Eurographics Symposium on Rendering (2006)
Tomas Akenine-Möller and Wolfgang Heidrich (Editors)

Ambient Occlusion for Animated Characters

Janne Kontkanen[1] Timo Aila[1,2]

[1]Helsinki University of Technology [2]Hybrid Graphics, Ltd.

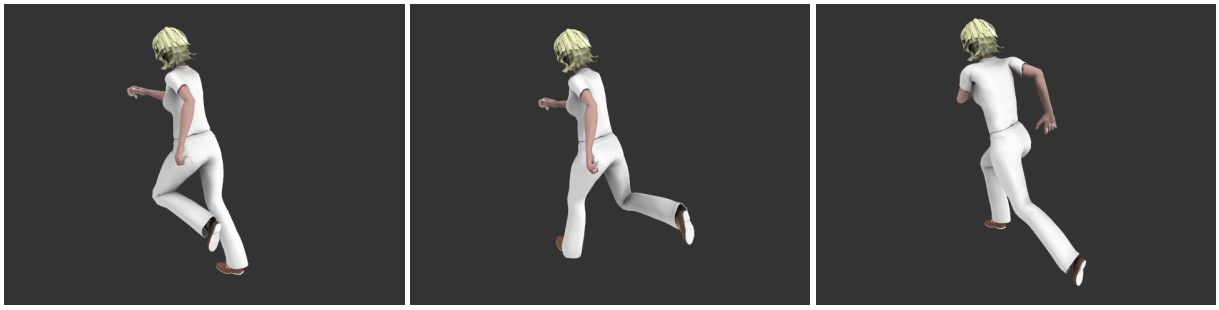


Figure 1: *Three frames from an animation. Notice the changing ambient occlusion especially around the knees and armpits. Our method produces such dynamic ambient occlusion effects with an acceptable runtime cost, and should thus be immediately applicable to real-time applications such as computer games.*

Abstract

We present a novel technique for approximating ambient occlusion of animated objects. Our method automatically determines the correspondence between animation parameters and per-vertex ambient occlusion using a set of reference poses as its input. Then, at runtime, the ambient occlusion is approximated by taking a dot product between the current animation parameters and static per-vertex coefficients. According to our results, both the computational and storage requirements are low enough for the technique to be directly applicable to computer games running on current graphics hardware. The resulting images are also significantly more realistic than the commonly used static ambient occlusion solutions.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism — Color, shading, shadowing, and texture

1. Introduction

Ambient occlusion [ZIK98] refers to the shadows arising from ambient lighting, i.e., from light that comes equally from all directions. While such special lighting conditions are rarely met in practice, ambient occlusion is a surprisingly plausible approximation in many practical applications. As has been shown by the movie industry and production rendering community [Lan02,Chr02], ambient occlusion is able to approximate effects that are otherwise attainable only by computing global illumination. For instance, sharp corners appear darker than open areas and objects cast plausible contact shadows on the surfaces they are resting on.

Compared to a full global illumination solution, ambient occlusion is significantly faster to compute. In addition, the self-occlusion of a rigid object can be computed as a preprocess and then re-used in different environments. Since the data can be stored in a texture map or as vertex attributes, the technique has a negligible run-time overhead. Therefore ambient occlusion is steadily gaining interest in the real-time graphics community [Pha04]. Recent methods can also handle the inter-object ambient occlusion of dynamically moving rigid objects by using pre-computed data structures [KL05, MMAH05, ZHL*05].

Animated objects are much more difficult to deal with because the ambient occlusion needs to be updated dynamically. Consider an animated character: if the ambient occlusion is computed for a default pose or as an average of several reference poses, the overall appearance of the character

tends to be convincing but there are typically some striking errors that shatter the illusion. These can be seen, for example, when the character raises her hand. Obviously the armpit should get lighter in this case. One solution is a more or less brute force re-computation of the ambient occlusion using optimized rasterization methods and heuristic approximations [KLA04, Bun05]. These methods do not need any a priori information about the animation and are thus universally applicable, although the generality comes with a rather costly runtime evaluation.

Our new approximation is based on parameterizing the ambient occlusion as a linear combination of animation parameters, as detailed in Section 3. The animation parameters can be pretty much anything, although in our test cases they were simply the angles of the joints. The parameterization is computed automatically from a set of reference poses and their pre-computed ambient occlusion values. As a result, if a character raises her arm, the armpit gets lighter due to its dependence on the shoulder joint parameters. See Figure 1 for another example.

The results (Section 4) demonstrate that our model produces significantly more believable results than the commonly used average ambient occlusion method while still having an acceptable runtime cost.

2. Related Work

In this section we concentrate on reviewing algorithms that compute shadows from very large light sources, i.e., from either ambient illumination or environment maps. Additionally, a substantial amount of literature exists about soft shadows from smaller area light sources, see Hasenfratz et al. [HLHS03] for a recent survey.

Accessibility shading [Mil94] is a predecessor of the ambient occlusion technique. Accessibility shading models the local variations of surface materials due to processes such as dirtying, cleaning, tearing, aging or polishing. For example, the accessibility of a surface location determines how much dirt is gathered into it. Computing ambient occlusion can be seen as computing the accessibility of light.

Ambient occlusion has become a popular technique in production rendering [Lan02, Chr02]. It is typically computed on the surfaces of each object and stored in a texture map or as vertex attributes. The most straightforward method is to use rasterization or ray tracing to sample the hemispherical visibility around the surface of the object, but a similar result can be achieved by rendering the object from multiple directions and accumulating the visibility on each surface element. In any case, the goal is to evaluate

$$A(\mathbf{x},\mathbf{n}) := \frac{1}{\pi}\int_{\Omega} V(\mathbf{x},\omega)\lfloor\omega\cdot\mathbf{n}\rfloor\, d\omega, \qquad (1)$$

on the surface of the object. Here $\mathbf{x}$ is the location and $\mathbf{n}$ is the normal vector on the receiving surface. $V(\omega,\mathbf{x})$ is the visibility function that has value zero when no geometry is visible in direction ω and one otherwise. In the above, $\int_{\Omega}$ refers to integration over a hemisphere oriented according to the surface normal $\mathbf{n}$.

Ambient occlusion is a simplified version of the obscurances illumination model [ZIK98], where the visibility function V of ambient occlusion (Equation 1) is replaced by a function of distance, giving the obscurance W:

$$W(\mathbf{x},\mathbf{n}) := \frac{1}{\pi}\int_{\Omega} \rho(d(\mathbf{x},\omega))\lfloor\omega\cdot\mathbf{n}\rfloor\, d\omega, \qquad (2)$$

in which $d(\mathbf{x},\omega)$ refers to the distance of the first intersection when a ray is shot from $\mathbf{x}$ towards ω. ρ is a function that maps the distance suitably. Iones et al. [IKSZ03] suggest $1-e^{\tau d}$, where τ is a user-defined constant.

Mendez et al. [MSC03] introduce a method for dynamically updating the obscurances information in the presence of moving objects. Obscurances are re-sampled only in a selected region of the scene by utilizing temporal coherence. However, since the obscurances are evaluated per patch, the method requires a huge number of patches to account for high quality contact shadows. Thus, while the method is usable for approximating the global illumination in large scale, the fine detail present in contact shadows requires a different approach.

Kautz et al. [KLA04] introduce a lookup-table-based rasterization method for quick computation of hemispherical occlusion. As a further optimization they use a two-level hierarchy of the triangle data in order to compute shadows from low-frequency lighting environments at interactive rates. Similarly to all methods that compute the shadows at vertices instead of pixels, artifacts are hard to avoid without highly tessellated geometry. Sattler et al. [SSZK04] compute the visibility from the vertices into a number of directions using GPU. They also utilize the coherence in the visibility function to achieve interactive frame rates with deformable objects in dynamic distant illumination. Bunnell [Bun05] approximates triangles using disks, and combines the occlusion of multiple disks heuristically. The process results in exaggerated in occlusion in many cases, but the author fixes this using a clever iterative algorithm. Furthermore, a hierarchical presentation of the disks reduces the execution time to an acceptable level.The process is still, however, quite expensive compared to methods that do not re-compute the visibility dynamically. Our method belongs to that class of approximations.

Kontkanen and Laine [KL05] pre-compute an *ambient occlusion field* around each rigid object. The field stores attenuation functions that tell approximately, for each point in space, how much occlusion the object causes to that point. The effects of multiple objects are combined heuristically, and the inter-occlusion of dynamically moving rigid objects can be approximated in real-time. Zhou et al. [ZHL*05] discuss a closely related method, which pre-computes the oc-

clusion caused by a rigid object in a set of points outside the object. The occlusion is represented using either Haar wavelets or Spherical Harmonics, and thus the contribution of multiple objects can be handled more accurately than in the method of Kontkanen and Laine, albeit at higher computational cost. Malmer et al. [MMAH05] further develop the method by Kontkanen and Laine. They use a regular 3D grid instead of a cube map and pay more attention to questions such as how to choose the resolution of the grid, and how to combine occlusion from several shadow casters.

James and Fatahalian present a general method to precompute deformable scenes [JF03]. They simulate both the deformation state and illumination with their data-driven model. This method is closely related to ours, but since we only concentrate on synthetizing the illumination (ambient occlusion), our method is simpler and thus more directly usable in applications such as computer games.

3. Ambient Occlusion for Animated Characters

The input of our algorithm consists of reference poses along with the pre-computed ambient occlusion values for all vertices. Each reference pose is represented using a pose vector $\mathbf{j}$ containing animation parameters $j_0, j_1 \ldots j_{N-1}$. The animation parameters can be, for example, angles of joints. Typically the valid range of each animation parameter is readily available from the character animation rig. In our implementation the reference poses were generated by constructing random poses from the allowable pose space, which was defined by limiting the valid range of each parameter separately. Such reference poses are particularly suitable when no a priori information of runtime animation is available.

Our goal is to establish a linear mapping from an arbitrary pose vector $\mathbf{j}$ to the approximate ambient occlusion value at each vertex a_v:

$$a_v = \mathbf{j}^T \mathbf{t}_v \tag{3}$$

where $\mathbf{t}_v$ is a per-vertex vector that has N coefficients.

We will now look at the computation of the coefficient vectors $\mathbf{t}_v$, and then discuss the implicit assumptions and limitations built into this model.

3.1. Per-vertex coefficients

Given the matrix of pre-computed ambient occlusion values $\mathbf{A}$ and animation parameters $\mathbf{J}$ for all the reference poses, our goal is to establish the mapping:

$$\mathbf{A} = \mathbf{JT} \tag{4}$$

in which the matrices are built as

$$\mathbf{J} = \begin{pmatrix} \text{all animation parameters of reference pose 0} \\ \text{all animation parameters of reference pose 1} \\ \vdots \end{pmatrix}$$

$$\mathbf{T} = \begin{pmatrix} \text{effect of animation parameter 0 to all vertices} \\ \text{effect of animation parameter 1 to all vertices} \\ \vdots \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} \text{per-vertex ambient occlusion of reference pose 0} \\ \text{per-vertex ambient occlusion of reference pose 1} \\ \vdots \end{pmatrix}$$

Each column of $\mathbf{T}$ describes how all the animation parameters affect a particular vertex. This is exactly what we want to solve. Each column of $\mathbf{A}$ contains the ambient occlusion of all reference poses in one vertex.

Equation 4 is a group of overdetermined systems of linear equations since there, in general, are more reference poses than there are animation parameters. So, there is no exact solution $\mathbf{T}$. However, an optimal solution in MSE-sense is obtained by:

$$\mathbf{T} = \mathbf{J}^{+}\mathbf{A} \tag{5}$$

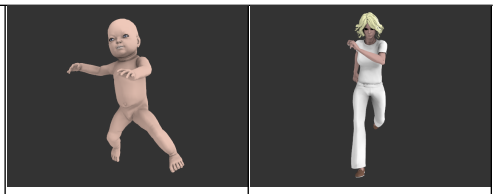
where $\mathbf{J}^{+}$ refers to the pseudo-inverse [GHG96] of the pose matrix $\mathbf{J}$. Once $\mathbf{T}$ has been computed, the run-time evaluation at each vertex can be done according to Equation 3. Now $\mathbf{t}_v^T$ is a column of $\mathbf{T}$ that corresponds to the vertex, and $\mathbf{j}$ is the pose vector containing the animation parameters for an arbitrary pose. The amount of runtime work is thus proportional to the number of animation parameters.

So far our discussion overlooks one important special case: what if the ambient occlusion at a vertex does not depend on any of the animation parameters? In order to handle this situation, we augment $\mathbf{J}$ with one column that contains constant data, i.e., is the same for all reference poses. This creates a new virtual animation parameter that captures ambient occlusion that is not dependent on any of the actual animation parameters.

3.2. Assumptions and Limitations

There are two simplifying assumptions built into this model. First, the model assumes that the ambient occlusion depends linearly on the animation parameters. In many cases this is an acceptable approximation, e.g., the ambient occlusion caused by upper arm on lower arm is probably almost linearly dependent on the angle of the elbow joint, but admittedly not all animation parameters behave so nicely.

The second assumption is that the animation parameters can be handled independently so that the ambient occlusion is simply a sum of the occlusions resulting from individual parameters. Obviously this assumption does not always hold. For instance, whether or not the character's hand causes a contact shadow on her nose depends on many animation parameters, such as joint angles in the shoulder and elbow. It would be possible to capture such "higher-order" effects as well, but only at significantly higher computational and storage cost. Therefore we decided to disregard such effects in favor of faster runtime implementation.



	Baby	Jessi
Vertices	11281	41947
Params	73	54

Table 1: *The scenes used in test animations.*

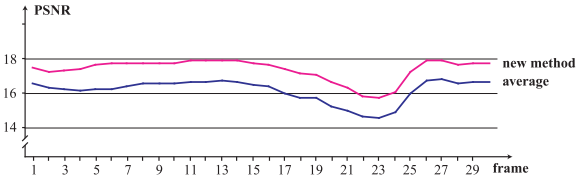


Figure 3: *Peak signal-to-noise ratio (PSNR) of ambient occlusion during 30 frames when Jessi is running. The error is evaluated against a ray traced reference solution. Higher PSNR indicates less error. Clearly, the new method gives consistently better approximation of ambient occlusion.*

Provided that the precomputation has been done with a large amount of reference poses, the effects that cannot be represented by our model get averaged away and thus do not in general manifest themselves as distracting artifacts.

4. Results

We tested our algorithm in two animated scenes, Baby and Jessi (Table 1). The tests were executed on a 3GHz Pentium 4 with 1GB of memory. The ambient occlusion of the reference poses was computed by custom ray tracer, and took 6 hours for 1000 random reference poses in the Baby scene and 20 hours for 600 random poses of Jessi. Obviously faster methods, e.g. GPU-based computation, could be exploited in the pre-computation stage, but we did not investigate that. The Equation 5, including the pseudo-inverse took only a few seconds in Matlab. The reference poses were generated by choosing random values for the animation parameters. To avoid illegal poses, the knowledge of the valid ranges of the parameters was utilized.

The ground truth solutions were computed using ray tracing with 1000 samples per vertex. We compared our method against a static average solution, which was computed by simply averaging the ambient occlusion from all the reference poses. Figure 2 shows a comparison of the errors arising from our method as well as from the comparison method. While the difference is fairly clear in the still images, it is even more pronounced in animations. Figure 3 shows the peak signal-to-noise ratios (PSNR) when compared to the ground truth. As can be seen, our method results in consistently higher PSNR and thus lower error. Further comparison images are shown in Figures 4 and 5.

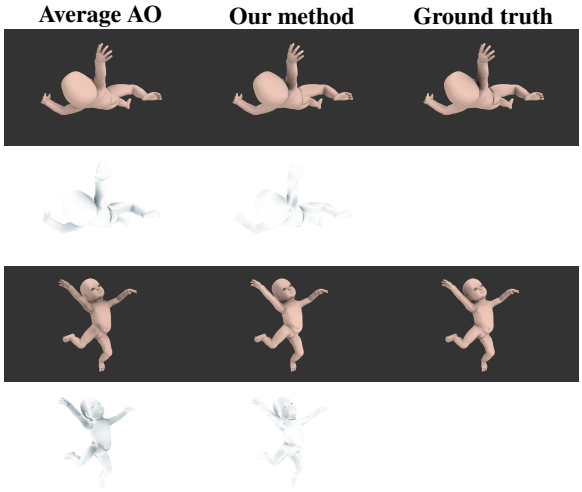


Figure 4: *Static average ambient occlusion (**Left**) and our method (**Middle**) compared against the ground truth (**Right**). The ground truth was obtained by tracing 1000 quasi-random rays per vertex. The second and fourth rows show the difference images against the ground truth.*

The runtime cost of our method consists solely of the per-vertex dot product between the current animation parameters and the pre-computed weights. In Baby and Jessi this means 73 and 54 scalar multiply-accumulations per vertex, respectively. The amount of computations is roughly four times the cost of transforming a vertex. For example, NVIDIA 7900 GTX could theoretically process hundreds of millions of vertices per second in these scenes, assuming 650MHz clock and 8 vertex shaders running in parallel, each being a 4-way SIMD unit. Admittedly, in practice the performance would be somewhat lower, but in any case we did not consider it necessary to start experimenting with an optimized runtime implementation. The storage requirements using 16-bit floats are 1.6MB for Baby and 4.3MB for Jessi.

5. Discussion and Future Work

The presented new ambient occlusion method for animated characters should be usable in computer games today. The improved accuracy, compared to the the static average ambient occlusion, comes with a reasonable additional cost. One practical possibility would be to select either the old or the new technique at runtime for each character, and thus implement level of detail for ambient occlusion.

In this paper we used a dense run-time representation for storing the per-vertex coefficients. However, in a typical scene only a small portion of the coefficients are significantly large. For, instance in the baby-scene 95% of the coefficients had an absolute value less than 0.1 and 53% were smaller than 0.01 (for computing these statistics, the animation parameters were scaled and translated to range $[-0.5, 0.5]$). Clamping these values to zero would enable

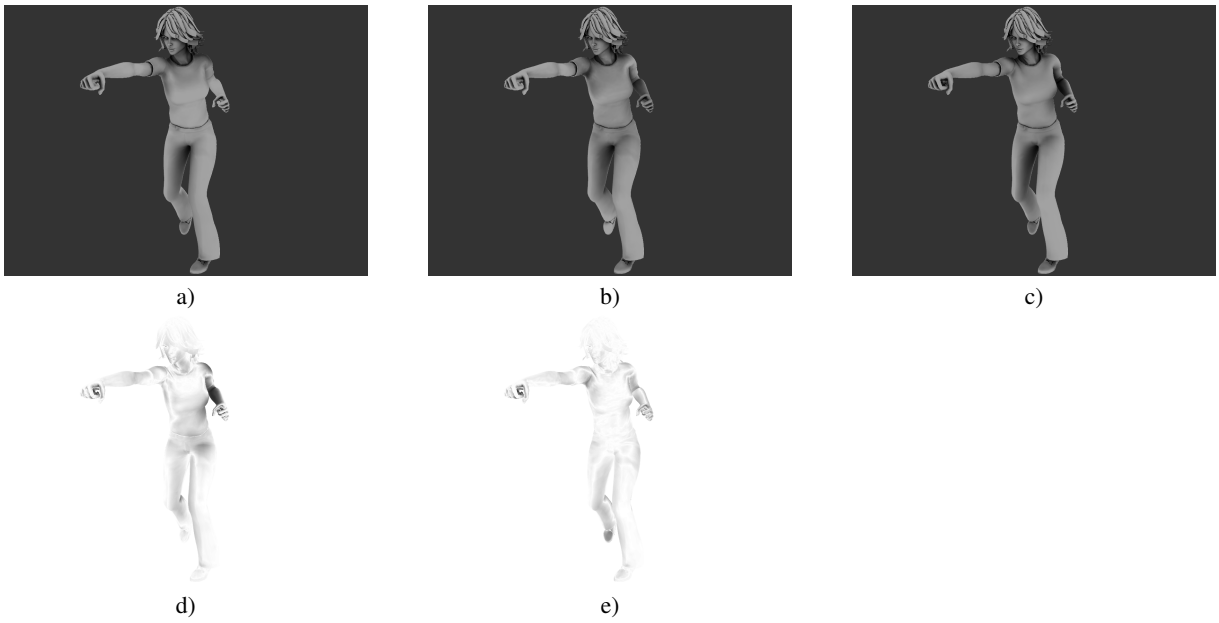


Figure 2: *A comparison of the average ambient occlusion model (a) and our new technique (b) against the ground truth (c). As can be seen from the corresponding difference images (d) and (e), our new model is more accurate. The real difference, however, is much more apparent in animations. The difference images have been exaggerated for illustration purposes.*

simple compression techniques. Alternatively, it might be worthwhile to experiment with clustered PCA compression of the coefficients.

Another interesting possibility would be to try using something else than joint angles as animation parameters. For example, the distance of two joints might have advantages, although that would easily result in data amplification, and more elaborate compression techniques would be required.

Acknowledgements Thanks to Jaakko Lehtinen for his insightful comments. Janne Kontkanen was supported by Bitboys, Hybrid Graphics, Remedy Entertainment, Anima Vitae, and the National Technology Agency of Finland. Timo Aila was supported by the Academy of Finland. Jessi and Baby meshes were exported from Poser 6 by Curious Labs.

References

[Bun05] BUNNELL M.: Dynamic ambient occlusion and indirect lighting. In *GPU Gems 2* (2005), Addison Wesley, pp. 223–234.

[Chr02] CHRISTENSEN P. H.: Note #35: Ambient Occlusion, Image-Based Illumination, and Global Illumination. *PhotoRealistic RenderMan Application Notes* (2002).

[GHG96] GENE H. GOLUB C. F. V. L.: *Matrix Computations, 3rd edition*. Johns Hopkins University Press, 1996.

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A Survey of Real-Time Soft Shadows Algorithms. *Computer Graphics Forum 22*, 4 (2003).

[IKSZ03] IONES A., KRUPKIN A., SBERT M., ZHUKOV S.: Fast, Realistic Lighting for Video Games. *IEEE Computer Graphics and Applications 23*, 3 (2003), 54–64.

[JF03] JAMES D. L., FATAHALIAN K.: Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 879–887.

[KL05] KONTKANEN J., LAINE S.: Ambient occlusion fields. In *Proc. I3D* (2005), ACM Press, pp. 41–48.

[KLA04] KAUTZ J., LEHTINEN J., AILA T.: Hemispherical Rasterization for Self-Shadowing of Dynamic Objects. In *Rendering Techniques 2004 (Proc. EGSR)* (2004), pp. 179–184.

[Lan02] LANDIS H.: RenderMan in Production, ACM SIGGRAPH 2002 Course 16, 2002.

[Mil94] MILLER G.: Efficient algorithms for local and global accessibility shading. In *Proc. SIGGRAPH 94* (1994), pp. 319–326.

[MMAH05] MALMER M., MALMER F., ASSARSSON U., HOLZSCHUCH N.: *Fast Precomputed Ambient Occlusion for Proximity Shadows*. Tech. Rep. RR-5779, INRIA, 2005.

[MSC03] MÉNDEZ A., SBERT M., CATÀ J.: Real-time Obscurances with Color Bleeding. In *Proceedings of the 19th spring conference on Computer graphics* (2003), pp. 171–176.

[Pha04] PHARR M.: Ambient occlusion. In *GPU Gems* (2004), Fernando R., (Ed.), Addison Wesley, pp. 667–692.

[SSZK04] SATTLER M., SARLETTE R., ZACHMANN G., KLEIN R.: Hardware-accelerated ambient occlusion computation. In *Proc. VMV '04* (2004), pp. 119–135.

[ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.-Y.: Precomputed shadow fields for dynamic scenes. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (2005), 1196–1201.

[ZIK98] ZHUKOV S., IONES A., KRONIN G.: An ambient light illumination model. In *Rendering Techniques '98 (Proc. EGWR)* (1998), pp. 45–55.

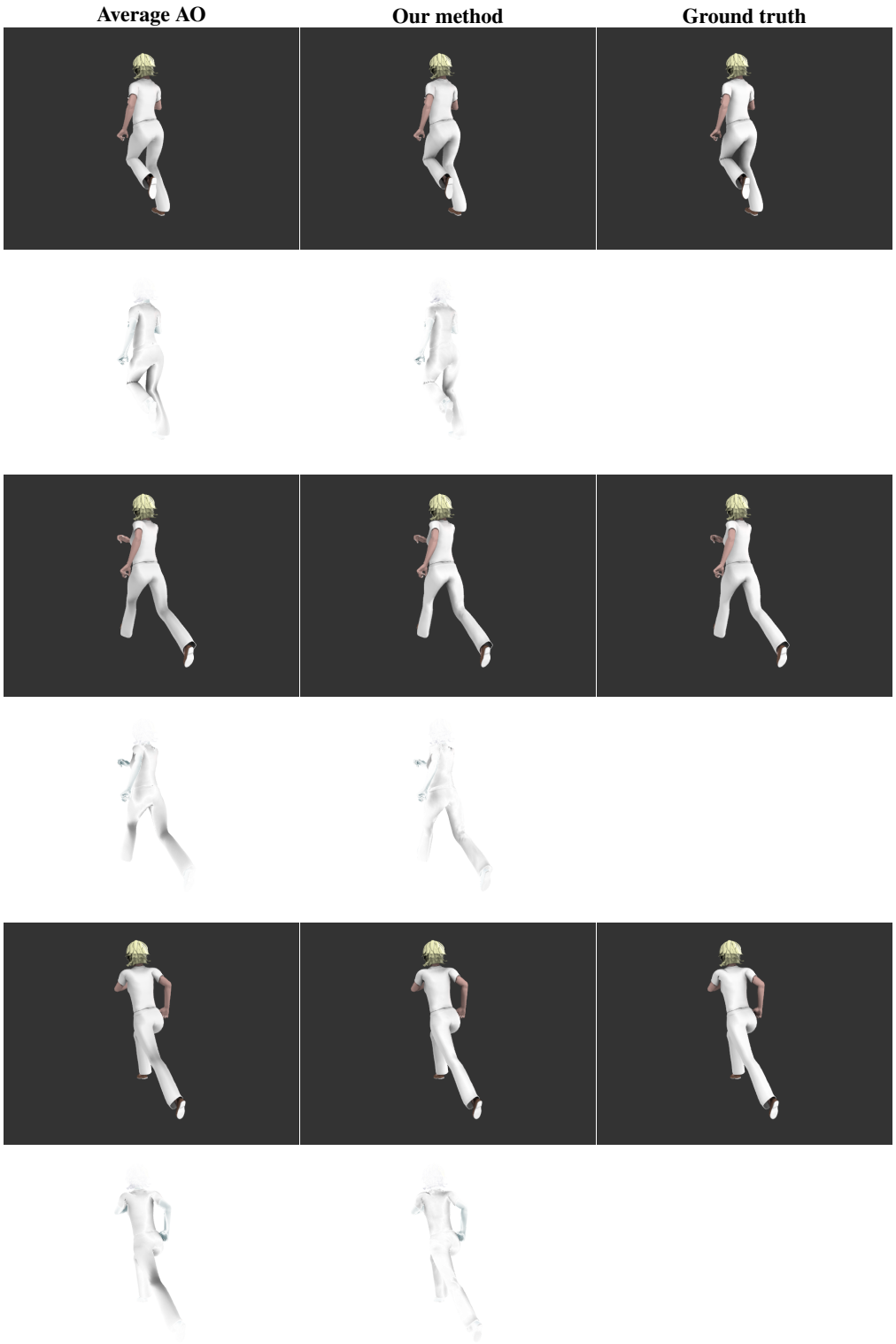


Figure 5: *Static average ambient occlusion (**Left**) and our method (**Middle**) compared against the ground truth (**Right**). The ground truth was obtained by tracing 1000 quasi-random rays per vertex. The second, fourth, and sixth rows show the difference images against the ground truth.*