), pp. 17–24. 1, 2

[Sho11] SHOOK A.: *Deep Shadow Maps from Volumetric Data on the GPU*. Tech. rep., University of Maryland, U.S.A., 2011. 3

[Sph] Test spheres dataset. http://lgdv.cs.fau.de/External/vollib. Stefan Roettger, University of Erlangen, Germany [accessed December 10, 2017]. 3

[Vor] Voreen: Volume rendering engine. https:/voreen.uni-muenster.de. University of Münster, Germany [accessed December 10, 2017]. 2, 3

[Wal] Walnut ct dataset. https:/www.uni-muenster.de/Voreen/download/workspaces_and_data_sets.html. European Institute for Molecular Imaging, University of Münster, Germany [accessed December 10, 2017]. 3

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph. 12*, 3 (Aug. 1978), 270–274. URL: http://doi.acm.org/10.1145/965139.807402, doi:10.1145/965139.807402. 1