偏光を考慮した鏡面反射のリアルタイムレイトレーシング

Real-time ray tracing of specular reflection with polarization

今給黎 隆
Takashi Imagire

東京工芸大学、164-8678　東京都中野区本町2－9－5
2-9-5 Hon-cho, Nakano-Ku, Tokyo 164-8678, Japan

概要

ゲームグラフィックスの世界において、GPUの性能の向上からリアルタイムレイトレーシングが注目を集めている。ゲームグラフィックスの世界では、レイトレーシングにおける光線の反射率に関して、Schlickによる近似を用いたフレネルの式が用いられる場合が多い。しかしながら、レイトレーシングが得意な鏡面反射では偏光の効果が生じやすく、Schlickによる近似との差がレンダリング結果に表れやすいものと考えられる。今回は、偏光を考慮したリアルタイムレンダリングの手法を定式化し、偏光を考慮したレイトレーシングでの高速化手法を検証する。

1. はじめに

Windows 10において、2018年のOctoberアップデートでDirectX Raytracingが導入され、デジタルゲームでのレイトレーシングが身近な存在となってきた。しかしながら、コンピュータグラフィックスの世界では、高速に計算を行うための近似計算が随所に用いられている。頻繁に用いられるものにフレネルの式に対するSchlickの近似がある[1]。Schlickの近似は偏光していない光に対しての近似式である。一方、光の反射には偏光が伴う。偏光に伴う反射率の違いは、PLフィルターや偏光サングラスを用いることで観察できることが知られているが、必ずしもこれらの機器は必要ではない。p波の光は特定の反射角度で反射率が0になるブリュースター角が存在する。ブリュースター角で反射させた光を、そのp波の向きと反射後のs波を直交させ、ブリュースター角での反射をさらに行えば、p波の光の反射率は0になるため、その光は完全に吸収される。このように、偏光の効果はレンダリングにおいて目で見える結果として観測されるはずであり、高次反射を考慮しつつも現状では十分にレイの本数を増やすことができないリアルタイムレイトレーシングの分野では、偏光による効果のレンダリング結果への影響度の調査は重要なテーマとなる。本研究では、リアルタイムレイトレーシングへの偏光の導入を定式化すると共に、レイ間で受け渡すデータを削減することによる偏光処理の高速化を提案し、その効果を検証する。

2. 偏光を伴う反射

光の運動は電磁波としてMaxwell方程式によって記述される。真空中の光が屈折率nの物体の表面で反射される際は、反射後の電場の強度は、反射面に平行な成分（s波）と反射方向及びs波に垂直な成分（p波）に対して、フレネルの式として知られる振幅反射率r_s,r_pの係数が乗算された値となる。

$$r_s = \frac{\cos\theta - n\sqrt{1-\frac{\sin^2\theta}{n^2}}}{\cos\theta + n\sqrt{1-\frac{\sin^2\theta}{n^2}}} = \frac{\cos\theta - \sqrt{n^2-1+\cos^2\theta}}{\cos\theta + \sqrt{n^2-1+\cos^2\theta}},$$

$$r_p = \frac{n\cos\theta - \sqrt{1-\frac{\sin^2\theta}{n^2}}}{n\cos\theta + \sqrt{1-\frac{\sin^2\theta}{n^2}}} = \frac{\cos\theta - \frac{1}{n^2}\sqrt{n^2-1+\cos^2\theta}}{\cos\theta + \frac{1}{n^2}\sqrt{n^2-1+\cos^2\theta}}.$$

ここで、θ は光の入射角及び反射角である。拡散反射や鏡面反射光の相関は切りたい。そのため、電場ではなく、電場の二乗であるエネルギー及び輝度に関して、s波とp波の分解を行う。

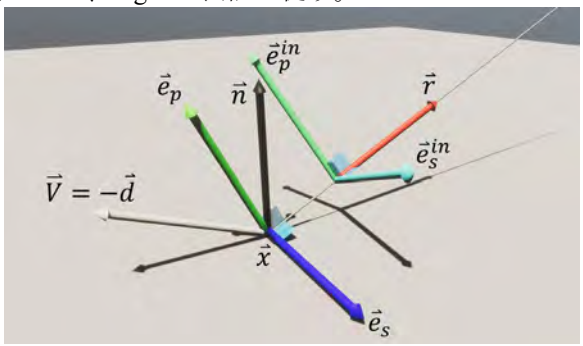
$$\vec{I}_s = (1-r_s^2)\frac{\vec{I}^D}{2} + r_s^2\left(\frac{\vec{I}^S}{2} + \vec{I'}_s\right),$$

$$\vec{I}_p = (1-r_p^2)\frac{\vec{I}^D}{2} + r_p^2\left(\frac{\vec{I}^S}{2} + \vec{I'}_p\right),$$

$$\vec{I'}_s = |\vec{e}_s \cdot \vec{e}_s^{in}|\vec{I}_s^{in} + |\vec{e}_s \cdot \vec{e}_p^{in}|\vec{I}_p^{in},$$

$$I'_p = |\vec{e}_p \cdot \vec{e}_s^{in}|\vec{I}_s^{in} + |\vec{e}_p \cdot \vec{e}_p^{in}|\vec{I}_p^{in}.$$

I_s, I_pはレイが交差した点でのレイの方向への放射輝度であり、I'_s, I'_pは反射方向から入射する輝度、I^D, I^Sはレイの方向への拡散反射光及び鏡面反射光である。その他の向きについては、dを入射方向、rを反射方向として、Fig. 1の表記に従う。



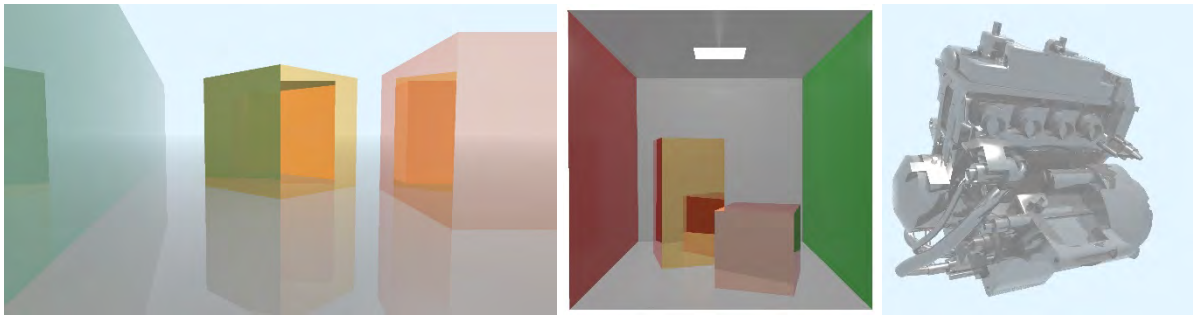


Fig. 2. 提案法による結果。左からガラスの壁や金属の箱が置かれたシーン。コーネルボックス。オートバイのエンジン。

Fig. 1 本手法の反射における記号の可視化

3. 偏光反射におけるペイロードの削減

高次のレイを発生させた際にレイ間で受け渡す情報がペイロードである。今研究では、高速化のためにペイロードのデータを削減する提案を行う。

3.1 s波の余弦の保持

ペイロードには、s波の振動する向きが必要である。この向きは3次元ベクトルであるが自由度を減らすことができる。振動の向きが進行方向に直交することから、自由度は進行方向周りの1次元の角度となる。また、角度に関して360度ではなく、180度の自由度を扱うことができれば良いので、入射角の余弦を保持することで、s波の振動の向きを復元できるようにする。

3.2 s波の係数としてのp波の保持

ペイロードのデータ数を増やしている要因の一つは、s波とp波の2つの色ベクトルを伝搬することにある。p波の大きさをs波のスカラー倍として持つことで、ペイロードの容量を8バイト削減する。

4. 実験

手法の実装と評価を行った。実験に用いたPCはCPUにCore i7-13900KF、メインメモリ32GB、ビデオカードは24GBのメモリを搭載したGeForce RTX 4090を用いた。Fig. 2 において、ガラスの壁や金属の箱が置かれたシーンは36頂点28ポリゴン、コーネルボックスが74頂点38ポリゴン、オートバイのエンジンは10448頂点、12510ポリゴンである。解像度は、壁や箱のあるシーンが、1920×1080、コーネルボックスとエンジンのシーンは、2048×2048である。表1がペイロードの削減手法の違いによる実行速度の変化である。Schlick近似が最も高速にしても、その後は右肩上がりに実行速度が向上することが期待されたが、今回の提案の高速化手法では、残念ながら、大きな高速化を果たすことはできなかった。むしろ各グラフの一番右の無偏光の正確なフレネルの式の方が、Schlickによる近似手法よりも高速に計算することができた。

5. まとめと課題

レイトレーシングにおける偏光の効果の検証と、偏光を考慮したリアルタイムレイトレーシングにおけるペイロードの削減をする手法を提案した。ペイロードを削減した際の実行速度の向上は今回の実験では見られなかった。特にp波をs波のスカラー倍とするペイロードの削減方法の提案では、状況によってはレンダリングされる色に大きな誤りが出ることが判明した。しかしながら、よく使われているSchlickの近似法は、現代的なリアルタイムレイトレーシングの環境では必ずしも高速ではなく、s波とp波の平均の反射率を用いるのが、導入が容易でレンダリング品質を向上させる実用的な手法であることが判明した。今回の結果は、GPUの種類や描画するシーンによって結果が変わる可能性があるため、今後は、より詳細にボトルネックを調査し、効果的な改善手法を提案したい。

6. 参考文献

[1] Schlick, C. “An inexpensive BRDF model for physically based rendering.” In Computer graphics forum, Vol. 13, No. 3, pp. 233-246 (1994).

表1. 各シーン及び手法における実行速度