Fast Geometric Sampling for Phong-Like Reflection

Shuzhan Yang[1,2] and Han Su[1,2](✉)

[1] School of Computer Science, Sichuan Normal University, Chengdu, China
jkxy_sh@sicnu.edu.cn

[2] Visual Computing and Virtual Reality Key Laboratory of Sichuan Province, Chengdu 610066, China

Abstract. Importance sampling is a critical technique for reducing the variance of Monte Carlo samples. However, the classical importance sampling based on the Bidirectional Reflectance Distribution Function (BRDF) is often complex and challenging to implement. In this work, we present a simple yet efficient sampling method inspired by Phong's reflectance model. Our method generates samples of rays using geometric vector operations, replacing the need for BRDF. We explain our implementation of this method on WebGL and demonstrate how we obtain per-pixel random numbers in GLSL. We also conduct experiments to compare our method's speed and patterns to the Phong distribution. The results show that our sampling process can simulate reflections similar to Phong, but is about three times faster than traditional Phong or other BRDF importance sampling methods. Our sampling method is applicable to both real-time and offline rendering, making it a useful tool for computer graphics applications.

Keywords: Sampling · BRDF · Monte Carlo

1 Introduction

Importance sampling is a critical sampling strategy used in Monte Carlo integration to improve the efficiency of ray tracing. Importance sampling usually involves generating samples according to the Bidirectional Reflectance Distribution Function (BRDF). Ideally, The sampling distribution should be consistent with the BRDF curve to ensure that important proportions are sampled more. However, conventional methods of BRDF importance sampling can be complex and slow. Meanwhile, several accurate reflectance models have been proposed, but some lack simplicity. For instance, GGX is a popular model requiring a complex process to properly sample the Visible Normals Distribution Function (VNDF). Heitz [6,7] described this process, which involves projecting the hemispherical surface along the ray, uniformly sampling it on the projection surface, calculating the coordinates of the point on the hemispherical surface, transforming the hemispherical surface into a semi-ellipsoidal surface, calculating the normals, and finally transforming the normals to world space. Generally, this process requires several coordinate transformations, which add to the complexity and computational overhead of the importance sampling.

In addition to Monte Carlo sampling, there exist various techniques that aim to produce real-time glossy reflections. One such technique involves the use of precomputed blurred mipmaps, as demonstrated by Ashikhmin et al. [1] and McGuire et al. [10] to simulate Phong or Blin-Phong reflections. However, these techniques are limited in their ability to handle dynamic lighting or occluded objects, leading to decreased accuracy and versatility in certain scenarios.

To improve the efficiency of Monte Carlo ray tracing, we propose a new method for sampling that can bring a significant increase in performance by simplifying the sampling process. Specifically, we aim to skip coordinate transformations and obtain vectors directly in world space. In Sect. 5 of this paper, we demonstrate the effectiveness of our approach, showing that it can result in a steady increase in rendering performance. While the improvements achieved may seem small, given the large number of times that sampling is performed per frame, even a small gain in efficiency can be observed and is worth pursuing.

Given the complexities of BRDF importance sampling, we seek an efficient reflectance model or an approximation that simplifies the sampling process. In this paper, we focus on Phong [11], a widely used BRDF model [8] that is simple and fast, but has some limitations. Phong BRDF is simple and fast though it clearly has some limitations as compared to other modern BRDFs: non-energy conserving, non-reciprocal. We propose a method to generate samples similar to Phong but faster, while also making up for some of its shortcomings. In Sect. 2, we introduce some details of Phong, and in Sect. 3 about its sampling routine.

In this paper, we introduce a simple geometric sampling routine for simulating Phong-like isotropic specular reflections. We review the BRDF importance sampling procedure and identify slow parts of sampling process, and then compare with traditional importance sampling in terms of efficiency and similarity. Finally, we implement on a WebGL application and discuss how to extend for fast refraction.

2 Related Work

2.1 Phong-Like Distribution

In 1975, Phong proposed a simple shading model [11] that has since been widely used in various graphics applications, particularly in the rasterization pipeline, owing to its simplicity. However, this model is not suitable for ray tracing. Even though the Phong model seems straightforward, it still requires rotating vectors to generate random rays around a given light direction. On the other hand, more precise models were favored in ray tracing, and accuracy was given more importance than speed. Taking inspiration from Phong's work, we devised a method for generating samples without the need for rotation.

The limitations of the Phong model are evident when compared to other modern distributions. It is a rough model and cannot accurately represent rays that form an obtuse angle against the perfect specular reflective direction. Additionally, compensating for energy loss is required when ray samples are below the surface. Despite these limitations, the Phong model is still widely used in the industry, particularly in cases where a perfectly photo-realistic result is not required. For instance, Gooch et al. [5] used the Phong lighting model as a basis for approximating human-drawn technical illustrations. Nowadays, the Phong shading model is available in almost every graphics library.

Inspired by the Phong model, we want to find a sampling framework that accommodates such reflections that are similar to Phong. We define Phong-like reflection as the reflectance distribution of rays that are symmetric with respect to the perfect specular reflective direction. These reflections are generally simple because they are irrelevant to the normal(though we still have to deal with cosine weight and fresnel).

3 Importance Sampling for Phong

Considering the direct light from a light source, the original Phong model for specular reflection is:

$$L_s = f_s E = k_s \cos^n \alpha E \tag{1}$$

where L_s is the reflected specular radiance,f_s the BRDF, E the incidence irradiance from the light, k_s the specular reflectivity, n the exponent defining surface smoothness, $\alpha \in [0, \frac{\pi}{2}]$ the angle between the perfect specular reflective direction and the outgoing direction.

But it is difficult to sample the Phong distribution while maintaining energy conservation. Lafortune et al. [9] proposed a modified model to keep energy conservation approximately. This way of doing importance sampling for BRDF includes these steps:

1. First, get the direction of the incoming ray which is usually in world space. If the ray starts from the camera, the camera location minus the point location(interpolated from the vertex shader) gets the incoming ray direction which is in world space. In other cases like path tracing, the incoming ray might be a reflection ray of another point, but this ray also has to be in world space.
2. Set up a local coordinate system usually based on the normal (Phong uses the ray direction as the base). Tom Duff summarised several ways of building an orthonormal basis [4]. But this process is not without cost, and according to their survey, this takes 8–20 ns.
3. Then, use the probability density function (pdf) to generate one or more samples of rays. Take Phong, for example. The specular part of modified Phong BRDF [9] is:

$$f_s = k_s \frac{n+2}{2\pi} \cos^n \alpha, \alpha \in [0, \frac{\pi}{2}] \tag{2}$$

where $\alpha \in [0, \frac{\pi}{2}]$ the angle between the perfect specular reflective direction and the outgoing direction, k_s the specular reflectivity, n the factor in defining how smooth the surface is, x the normalization factor.

But BRDF measures the viewing brightness of the surface rather than light density in space. So ideally, in order to do a complete importance sampling, we want to take into account the $\cos\theta$ factor and generate samples according to cosine weighted BRDF like:

$$x \cos^n \alpha \cos\theta \tag{3}$$

where θ is the angle between the normal and the outgoing ray direction. And the pdf should integrate to 1.

$$\iint_{\Omega} x \cos^n \alpha \cos\theta \sin\theta d\theta d\varphi = 1 \tag{4}$$

Calculating the analytical value of x is challenging due to the coherence between α and θ, as highlighted in [2]. As a compromise, a common approach is to omit $\cos\theta$, as suggested by Lafortune et al. [9], and use the following pdf:

$$pdf = \frac{n+1}{2\pi} \cos^n \alpha, \alpha \in [0, \frac{\pi}{2}] \tag{5}$$

4. Transform the resulting vector from local space to world space by rotation. This process can be done with vector operations or a matrix.

It is apparent that classical importance sampling is not straightforward. In Sect. 3 we are going to describe how to get rid of steps 2 and 4 and generate samples just in world space.

4 Geometric Sampling

The reason we call it geometric sampling is that we see the resulting samples as vectors and use vector operations to manipulate them. Our task is to generate a couple of rays given an incoming ray direction. In a nutshell, our approach consists of these steps:

1. Generate a random vector.
2. Move it up a little bit.
3. Add (blend) it to the ray.
4. Normalize(optional).

4.1 Intuition

Inspired by Phong distribution, first we can assume the surfaces is perfectly smooth and calculate the reflection ray r (which has been normalized), then generate samples r_i based on r. In other words, we want a function f:

$$r_i = f(r, V_{rand}) \tag{6}$$

where V_{rand} is a vector $(x_1, x_2, x_3 ... x_n)$ that is generated by random numbers $(\xi_1, \xi_2, \xi_3 ...)$. This way, we are able to generate multiple samples of rays followed by a single reflection operation by varying the random numbers. By comparison, the conventional methods of sampling the normals generate one sample for a reflection operation. Notice that, although glsl has a built-in "reflect()" function, it can be relatively slow. According to Khronos Group's document, it is calculated as I - 2.0 * dot(N, I) * N, where I is the incoming ray and N the normal.

The intuition of Phong is to generate random rays by rotating the direction of the reflected ray, but we want to do it in another way. If a normalized vector p rotates around another normaliz vector p_0 by an angle θ, we can get the resulting p' that has the same length as p using the following formula:

$$p' = p\cos\theta + (1-\cos\theta)(p_0 \cdot p)p_0 + \sin\theta(p_0 \times p) \tag{7}$$

Basic arithmetic operations such as addition and subtraction are computationally efficient, requiring only one CPU cycle to execute. However, more complex operations such as sine and cosine functions are considerably slower and can be a power-intensive process. Although some operations can be parallelized, the computational cost of such operations can still be high. The above rotation process can also be done using a transform matrix, but not necessarily faster.

4.2 Add(blend) with the Ray

Since vector rotation is so costly, the original thought is to replace it with vector addition. First, we generate a random vector V_{rand} from the position of one pixel in world space. Then we add it to the reflection direction r. Now that we have the result ray r_i, we can intersect it with the area light and do Monte Carlo integration. That seems nice and clean, but another problem occurs. Considering a very rough surface like Lambertian, the reflection rays are completely random, which means we need the coherence of the original ray to be zero otherwise the added vector should be infinite. In our practice (3), r needs to be blended with R_s (short for roughness), which can limit the vector length to a finite value.

$$r_i = f(r, V_{rand}) = r(1-R_s) + V_{rand}R_s \tag{8}$$

We can transform the reflection ray r to obtain samples r_i, as shown in Eq. 8. If maintaining the length of r_i is necessary, it can be further normalized. This step can be performed using GLSL's built-in function "mix()", which is one of the fastest built-in functions in GLSL.

4.3 Generate Desired V_{rand}

The real question is how to find a V_{rand} distribution to make r_i satisfy the reflection that we want. In theory, we can simulate any BRDF using (3), as long as we figure out the V_{rand} distribution in 3D space. For Phong-like distribution, we consider its V_{rand} as a form of symmetric distribution inside a sphere, and this distribution can be broken down into two parts: one is uniform distribution on a sphere, the other is a factor that controls the distance of samples to the sphere center. We generate a 3d random vector using three uniform random numbers ξ_1, ξ_2, ξ.

$$\begin{cases} V_x = 2\xi_1 - 1, \\ V_y = \sqrt{1 - V_x^2}\sin(2\pi\xi_2), \\ V_z = \sqrt{1 - V_x^2}\cos(2\pi\xi_2), \end{cases} \tag{9}$$

Now that we have the uniform sphere distribution, we add $N \cdot R_s$ to avoid energy loss (discussed in Sect. 4.4), and then multiply it with another factor $D(\xi)$ to get the distribution of V_{rand}. $D(\xi)$ is a critical parameter that actually controls the size and shape of highlights. In Sect. 4.6, we will discuss various values of $D(\xi)$ in more detail.

$$V_{rand} = (V_{xyz} + N * R_s)D(\xi) \tag{10}$$

where D(ξ) is used to control the radius of distribution, Rs the roughness. D(ξ) also can be precalculated. Equation 8, 9, 10 are the core formulas of our method, and we leave lines of glsl codes of how they are implemented in the appendix.

4.4 Avoid Energy Loss

Sometimes rays are intercepted by the surface (Fig. 9), which will cause energy loss. For energy conservation, it is possible to check these situations first and then invert those intercepted rays. That is the same issue that Heitz mentioned [6], the difference between NDF and VNDF. Traditional reject sampling of NDF is inefficient, and causes fireflies. And we use a simpler and faster solution in Eq. 10.

Another thing that should be considered is the cosine weight factor. In the previous description, we generate uniform samples on a sphere and scale it using D(ξ). This is a very quick approximation when the light direction is not close to the surface. But when the roughness is close to 1, we must take into consideration the $\cos\theta$ between N and r_i. We know that for Lambertian reflection, the probability density forms a sphere on the surface. Notice that we blend r with R_s to get r_i.

Our solution for these two problems is to move all the samples up a little bit. In formula 10 we add $R_s * N$ to r_i, not only making an approximation of $\cos\theta$ weighted integral, but also moving all the invisible rays to be exactly above the surface, which means we don't need an extra calculation to decide if a ray is intercepted by the ground.

It is feasible because $\cos\theta$ has little impact when R_s is small. On the other hand, the precision of Phong itself is not high, especially when R_s increases, whereas our model blends into Lambertian, so it is not inappropriate to simulate $\cos\theta$ roughly. We measured the precision in our experiments in Fig. 10.

4.5 Monte Carlo Integration

In the previous section, we generated samples, and the next step is to complete the Monte Carlo integration. Since it is often difficult to obtain a closed-form expression for BRDF, we will represent it as $f(r, r_i)$ for convenience. Assuming that our generated samples r_i follow a probability density function of $p(r_i)$, we can calculate the radiance of a point x as seen from direction r_0 using the following equation:

$$\begin{aligned} L(x, r_0) &= \int_{\Omega} L(r_i) f(r, r_i) \cos(\theta) \mathrm{d}r_i \\ &= \int_{\Omega} L(r_i) \frac{f(r, r_i)}{p(r, r_i)} p(r, r_i) \cos(\theta) \mathrm{d}r_i \end{aligned}$$

where $\rho(r, r_i)$ is BRDF, θ the angle between normal and r_i. According to importance sampling, we have Monte Carlo integral at point x:

$$L(x, r_0) = \lim_{n \to +\infty} \frac{1}{n} \sum_{i=1}^{n} L(r_i) \frac{f(r, r_i)}{p(r_i)} \cos(\theta) \tag{11}$$

This equation is the common form of importance sampling. And in Sect. 4.4 we have tried additional processes to fit $p(r_i)$ to approximate $f(r, r_i)\cos(\theta)$, but we still have not considered the Fresnel effect, so the Frenel coefficient $F(r_i)$ is left in the integral.

$$L(x, r_0) \approx \lim_{n \to +\infty} \frac{1}{n} \sum_{i=1}^{n} L(r_i) F(r_i) \tag{12}$$

in which $L(r_i)$ is the radiance received by r_i, N the normal, $F(r_i)$ is Fresnel coefficient.

4.6 Distribution

The form of distribution is not fixed but mainly depends on three factors: D(ξ), which we use to control the shape of the highlight; Rs, the roughness which determines how $\cos\alpha$ weight influence the distribution; View angle as compared to normal, which affect the accuracy of our approximation towards the microfacet model. But if we only take D(ξ) for consideration and ignore the 3.3 step, it is actually percentile point function (PPF), we can inverse it to get the following cumulative distribution function (CDF):

$$cdf(\xi) = D^{-1}(\xi), \xi \in [0, 1] \tag{13}$$

where ξ measures the deviation from the original ray r. D(ξ) $= \frac{1}{R_s} ||r_i - r + r \cdot R_s||$. The probability density function is the derivative of cdf:

$$pdf(\xi) = \frac{\mathrm{d}D^{-1}(\xi)}{\mathrm{d}\xi}, \xi \in [0, 1] \tag{14}$$

First, let us consider when roughness = 1. It can be proved that the distribution is equal to the Lambert distribution.

When the roughness is not 1, we simulate it and draw the probability density in Fig. 1. These samples are observed from the r direction, and the maximum radius is equal to R_s. Definitely, we want it to look close to a normal distribution while keeping it as simple as possible. Also, we compute the pdfs with respect to ξ and draw them in Fig. 2. We test a few examples, and $\sqrt{\xi}$ is supreme for its simplicity while $1-\sqrt{1-\sqrt{\xi}}$ has a smooth edge.

For instance, let's consider the function $D(\xi) = -\log(1-\sqrt{\xi})$, which can map the ξ values from the interval (0, 1) to (0, infinity), thus providing a long tail distribution similar to the GGX model. However, this type of function can cause rays to cross the surface, which disrupts our procedure and requires us to detect whether a ray is intercepted, and compensate for the resulting energy loss. Moreover, this can also lead to visible noise, making it an inefficient choice for practical implementations.

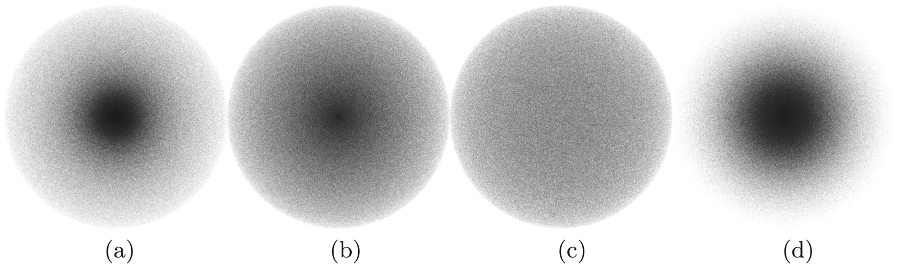

Fig. 1. Comparing different distribution for different D(ξ). (a): D(ξ) = ξ, (b): D(ξ) = $\sqrt{\xi}$, (c):D(ξ) = $\sqrt[3]{\xi}$, (d): D(ξ) = $1-\sqrt{1-\sqrt{\xi}}$. Viewing from the r direction(the reflection ray of a smooth surface), we record the position of the samples. The maximum radius of each distribution is R_s. In the first picture, the samples are too close to the center, whereas the second and third have a sharp edge.

4.7 Compared to Phong

The modified Phong BRDF for specular reflection [9] can be written as $k_s \frac{n+2}{2\pi}\cos^n\alpha$. One limitation is that θ must be clamped to [0, $\frac{\pi}{2}$] [9], while in our method, rays of $[\frac{\pi}{2},\pi]$ are also included, which means we can handle well the transition between roughness 0 to roughness 1.0. Figure 4 shows the variance for metallic materials at different roughness.

In Fig. 5, we render a glossy surface under an area light for a few commonly used BRDFs. The results show that the result of ours2 is pretty close to Phong. In the first picture, the highlight of ours1 is darker with a sharper edge as expected. The result fits the pdf curves in Fig. 3, since the distribution of ours1 is scattered from the center. Note that GGX appears to be much rougher at the same roughness as ours. Therefore we use 0.1 for GGX and 0.3 for ours, but still, GGX has a wider gradient around the highlight than the others.

In Fig. 6, we add environment to our scene, and as expected, the difference is small in general. In the first row, they all seem very similar at a low roughness.

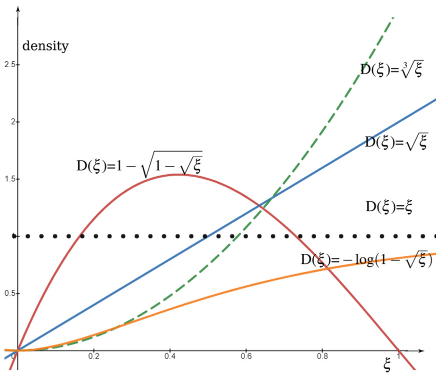

Fig. 2. Pdfs for different D(ξ). Normally, ξ should be restricted to [0, 1], otherwise the operation in 3.3 to prevent energy loss is invalid.

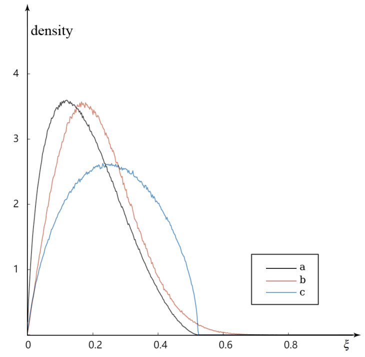

Fig. 3. $pdf(\alpha)$ for different distribution. (a): $D(\xi) = 1 - \sqrt{1-\sqrt{\xi}}$, (b): Phong, (c): $D(\xi) = \sqrt{\xi}$.

While in the second row, ours1 seems a bit rougher, although its R_s is 0.5, equal to ours2. But we can also notice that the Phong model becomes inaccurate for its sharpest highlight in the second row. As we mentioned, α is clamped to $[0, \frac{\pi}{2}]$.

Regarding other advantages: We can save the major computation of random numbers generation and reuse it every frame from a texture; No need to convert between world space and tangent space. All the samples of rays are on the same side as normal, not intercepted; We can do the reflection operation once and generate multiple rays.

Blinn [3] pointed out that the experimental results generally match the Phong model, except when the view direction is almost parallel to the surface. Phong doesn't actually sample according to a particular normal distribution, so the actual distribution is not as accurate as Torrance-Sparrow's [12] model. However, in actual production, the tiny difference would not be as obvious as it is in the experiment if we use a normal map or bump map for the material and the normals constructed from NDF will likely be hidden by the normal map.

5 Performance

It is actually hard to measure the accurate time of GPU drawing a frame. To estimate the efficiency of our method, we make a simple quad filling up the screen of 1880 * 1812, and use a triangle light on it. Then we switch to different BRDFs and record their rendering time. The frequency of our GPU rx580 is restricted to 300 mhz to make the result noticeable. With our approach, the time spent on sampling per frame is within 0.1 ms, which is much shorter than the traditional ways.

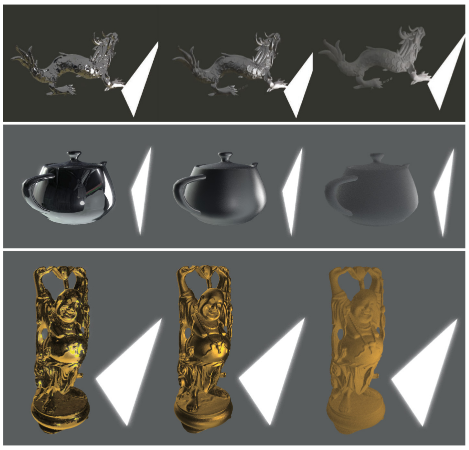

Fig. 4. Unlike Phong, our method is able to handle well the transition between smooth and rough surfaces.

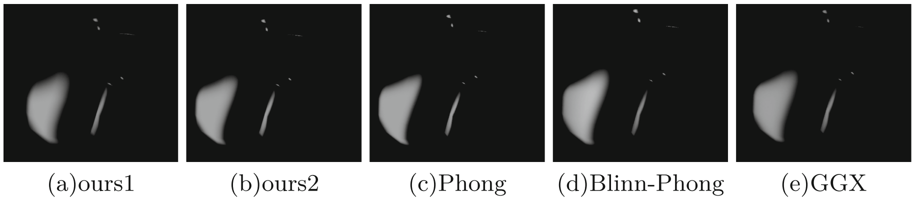

Fig. 5. Shape of highlights comparing different BRDFs. ours1: D(ξ) = $\sqrt{\xi}$, ours2: D(ξ) = $1 - \sqrt{1 - \sqrt{\xi}}$

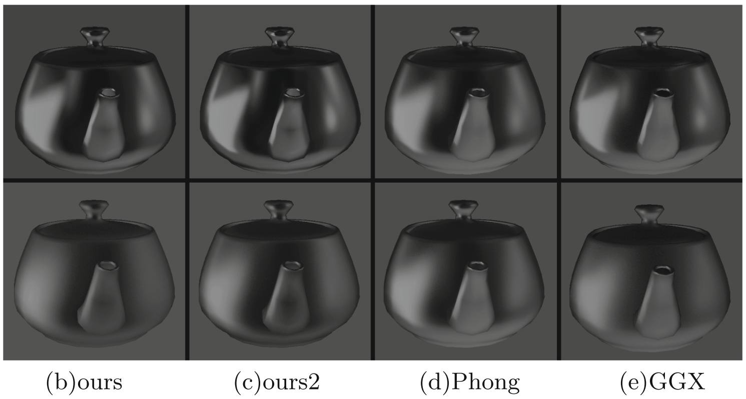

Fig. 6. Teapot under the environmental light compared to other BRDFs. ours1: D(ξ) = $\sqrt{\xi}$, ours2: D(ξ) = $1 - \sqrt{1 - \sqrt{\xi}}$

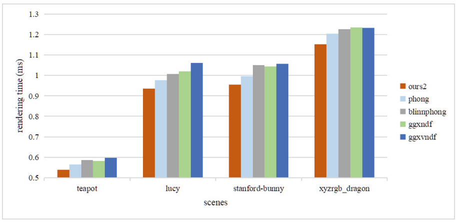

Fig. 7. Performance comparison across different scenes. The result shows that our method gains a steady performance increase across different scene. Note that as the polygonal faces increases, more time is spent on other process.

	1sample(without raytracing and base drawing)	100samples (without raytracing)	Drawing latency
phong	0.256	27.3	42.7
blinn-phong	0.289	30.6	46.0
GGX	0.388	40.5	56.8
ours	0.067	8.36	23.2
Base drawing time	1.71	1.71	1.71
Raytracing time (including base draw)	0.178	19.5	19.5

Fig. 8. performance comparison. We tested rendering a simple glossy quad, and the results show the remaining sampling time after subtracting the ray tracing time.

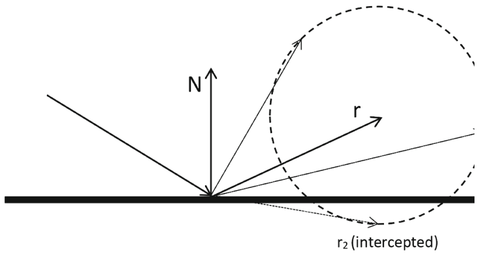

Fig. 9. Offset samples along the normal direction. Some rays are intercepted before we add $N * Rs$.

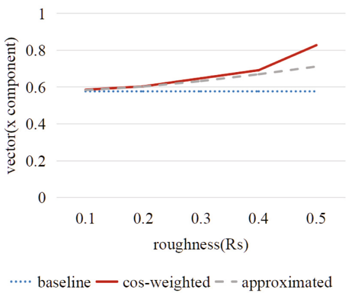

Fig. 10. Adding $N * R_s$ to approximate $\cos\theta$. We measured a ray reflected on surfaces with different roughness, which is acceptable.

Figure 8 shows that our method is three times faster than traditional sampling. With our method, the major time is left on ray-triangle interception. Note that the drawing latency does not equal the ray tracing time plus the sampling time because some operations are possibly parallel on GPU.

In Fig. 7, we measure the performance comparison across different scenes. The result shows that our method gains a steady performance increase across different scenes. Note that as the polygonal faces increase, more time is spent on rasterization or vertex interpolation.

6 Conclusion and Future Work

This paper introduces a novel reflectance model and a sampling method for Phong-like reflection that offers significant acceleration in the sampling process while retaining precision. Our method is ideal for physically based rendering because it doesn't result in energy loss and avoids the 'VNDF' problem. It is also flexible across roughness values from 0 to 1. However, the R_s factor in our model needs further experimentation to better align with reality. Additionally, the BRDF is currently unavailable in closed form, and our approach is currently only available as a fast sampling method.

Overall, our method provides a fast and accurate approach for sampling Phong-like reflections in physically based rendering. It has the potential to improve the efficiency and realism of ray tracing in various lighting scenarios. However, there are still limitations and areas for improvement, such as the fitting of the R_s factor to real-world materials and the extension to anisotropic reflection. Further research and experimentation are needed to fully explore the capabilities and limitations of our method.

References

1. Ashikhmin, M., Ghosh, A.: Simple blurry reflections with environment maps. J. Graph. Tools **7**(4), 3–8 (2002)
2. Baker, J.A.: Integration over spheres and the divergence theorem for balls. Am. Math. Mon. **104**(1), 36–47 (1997)
3. Blinn, J.F.: Models of light reflection for computer synthesized pictures. In: Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques, pp. 192–198 (1977)
4. Duff, T., et al.: Building an orthonormal basis, revisited. J. Comput. Graph. Tech. (JCGT) **6**(1) (2017)
5. Gooch, A., Gooch, B., Shirley, P., Cohen, E.: A non-photorealistic lighting model for automatic technical illustration. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pp. 447–452 (1998)
6. Heitz, E.: Sampling the GGX distribution of visible normals. J. Comput. Graph. Tech. **7**(4), 1–13 (2018)
7. Heitz, E., d'Eon, E.: Importance sampling microfacet-based BSDFs using the distribution of visible normals. In: Computer Graphics Forum, vol. 33, pp. 103–112. Wiley Online Library (2014)
8. Kurt, M.: Real-time shading with Phong BRDF model. J. Sci. Eng. **21**(63), 859–867 (2019)
9. Lafortune, E.P., Willems, Y.D.: Using the modified Phong reflectance model for physically based rendering (1994)
10. McGuire, M., Evangelakos, D., Wilcox, J., Donow, S., Mara, M.: Plausible Blinn-Phong reflection of standard cube MIP-maps. Technical report, Citeseer (2013)
11. Phong, B.T.: Illumination for computer generated pictures. Commun. ACM **18**(6), 311–317 (1975)
12. Torrance, K.E., Sparrow, E.M.: Theory for off-specular reflection from roughened surfaces. Josa **57**(9), 1105–1114 (1967)