第 27 回日本バーチャルリアリティ学会大会論文集（2022 年 9 月）

ゲームエンジンを用いた再帰透過光学素子における迷光の位置・変形の再現

林 竜吾 [1)]，齋藤 旭 [1)]，小泉 直也 [1)]
Ryugo HAYASHI, Asahi SAITO, and Naoya KOIZUMI

1) 電気通信大学（〒 182-8585 東京都調布市調布ヶ丘 1-5-1, hayashi, asahi@media.lab.uec.ac.jp, koizumi.naoya@uec.ac.jp）

概要: 本研究では、空中像を観察する視点と空中像に対する迷光の位置・変形の関係を調査し、ゲームエンジン内でそれを再現した。再帰透過光学素子を用いた空中像装置で発生する迷光の位置・変形の把握を目的として、空中像と迷光のアフィン変換行列をシミュレーションデータから求めた。得られた変換行列を用いてゲームエンジンで迷光と空中像を描画し、実物と比較してこの妥当性を確認した。

キーワード： 空中像光学素子，迷光，CG

1. はじめに

再帰透過光学素子は光源からの光を光学素子に対して面対称な位置に結像させる光学素子である。再帰透過光学素子はその特性から、主に空中像を生成するために用いられる。再帰透過光学素子は、結像する空中像が再帰性反射材を用いた空中像よりボケが少ないという利点や、他の光学素子と組み合わせずに空中像を生成できるという利点がある。

再帰透過光学素子の主な課題として、迷光を生じてしまうことが挙げられる（図 1）。迷光は空中像付近に発生し、空中像と重なる場合がある。このとき迷光はユーザーの空中像への観察や集中を妨げてしまう。そのため、ユーザーが空中像に集中できるように装置を設計するためには、迷光の位置・変形の把握が重要である。迷光の位置・変形の把握のためには、空中像装置を何度も検討し組み立てたり、複雑な計算を行ったりする必要がある。これらの作業には多くの労力を要するため、困難である。そこで、迷光がどのような位置にどのような形で発生するかをあらかじめ把握することができれば、空中像装置の設計時に迷光を考慮することができ、ユーザーが空中像に集中できる装置の検討が可能となる。

我々は空中像体験のプロトタイプをする手法として、ゲームエンジンを用いて空中像光学素子を再現し、VR 空間内で空中像体験を設計することを提案する。まず本稿では一般的なゲームエンジンを用いて迷光の位置・変形をシミュレーションし、実際の再帰透過光学素子をにより生じる迷光と比較することとした。

2. 関連研究

2.1 再帰透過光学素子

再帰透過光学素子には Micro Mirror Array Plates (MMAP) や Dihedral Corner Reflector Array (DCRA)[1]、Radially arranged DCRA[5] が挙げられる。本稿では後述するシミュレーションを用いるため MMAP を対象とした。

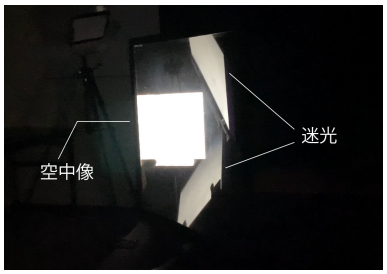

図 1: MMAP を用いた光学系で迷光が生じている様子

MMAP は、細い鏡が並べられた Slit Mirror Array (SMA) の構造になっている層が直交するように重ねられた光学素子である。MMAP 内で 2 回反射した光が MMAP に対して面対称な位置に結像する。このとき光は 2 つの SMA で奇数回ずつ反射する。

再帰透過光学素子を用いた空中像装置では迷光が生じる。2 つの SMA のどちらかまたは両方で偶数回反射した光が迷光となる。迷光の位置や形は視点や光源の位置関係、光源の発する光の特性によって変化する。

2.2 シミュレーション

空中像と迷光のシミュレーションとして、いくつかの手法が提案されている。

MMAP における空中像や迷光の位置をヒートマップで表現した例がある [2]。この手法では、ヒートマップで表現することで空中像や迷光の位置を調査している。しかし、空中像や迷光がユーザーからどのように見えるかはこのシミュレーションから再現できない。

Kiuchi ら [3] はレイトレーシング法を用いた MMAP のシミュレーション手法を提案している。レイトレーシング法は光線を追跡することでレンダリングを行う方法である。この手法では、MMAP を正確にモデリングし、物理現象に従って光線を追跡することで、空中像や迷光の見え方を現実と同様に再現できる。また、Kiuchi らは迷光の形が空中

像を平行移動およびせん断した形であることを確認し、これらの両方が行えるアフィン変換で迷光の変形を計算できるはずだと説明している。

星ら [4] はレイトレーシング法を用いた MMAP のシミュレーションと画像処理を組み合わせて、空中像が迷光なしに観察できる範囲を調査している。迷光を生成する光と空中像を生成する光の、MMAP 内での反射回数が異なる点に注目している。反射回数ごとに光線を分けてレンダリングし、差分を取ることで迷光を自動で検出する手法を提案している。

しかし、これらの手法にはリアルタイムな描画が難しい、迷光の変形のシミュレーションが困難といった課題がある。CG 環境で空中像とのインタラクションを行うためにはリアルタイムな視点移動に対応して迷光の位置・変形が再現できなければならない。また、CG 環境で現実と同様に空中像装置を検証するには、現実と同様の見え方で迷光を再現する必要がある。レイトレーシング法を用いたシミュレーションは、現実と同様に迷光の位置・変形が再現できる一方、計算量が多くレンダリングに時間がかかる。そのため、視点移動に対してリアルタイムに対応したシミュレーションを行うことが難しい。

本稿ではアフィン変換行列を用いて迷光の位置・変形を再現し、VR 空間内で空中像体験を設計することを提案する。変換行列を用いて迷光の位置・変形を計算することで、迷光の位置・変形の見え方をリアルタイムに再現し、CG 環境でインタラクション可能なシミュレーションを可能にする。

3. 提案手法

レイトレーシング法による従来のシミュレーション手法を用いて、迷光の位置・変形を計測した。計測結果から迷光の位置・変形の式を定めた。また、Unity で提案手法を実装した。

3.1 迷光の描画方針

本稿では迷光の空中像に対する変形を計測し、アフィン変換を用いることで迷光の位置・変形を再現する。迷光の描画方法として、上下の迷光の各頂点の三次元座標と形状を全て測定し同様な形状のオブジェクトを配置する方法や、迷光の空中像に対する変形を計測しシェーダーで変換行列を用いる方法が挙げられる。迷光の各頂点の三次元座標や形状をすべて計測するのには多くの労力を要する。そのため、変換行列を用いることにする。

迷光の位置・変形をシミュレーションするために Unity 2020.3.28f で作成する光学系を図 2 に示す。MMAP に対して Display と面対称な位置に空中像が結像する。実際は空中像の位置にはオブジェクトを配置しない。Virtual Camera の Display に対する位置・向きが、視点の空中像に対する位置・向きと合同になるように Virtual Camera を配置する。これにより、Virtual Camera の出力は視点からの空中像の見え方を撮影しているのと同様になる。また、MMAP と同じ位置に空中像や迷光を表現するための Quad を配置する。

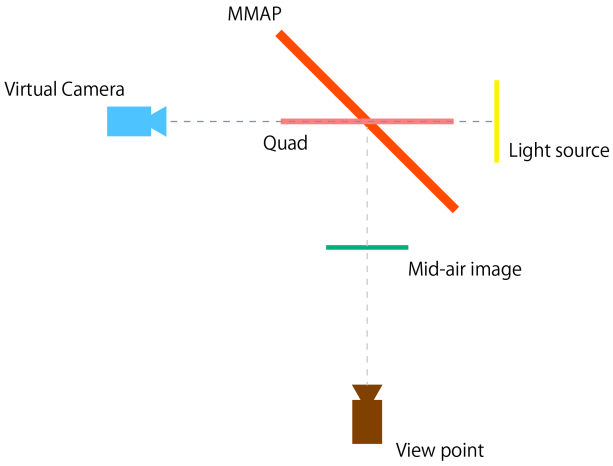

図 2: Unity での構成

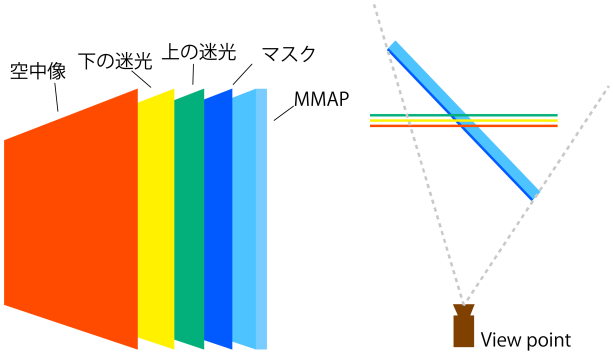

図 3: MMAP 部分: (a) 構成; (b) 視域

図 2 の配置では図 1 のように、空中像の右上と右下に迷光が生じる。このとき、右上に生じる迷光を上の迷光、右下に生じる迷光を下の迷光と呼ぶことにした。上の迷光と下の迷光では空中像に対する変形が異なるため、それぞれ異なる行列を用いる。

MMAP は、上の迷光を表示する部分、下の迷光を表示する部分、空中像を表示する部分、SMA とガラスの部分に分けて再現する。上下の迷光を表示する部分と空中像を表示する部分はそれぞれ Quad を用いる。SMA とガラスの部分は Cube を板状に配置して表現する。

Virtual Camera の出力になっているレンダーテクスチャをシェーダーでアフィン変換し、Quad のテクスチャにすることで空中像および迷光の位置・変形を再現する。空中像と上の迷光、下の迷光はそれぞれ異なるシェーダーを用いて描画する。空中像はレンダーテクスチャをそのまま用いて表現する。上の迷光と下の迷光はそれぞれ異なる変換行列によりアフィン変換を行うことで位置・変形を表現する。空中像・迷光が正しい向きで観察できるようにレンダーテクスチャを貼り付ける Quad の法線が視点方向を向くようにする。迷光を表現する Quad は MMAP を表現する Cube の 1.2 倍の大きさにする。

空中像や迷光などが描画されない範囲は透明になるように、シェーダーでクロマキーを実装する。Unity のレイヤー機能を用いて Virtual Camera は Display のみを撮影する

ように設定する。また、Virtual Camera の背景は任意の単色に設定する。空中像と上下の迷光を描画するシェーダーにおいて Virtual Camera の背景に用いた色でクロマキーを行うようにすることで、Quad の空中像や迷光以外の部分は透明になるようにする。Virtual Camera の画角は視点と MMAP の距離を d として、$2\arctan(48{:}8 \times 1{:}2{=}2d)$ に設定した。

MMAP の表面にマスクをかけ、視点から見てマスクと重なっている領域だけ空中像と迷光を描画するようにシェーダーを作成することで視域を再現する。現実の空中像装置において、空中像や迷光は視点と MMAP の間にのみ生成される。この視域の制限を再現するために、MMAP の表面にマスクとなる Quad を配置する。視点から見て空中像や迷光のマスクと重なる部分だけが描画されるようにシェーダーを設定する。これにより現実と同様の視域が再現できる。

3.2 迷光の計測

Blender 内で MMAP と光源を配置し視点となるカメラの位置を変化させ、各条件でレンダリングを行った。Blender 3.0 を利用し、MMAP には Kiuchi ら [3] が作成した MMAP のモデルを用いた。レンダリングには Blender に組み込まれている Cycles を用いた。このモデルによるシミュレーションでは空中像・迷光の位置や形が実際の光学系と同様に観察できる。実際の光学系での計測と比べて、このモデルによるシミュレーションを用いた計測は、光学系のパラメータの変更や細かい間隔での計測が容易である、計測誤差が少なくなる、といった利点がある。

計測の概要を図 4 に示す。現実での 1 cm を Blender での距離 1 とした。MMAP は高さおよび幅を $H_m = 97{:}6$ cm とした。その他のピッチ幅などの MMAP のパラメータはデフォルトの設定のままにした。MMAP–光源間距離 L_d は 15 cm とした。光源は拡散稿を出射する半径 0.75 cm の円盤で、円の中心間距離が 8 cm となるように同一平面上に正方形に配置した。このとき、4 つの円盤を含む平面が YZ 平面に平行になるように配置した。カメラは空中像から距離 150 cm かつ、XZ 平面に平行な平面上を移動させた。カメラが移動する平面の、空中像の正面を原点として、z = 0 の条件で x 座標 −90 cm から 90 cm まで 2 cm ごとに 91 点、x=0 の条件で z 座標 −90 cm から 90 cm まで 2cm ごとに 91 点、以上の 2 条件で計 182 点の視点からレンダリングを行った。この範囲は、空中像に対する視線の方位角および仰角の、約 −0:54 rad–約 0.54 rad の範囲に相当する。レンダリング時に、カメラは空中像として結像される 4 つの円からなる正方形の中心を常に向くように設定した。MMAP とカメラの距離を d として、カメラの画角は $2\arctan(48{:}8 \times 1{:}2{=}2d)$ としてレンダリングした。

各条件において視点–空中像間の距離が一定でない。しかし、視点–空中像間の距離は迷光の位置・変形に影響しないため、これらは方位角および仰角方向の視点移動とみなせる。

各条件でレンダリングした画像から空中像に対する迷光の位置・変形を 2 × 3 のアフィン変換の変換行列として計

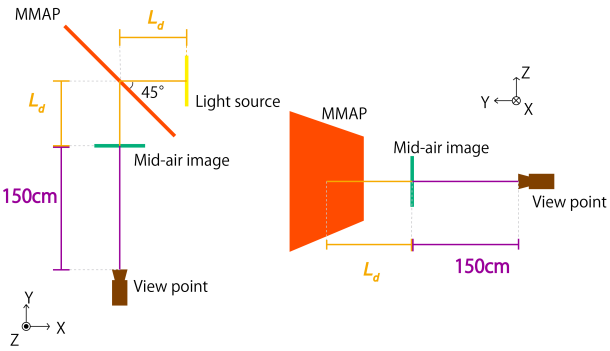

図 4: 測定概要図: (a) 俯瞰して見た様子; (b) 横から見た様子

測した。OpenCV の connectedComponentsWithStats 関数を用いて空中像および迷光となった 4 つの円形の像の重心を求めた。空中像・迷光の 4 つある重心位置のうち 3 つを用いて、getAffineTransform 関数で空中像に対する迷光の変形をアフィン変換の変換行列として求めた。これらの操作を上の迷光と下の迷光のそれぞれについて行った。

3.3 変換行列の定式化

計測した変換行列から迷光の位置・変形の式を定めた。視点の移動に対する上の迷光の変換行列の変化を行列の要素ごとに観察し、曲線のフィッティングを行った。2 × 3 のアフィン変換行列のうち、i 行 j 列目の要素を要素 $3(i-1)+j$ と定めた。

z = 0 の条件で x 座標 −90 cm から 90 cm でレンダリングしたそれぞれの画像から得た、各視点における変換行列の 6 つの各要素について、縦軸を要素の値、横軸を空中像に向けた視線の方位角としてグラフにプロットした。

プロットした点に対して曲線をフィッティングすることで、変換行列の要素ごとに方位角　方向の視点移動に対する値の変化の式 $f(\ )$ を求めた。要素 n の式を $f_n(\ )$ とすると、各要素の式は以下のように求められた。

$$f_1(\ ) = 0{:}2053\ ^2 - 0{:}2499\ \ + 0{:}5505 \tag{1}$$

$$f_2(\ ) = -0{:}0142\ ^2 + 0{:}0022\ \ + 0{:}0314 \tag{2}$$

$$f_3(\ ) = -387{:}8775\ ^2 + 483{:}2539\ \ + 720{:}035 \tag{3}$$

$$f_4(\ ) = 2{:}1485\ ^3 + 1{:}9065\ ^2 + 1{:}1994\ \ + 0{:}6674 \tag{4}$$

$$f_5(\ ) = 0{:}016\ ^2 + 0{:}0154\ \ + 0{:}9777 \tag{5}$$

$$f_6(\ ) = -3148{:}12\ ^3 - 2926{:}41\ ^2 - 2058{:}05\ \ - 1080{:}5 \tag{6}$$

アフィン変換行列の各要素の値について、グラフにプロットし上記の曲線をフィッティングした結果を図 5 に示す。方位角方向の視点移動における x 座標 −90 cm から −88cm、−42 cm から −32cm、62 cm から 90 cm の範囲のレンダリング結果では、getAffineTransform 関数に必要な数の頂点座標が計測できなかった。また、仰角方向の視点移動について、上の迷光については z 座標 74 cm から 90cm、下の迷光については −90 cm から −74 cm の範囲のレンダリング結果で、getAffineTransform 関数に必要な数の頂点座標