G-12

Unity を用いた計算機合成ホログラムのための点群生成手法の改良
～HDRP による写実性の向上とレイトレーシングを用いた点群の奥行き計算による反射表現～
Improved Point Cloud Generation for Computer-Generated Holograms Using Unity
～Improvement of photorealistic effect by HDRP and reflection representation by depth calculation of point clouds using ray tracing～

○萱森 元弥 *, 山口 健 †, 吉川 浩 †
Motoya Kayamori*, Takeshi Yamaguchi†, Hiroshi Yoshikawa†

Abstract : Computer-generated hologram (CGH) is hologram in which interference fringe is computed using a computer. We has been researching on CGH and has developed software to generate point cloud data for CGH using Unity. This software was insufficient in expressing the reflection of light because it did not perform the depth shift of the point cloud for expressing the reflection of light. In this reserch, the photorealistic effect is improved by using Unity's HDRP and the reflection is expressed by moving the point cloud using ray tracing.

1. はじめに

ホログラムとは錯覚を利用しない 3 次元画像を干渉縞を利用して記録・再生することが可能な技術である．計算機合成ホログラム（以下 CGH）は干渉縞をコンピュータで計算するため，3DCG データなどの仮想物体をホログラムとして記録することができる．干渉縞は点群データから計算され，当研究室では Unity を用いて点群データを生成するソフトを開発した [1]（以下 U2Dat）．しかし，U2Dat では点群データを生成するときに光の反射を再現するための奥行きの移動を行っておらず，光の反射表現が不十分であることが課題であった．

本研究では点群データを生成する際にレイトレーシングを用いて点群の奥行きの移動を行うことで，光の反射による奥行きの変化を再現する．また，Unity のバージョンを 2021 年のものに変更し，HDRP という最新のレンダリング環境を使用することで，シーンの写実性を向上させる．

2. 原理・方法

2. 1. CGH 作製の流れ

CGH は, ホログラムの記録過程をコンピュータで行ったものである．CGH 作製の手順を以下に示す．

1. CG モデル配置
2. 点群データ生成
3. 干渉縞計算
4. CGH 出力

U2Dat では 1 と 2 の工程を行う．1 では CG モデルを Unity のシーンに配置し，3D シーンの作成を行う．2 では 1 で作成したシーンからカラーバッファとデプスバッファを撮影し，色情報と 3 次元の位置情報を持つ点群データを生成する．3 では CGH を観察するために必要な干渉縞の計算を行う．4 では 3 で計算した干渉縞を感光材に記録する．本研究では，2 の点群データ生成の工程でレイトレーシングを導入し，デプスバッファからは取得できない光の反射があった時の視点とオブジェクトの距離を取得し点群の移動を行う．

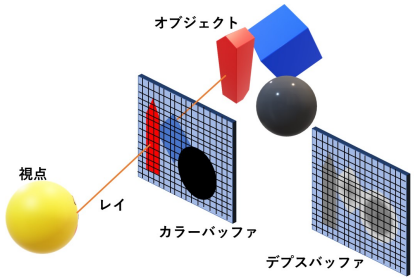


Figure 1: Raytracing

2. 2. HDRP とレイトレーシング

従来のソフトは，Unity の 2018 年バージョンのビルトインレンダーパイプラインと呼ばれる環境を用いて開発していた．本研究では，Unity の 2019 年バージョンから使用できる HDRP と呼ばれる最新のレンダリング環境を使用する．HDRP ではレイトレーシングを使用することができ，3D シーン内の光の透過・屈折・反射表現の写実性の向上が見込まれる．

レイトレーシングでは，Fig. 1 のようにレイと最初に交差するオブジェクトのポリゴンの色を取得し，スクリーンの画素を塗ることでカラーバッファを生成している．またデプスバッファは従来の手法では，スクリーンに一番近いポリゴンの位置に奥行きが設定されていたため，鏡の中の奥行きを再現することができなかった．

2. 3. レイトレーシングを用いた反射表現の改良

本研究では，RayCast と呼ばれるカラーバッファを作成するものとは別のレイトレーシング機能を用いて，Fig. 2 のように視点位置からレイを飛ばして鏡で反射させ，反射した先の物体までの距離を取得して奥行き情報とする．RayCast はレイをまっすぐ飛ばすものであり，自動で反射することはない．本研究では，鏡面反射する

*日大理工・学部・応用情報　†日大理工・教員・応用情報

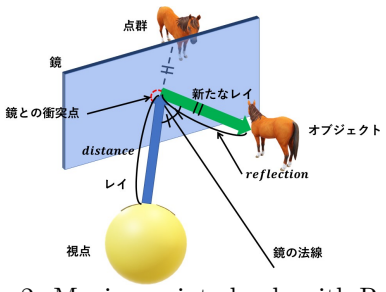


Figure 2: Moving point clouds with RayCast

オブジェクトにタグをつけ，タグが付いたオブジェクトにレイが衝突したときにレイのベクトルと鏡面の法線ベクトルをそれぞれ取得する．その後 2 つのベクトルから反射ベクトルを計算し，その方向に新たなレイを飛ばす．

また，RayCast はレイが物体と衝突したとき，レイの始点から衝突した位置までの距離を取得することができる．本研究では，物体と衝突したときの距離 D をレイから取得し，鏡の反射があった場合はその都度距離 R_n を新たなレイから取得して足し合わせることで点群データの奥行き Z とする（Eq. (1)）．

$$Z = D + R_1 + R_2 + \cdots \tag{1}$$

3. 結果

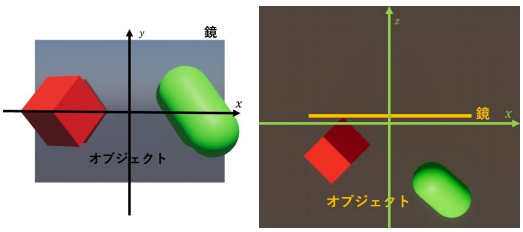
点群の奥行きの移動をを行うことで鏡の中の奥行きを再現し，再生像シミュレーションにかけてからレインボウホログラム（以下 CGRH）を撮影した．今回撮影を行った CGRH は解像度 288,000 × 216,000，画素ピッチ 0.35 × 0.35 [µm]，視点距離 0.5 m である．また使用したシーンの正面と俯瞰視点の画像を Fig. 3，に，点光源を Fig. 4 に，再生像シミュレーションを Fig. 5 に，CGRH を Fig. 6 にそれぞれ示す．

Fig. 4 を見ると，(a) では鏡に映った像は鏡の位置に奥行きが設定されているが，(b) では鏡に映った像の奥行きが鏡の後ろに移動している．また，Fig. 5 はホログラムのシミュレーション，Fig. 6 は実際に撮影した CGRH であるが，点群を移動させたことによる視差が発生した．一方，奥行きを移動させたことによって焦点調節ができるようになっているはずであるが，今回は焦点調節による違いがあまりわからなかった．

4. むすび

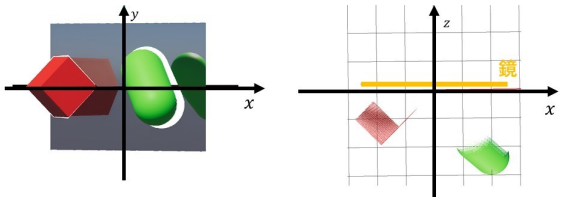
光の反射表現のために RayCast を用いて点群を移動させるスクリプトを作成し，実際に点群データを生成し再生像シミュレーション，CGRH の撮影を行うことで奥行きを移動させたことによる視差が発生したことを確認し，より写実的な CGRH の生成ができたことを確認した．

今後は光のふるまいとして反射と並んで重要な屈折に関して，ガラス玉などの半透明物体を光が通過することによる点群の移動を RayCast を用いて行う．最終的には光の透過・屈折・反射の表現が必要なシーンを作成し，点群の移動を行うことでより写実的な CGRH の生成を実現することが目標である．

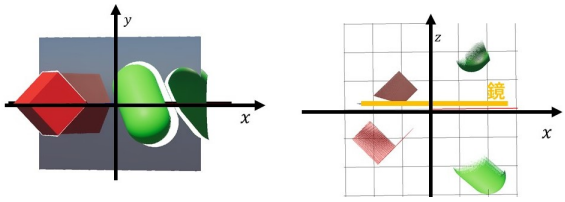


(a) Front　(b) Overlooking

Figure 3: Scene

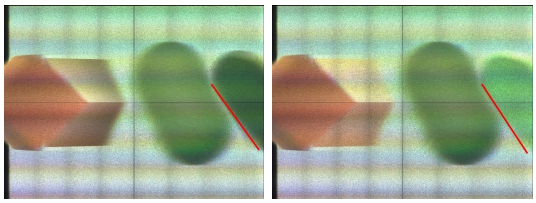


(a) Conventional methods



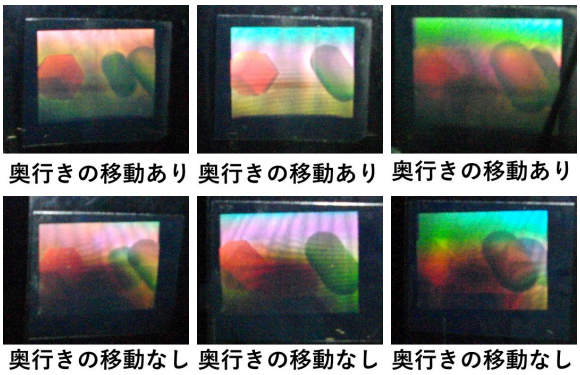
(b) RayCast-based methods

Figure 4: Point cloud



(a) With depth shift　(b) Without depth shift

Figure 5: simulation(x=-300 mm)



(a) x=-100 mm　(b) x=0 mm　(c) x=100 mm

Figure 6: CGRH

参考文献

[1] 湯浅, 山口, 吉川：“Unity を用いた CGH レンダリングソフトウェアの開発”, 3 次元画像コンファレンス (2019).