Vol.2017-CVIM-207 No.23
2017/5/10

大域照明の内挿による光源操作可能なARシステム

井手 絢香[1,a)]　間下 以大[1,2]　清川 清[3]　浦西 友樹[1,2]　Photchara Ratsamee[1,2]　竹村 治雄[1,2]

概要：近年，拡張現実感（Augmented Reality, AR）が様々な分野で応用されている．応用の一つに現実やバーチャルのオブジェクトを編集して見せられることを利用した部屋の模様替えシミュレーションなどのシステムがある．しかし，このようなシステムは，家具など現実でも動かすことの容易なオブジェクトに関するものが多い．対して本研究では，照明や窓など部屋の光源となるものを移動させることのできるARシステムを提案する．このようなARシステムを実現し実環境で動かすためには，光学的整合性とリアルタイム性を両立させる必要がある．そこで，本研究ではRGB-Dカメラを用いて部屋の3次元形状の復元を行い，光源となる物体を移動させたバーチャル空間での大域照明を数パターン事前計算した後，ユーザがARシステムを用いて光源を動かした場合の大域照明を事前計算の内挿によって求める手法を提案する．さらにその結果を現実映像に重畳することで，リアルタイム性と光学的整合性を両立したAR空間における光源編集を実現する．実験ではAR空間中で光源を移動させ，移動後の位置から照明を当てたような写実的な陰影がリアルタイムに近い早さで表現されていることを確認した．

キーワード：光学的整合性，拡張現実感，大域照明

1. はじめに

近年，拡張現実感（Augmented Reality, AR）を用いた技術が様々な分野で注目されている．AR技術の応用の一つとして，現実のオブジェクトやバーチャルなオブジェクトを現実にあるように編集して見せるものがある．その技術を利用したスマートフォン用アプリケーション“IKEAカタログ”[1]のような，AR家具配置システムが実用化されている．このようなシステムではARを利用することで現実の部屋に居ながらにして家具の配置を考えることができる．これらのシステムでは現実でも比較的動かしやすい家具などのオブジェクトに関するものが多い．一方，窓や壁紙，一部の照明などは物理的に動かしたり変更したりすることが難しく，動かすために多大な手間と費用が必要となる．こういった物をAR環境中で動かすことができればARの適用範囲を大幅に拡大することができる．そこで，本研究では，現実では容易に動かしにくい照明や窓といった光源となるものをリアルタイムに編集して見せることを目的とする．AR環境で光源を動かすことにおいて問題となるのが光学的整合性と計算時間の2点である．3D空間上の正確な光の状況をシミュレーションするには膨大な時間が必要となる．また，現実環境における反射率や表面のテクスチャを考慮することは非常に難しい．そこで，本研究では，AR環境において基本的には現実世界の映像を用いつつ，事前計算結果から光の成分のみ処理を加えることでリアルタイムに光源位置の編集を行うシステムを提案する．

2. 関連研究

2.1 AR環境でのオブジェクト編集

AR空間におけるオブジェクトの編集の研究やシステムは多く存在し，実用化されているものもある．ARを利用した家具編集システムの実用化の一例として，IKEAカタログが挙げられる[1]．このシステムでは，スマートフォンのカメラを通して実際に販売している家具を実世界に重畳表示させ，実際の間取りや大きさを考慮した家具配置を考えることが可能である．他にもAR空間におけるオブジェクト編集を行う研究としては，複数のマーカを利用して家具の配置や色の変更などのインタラクションを行えるPhanらのAR家具配置システム[2]や，坂内ら[3]の家具の3次元モデルを配置したり，現実に存在する家具を除去して見せたりすることができるシステム，AR空間上にあるオブジェクトのテクスチャを他のオブジェクトへ転写したり，オブジェクトの移動やコピーを行える森らのシステム[4]などがある．また，比較的現実で動かしにくいオブ

1 大阪大学情報科学研究科
2 大阪大学サイバーメディアセンター
3 奈良先端科学技術大学院大学
a) ide@am.sanken.osaka-u.ac.jp

ⓒ 2017 Information Processing Society of Japan

ジェクト編集の例としては，上田らの手持ちカメラで取得したフリーハンド映像上で，CG の壁紙を重畳表示することによる壁紙の変更シミュレーションシステム [5] などもある．

2.2 AR 環境での光学的整合性

AR 環境における光学的整合性に関する研究として，実環境の光源を推定してバーチャル物体に適用するような研究は多く行われている．Gruber ら [6] の手法では，リアルタイムに AR 空間における光源の推定とバーチャル物体と実環境の物体に対するキャストシャドウの計算を行い，見せることができる．Jachnik ら [7] の手法では，単眼カメラで平面を複数視点から撮影することにより取得した密な光源情報を鏡面成分と拡散成分に分離し、鏡面成分を環境マップの推定に利用する．安原ら [8] の手法では，平面を複数視点から撮影した画像の輝度値をもとにして光源方向を推定し，実画像と合成画像を比較するインバースレンダリングによって光源環境と物体の反射特性を推定する．しかしこれらの研究はどれも，現実の光源を直接編集しようとしたものではない．

2.3 光源変化画像の高速生成に関する研究

AR 空間でなく CG 上での表現において，光学的整合性が取れた画像を描画する方法は多く研究されている [9]．現実世界には光源からの光が直接及ぼす影響だけでなく，光源から出た光が他のオブジェクトに複数回反射，屈折してから到達する，間接照明と呼ばれる現象が存在し，間接照明を考慮してレンダリングすることで映像のリアリティは大幅に高められる．間接照明まで考慮して物体表面上の光の挙動をモデル化すること大域照明（グローバルイルミネーション，Global illumination, GI とも言う）と呼ぶ．その実現方法として，主なものにレイトレーシング法とラジオシティ法，フォトンマッピング法などがある．それらの手法を用いると，光学的整合性の取れた画像の生成は可能だが，どれも正確さを求めると非常に計算時間が膨大になる．そこで，より高速に光学的整合性の取れた CG 画像を生成する手法も研究されている．例としては複数の光源位置において事前計算された画像や照度分布から等照度線を求め，等照度線同士の対応付けを行う眞鍋ら [10] の手法や，近年のゲームにおけるグラフィック表現において，リアルさとゲームプレイの両立のために多く使われているライトマップと呼ばれる手法がある [11].

光学的整合性の取れた画像という点では，正確な光の反射・散乱の計算も必須である．それぞれの特性の計測を行う研究も多く行われているが，特殊な機材が必要である方法 [12] や形状が既知であれば比較的手軽に計測できるが不均一な反射特性を持つ物体には適用できない方法 [13] など，それぞれに制約があり，容易に行えるとは言い難い．[14]．このように，現実の物質の反射特性の計測は非常に難しいと言える．

本研究では，現実世界では容易に編集することのできない，照明など光源となるものを編集でき，大域照明も考慮した光学的整合性の取れた画像をリアルタイムに生成できる AR 環境での環境編集システムを提案する．具体的には，AR 環境において基本的には現実世界の映像を用いつつ，事前計算結果から光の成分のみ処理を加えることでリアルタイムに光源位置の編集を行うシステムを提案する．

3. 大域照明の内挿による実時間グローバルイルミネーション

3.1 光学的整合性とリアルタイム性の両立についての問題

写実的な AR システムでは，物体や光源の位置などを AR 空間中で変更した後に正しい陰影関係を持ったシーンの描画が必要になる．このような問題を光学的整合性問題と呼ぶ．AR シーン全体の光の編集を行うためには対象とする現実環境を再現する必要があり，CG において写実的なシーンの描画はレンダリング方程式（式 1）に基づいて計算される．レンダリング方程式では物体の表面におけるエネルギー保存を表しており，物体表面上のある点 $\boldsymbol{x}$ から $\vec{\boldsymbol{\omega}}$ 方向に出射される光のエネルギー L_o はその点から $\vec{\boldsymbol{\omega}}$ 方向に放射される光エネルギー L_e とその点で $\vec{\boldsymbol{\omega}}$ 方向に反射された光エネルギー L_r の和である，ということを示している．この式はある点 $\boldsymbol{x}$ に入射した光エネルギー L_i と BRDF を表す式 $f_r(\boldsymbol{x}, \vec{\boldsymbol{\omega'}}, \vec{\boldsymbol{\omega}})$，法線ベクトル $\vec{\boldsymbol{n}}$ を用いて式 2 のようにも書き換えることができる．

$$L_o(\boldsymbol{x}, \vec{\boldsymbol{\omega}}) = L_e\ (\boldsymbol{x}, \vec{\boldsymbol{\omega}}) + L_r(\boldsymbol{x}, \vec{\boldsymbol{\omega}}) \tag{1}$$

$$\begin{aligned} L_o(\boldsymbol{x}, \vec{\boldsymbol{\omega}}) = &\ L_e\ (\boldsymbol{x}, \vec{\boldsymbol{\omega}}) \\ &+ \int_{\Omega} f_r(\boldsymbol{x}, \vec{\boldsymbol{\omega'}}, \vec{\boldsymbol{\omega}}) L_i(\boldsymbol{x}, \vec{\boldsymbol{\omega'}})(\vec{\boldsymbol{\omega'}} \cdot \vec{\boldsymbol{n}}) d\vec{\boldsymbol{\omega'}} \end{aligned} \tag{2}$$

本研究では，レンダリング方程式を実時間で計算するため以下のように仮定する．

- $\boldsymbol{x}$: 現実環境の形状取得
  形状は既知とし，シミュレーション対象の 3 次元復元モデルは正確であり幾何学的整合性が得られるとする．
- f_r: 現実環境の反射率・BRDF 取得
  現実環境においてこれらを計測することは現実的でないので，計算対象は全て同じ反射率の一様なランバート面，つまりすべての方向に同じ強さで反射する面であると仮定する．また，物体の色や材質感などは現実の映像を利用することで表現する．
- L_i: 現実の光線の再現
  現実の光源位置と強さは既知かつ白色光源とする．大域照明の計算は時間がかかるため，光の計算は直接光成分と間接光成分に分離して行う．直接光成分は計算

時間が短いが，正確に描画されないと違和感のある画像になる．一方，間接光成分は正確に計算すると時間がかかるが，多少不正確であってもユーザに与える違和感は少ないと考えられる．このことから，直接光成分はその場で計算を行い，間接光成分は光源の移動する可能性のある範囲内で何か所分か事前計算を行い，その結果を用いた内挿によって高速に近似値を得る．

これらの仮定を用いて，式 2 のレンダリング方程式を変形する．対象が一様なランバート面であると仮定すると，f_r は入射光 $\vec{\omega'}$ と出射光 $\vec{\omega}$ に対して独立になるため式 3 となる．

$$L_o(\boldsymbol{x},\vec{\omega}) = f_r(\boldsymbol{x})\int_{\Omega} L_i(\boldsymbol{x},\vec{\omega'})(\vec{\omega'}\cdot\vec{\boldsymbol{n}})d\vec{\omega'} \tag{3}$$

さらに，L_i の仮定より L_i を計算時間の短い直接光成分 L_{iDir} と長い計算時間が必要な間接光成分 L_{iIndir} に分離する．また，既知の光源以外からは光が放射されることはないとすると，L_e の項は無視でき，最終的に本手法におけるレンダリング方程式は式 4 のように表される．

$$\begin{aligned} L_o(\boldsymbol{x},\vec{\omega}) = f_r &\int_{\Omega} L_{iDir}(\boldsymbol{x},\vec{\omega'})(\vec{\omega'}\cdot\vec{\boldsymbol{n}})d\vec{\omega'} \\ +f_r &\int_{\Omega} L_{iIndir}(\boldsymbol{x},\vec{\omega'})(\vec{\omega'}\cdot\vec{\boldsymbol{n}})d\vec{\omega'} \end{aligned} \tag{4}$$

この式に基づいて，間接光成分を事前計算結果を用いた内挿で計算することにより光学的整合性とリアルタイム性の両立を図る．

3.2 光のシミュレーション

前節で導かれたレンダリング方程式により，光源座標が $\boldsymbol{l}$ の場合の L_{iDir} の積分計算結果として得られる直接光成分を $L_{Dir}(\boldsymbol{l})$，L_{iIndir} の積分計算結果として得られる間接光成分を $L_{Indir}(\boldsymbol{l})$ とするとカメラで観測される画像 $I(\boldsymbol{l})$ は式 5 のように表せる．

$$I(\boldsymbol{l}) = f_r L_{Dir}(\boldsymbol{l}) + f_r L_{Indir}(\boldsymbol{l}) \tag{5}$$

異なる光源位置 $\boldsymbol{l'}$ の実画像 $I(\boldsymbol{l'})$ についても同様に考え，直接光成分を $L_{Dir}(\boldsymbol{l'})$，間接光成分を $L_{Indir}(\boldsymbol{l'})$ とすると式 6 のように表せる．

$$I(\boldsymbol{l'}) = f_r L_{Dir}(\boldsymbol{l'}) + f_r L_{Indir}(\boldsymbol{l'}) \tag{6}$$

式 7 のように式 6 を式 5 で割ると，f_r が定数であることより光の成分だけの比率を表す係数 $\lambda(\boldsymbol{l},\boldsymbol{l'})$ が得られる．

$$\begin{aligned} \frac{I(\boldsymbol{l'})}{I(\boldsymbol{l})} &= \frac{f_r L_{Dir}(\boldsymbol{l'}) + f_r L_{Indir}(\boldsymbol{l'})}{f_r L_{Dir}(\boldsymbol{l}) + f_r L_{Indir}(\boldsymbol{l})} \\ &= \frac{f_r(L_{Dir}(\boldsymbol{l'}) + L_{Indir}(\boldsymbol{l'}))}{f_r(L_{Dir}(\boldsymbol{l}) + L_{Indir}(\boldsymbol{l}))} \\ &= \frac{L_{Dir}(\boldsymbol{l'}) + L_{Indir}(\boldsymbol{l'})}{L_{Dir}(\boldsymbol{l}) + L_{Indir}(\boldsymbol{l})} \\ &= \lambda(\boldsymbol{l},\boldsymbol{l'}) \end{aligned} \tag{7}$$

つまり，得られた係数 $\lambda(\boldsymbol{l},\boldsymbol{l'})$ を実画像 $I(\boldsymbol{l})$ に掛けることで式 8 のように光源を $\boldsymbol{l}$ から $\boldsymbol{l'}$ に移動させた画像が得られるとわかる．

$$I(\boldsymbol{l'}) = \lambda(\boldsymbol{l},\boldsymbol{l'}) * I(\boldsymbol{l}) \tag{8}$$

本システムでは実画像から光源を編集した結果の画像を得たいので，係数 $\lambda(\boldsymbol{l},\boldsymbol{l'})$ が得られればよい．そこで，実環境と光源編集後の環境についてシミュレーションを行い，それぞれのシミュレーション結果の画像 $\tilde{I}(\boldsymbol{l})$，$\tilde{I}(\boldsymbol{l'})$ から式 9 のように λ を近似した係数 $\tilde{\lambda}(\boldsymbol{l},\boldsymbol{l'})$ を求める．

$$\lambda(\boldsymbol{l},\boldsymbol{l'}) = \frac{I(\boldsymbol{l'})}{I(\boldsymbol{l})} \approx \frac{\tilde{I}(\boldsymbol{l'})}{\tilde{I}(\boldsymbol{l})} = \tilde{\lambda}(\boldsymbol{l},\boldsymbol{l'}) \tag{9}$$

光源編集後のシミュレーションは間接光成分と直接光成分に分け，直接光は毎フレーム計算し，間接光は内挿で計算を行う．シミュレーションで用いる反射率をここでは $\tilde{f}_r$ とする．間接光成分シミュレーションでは，光源が動いた場合のシミュレーションを光源位置 $\boldsymbol{l_i}$（i は自然数）についてあらかじめ行っておき，現在のカメラ視点の画像 $\tilde{I}(\boldsymbol{l_i})$ を得て，そこから式 10 のように直接光成分 $\tilde{L}_{Dir}(\boldsymbol{l_i})$ を引き，間接光成分 $\tilde{L}_{Indir}(\boldsymbol{l_i})$ のみを残したものを作成する．

$$\tilde{f}_r\tilde{L}_{Indir}(\boldsymbol{l_i}) = \tilde{I}(\boldsymbol{l_i}) - \tilde{f}_r\tilde{L}_{Dir}(\boldsymbol{l_i}) \tag{10}$$

そして，間接光成分 $\tilde{L}_{Indir}(\boldsymbol{l_i})$ のみとなった画像数枚から，光源の事前計算位置と目標とする位置から最も近い k 個の事前計算位置についての重み付き平均を取った画像を生成し，間接光成分のシミュレート結果 $\tilde{L}_{Indir}(\boldsymbol{l'})$ とする．

$$\tilde{f}_r\tilde{L}_{Indir}(\boldsymbol{l'}) = \tilde{f}_r(\alpha_1\tilde{L}_{Indir}(\boldsymbol{l_1}) + \alpha_2\tilde{L}_{Indir}(\boldsymbol{l_2})...) \tag{11}$$

ここで，α_i は $\boldsymbol{l'}$ と $\boldsymbol{l}$ の距離の逆比から計算した重みとする．直接光成分 $\tilde{L}_{Dir}(\boldsymbol{l'})$ は先に述べた通り実際にシステムを利用する際に計算し，足し合わせることとする．

シミュレーションから得られた光源移動後の画像と実環境の画像において，それぞれの間接光成分のシミュレート結果を $\tilde{L}_{Indir}(\boldsymbol{l'})$，$\tilde{L}_{Indir}(\boldsymbol{l})$ と，直接光成分のシミュレート結果を $\tilde{L}_{Dir}(\boldsymbol{l'})$，$\tilde{L}_{Dir}(\boldsymbol{l})$ とおき，$\tilde{\lambda}(\boldsymbol{l},\boldsymbol{l'})$ を求めると式 12 のようになる．

$$\begin{aligned} \tilde{\lambda}(\boldsymbol{l},\boldsymbol{l'}) &= \frac{\tilde{f}_r(\tilde{L}_{Indir}(\boldsymbol{l'}) + \tilde{L}_{Dir}(\boldsymbol{l'}))}{\tilde{f}_r(\tilde{L}_{Indir}(\boldsymbol{l}) + \tilde{L}_{Dir}(\boldsymbol{l})} \\ &= \frac{\tilde{L}_{Indir}(\boldsymbol{l'}) + \tilde{L}_{Dir}(\boldsymbol{l'})}{\tilde{L}_{Indir}(\boldsymbol{l}) + \tilde{L}_{Dir}(\boldsymbol{l})} \end{aligned} \tag{12}$$

そして，得られた $\tilde{\lambda}(\boldsymbol{l},\boldsymbol{l'})$ を実画像 $I(\boldsymbol{l})$ に掛け合わせることで，光源を編集した結果の画像 $I(\boldsymbol{l'})$ が得られる．

$$I(\boldsymbol{l'}) \approx I(\boldsymbol{l}) * \tilde{\lambda}(\boldsymbol{l},\boldsymbol{l'}) \tag{13}$$

4. 実時間照明操作 AR システムの試作

本システムにおけるライティングの計算は全て Unity[15] のライティングシステムを利用する．Unity はゲームエンジンであり，大域照明の計算を行うことができる．Unity の大域照明には Baked，Mixed，RealTime の 3 種類があるが，いずれも事前計算を必要とし，static なオブジェクトにしか効果を及ぼさない．Baked では，シーン内の static と呼ばれる，静的なオブジェクトに対し，間接光を事前に計算し，ライトマップと呼ばれるテクスチャに保存する．そして実際にシーンを実行する際には保存された計算結果を参照して描画する．このようにして高い品質の大域照明をシステムの実行中に表現できる．間接光を利用した，よりリアルなソフトシャドウの生成も可能である．また，ライトマップは実行中の書き込みはできないが，切り替えは可能である．Mixed は static なオブジェクトは Baked と同じくライトマップでライティングを行い，static でないオブジェクトには直接光の影響だけを与えるというもので，大域照明としての Baked との違いはない．RealTime では，大域照明の計算を実行時に行うために，光の反射経路を全て事前に計算しておくものである．だが，この事前計算リアルタイム大域照明は Baked と比べ解像度が 10 分の 1 以下になるほか，大きなライトの変動があった場合は計算に時間がかかる等の問題がある [16]．本研究ではより写実的な間接光を表現するため，大域照明の計算には Baked を利用する．

4.1 処理の流れ

システムの実際の処理の流れを説明する．本システムは，事前処理と実際に照明編集を行う処理に分かれている．

4.1.1 事前処理

事前処理としては，大まかに以下の 3 つを行う．

(1) シーンの形状計測とデータの最適化
RGB-D カメラである XtionPRO を用い，3 次元復元ソフト ReconstructMe[17] で対象の空間の 3 次元復元を行う．復元したデータそのままではポリゴン数が多く，ライティングの計算に非常に時間がかかるため，一旦 3DCG ソフトウェアである blender[18] にインポートしポリゴン数の削減と余分なポリゴンの削除，UV マップの追加を行う．

(2) カメラのキャリブレーション
カメラのキャリブレーションも事前に行っておき，現実のカメラと同じ視点の映像を Unity 内で生成もしくは合成できるようにする．まず，web カメラの内部パラメータをあらかじめ GML Camera Calibration Toolbox[19] で取得しておき，2 次元画像上の点と Unity 内に取り込んだメッシュ上での 3 次元点の対応を 6 点以上手動で求めておく．それらのデータから，OpenCV[20] の CalcExtrinsic 関数を用いてカメラの外部パラメータを導出し，さらにその外部パラメータから Unity 内でのカメラの位置を求める．

(3) ライトマップのベイキング
手順 1 で編集したモデルを Unity にインポートし，シーン中の static なオブジェクトとしてライトマップのベイキングを行う．この時生成するライトマップは現実のシーンを再現した光源の場合と数パターンの光源を編集した場合である．

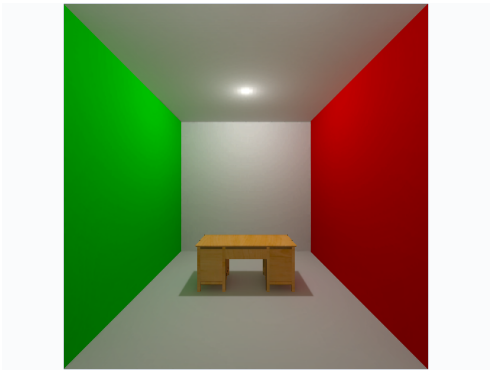


図 1 CG モデル 元画像

5. 動作実験

5.1 実験 1：CG 画像による評価

手法が意図通りに動作することを確認するため，CG 画像に対してシステムを使用し，評価を行った．現実の部屋を想定した部屋の画像を CG で合成し，その画像に対してシステムを適用することで手法の確認を行う．

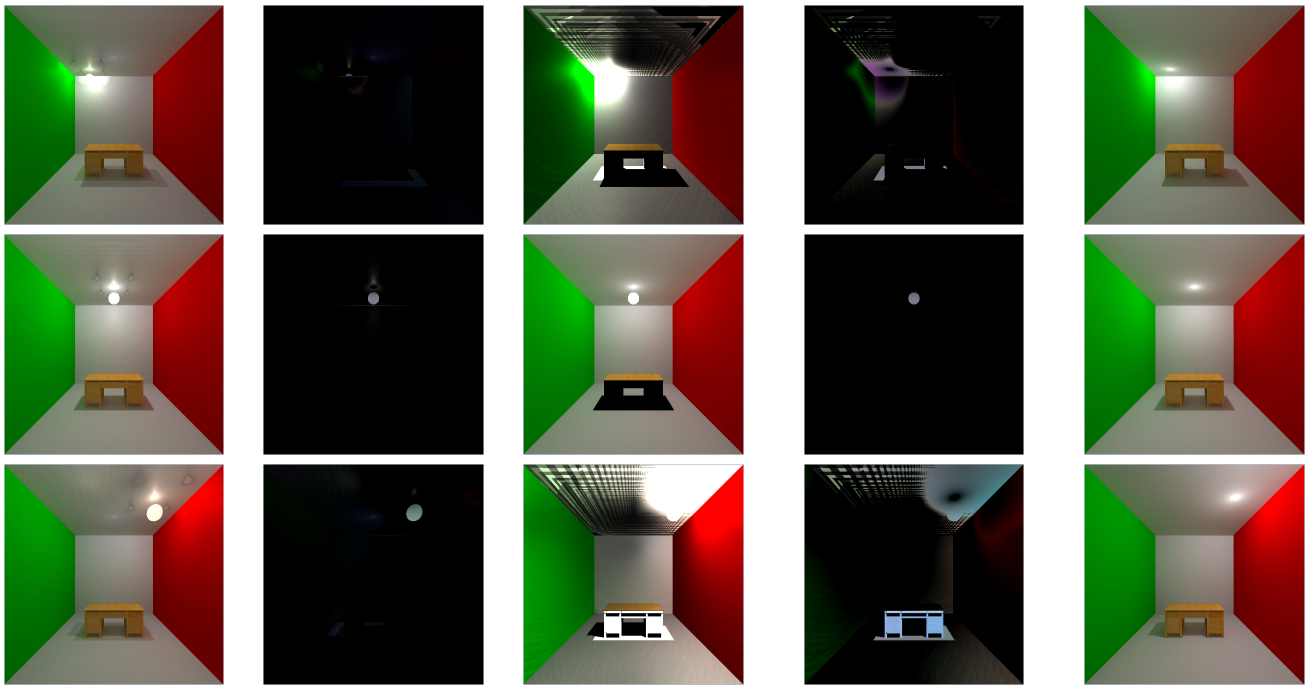
5.1.1 実験結果

システムを適用した元の CG 画像を**図 1** に示す．次に，システムの適用結果・目標画像・システムの適用結果と目標画像の差分画像を示す．また，比較のために直接光成分のみ式 13 に適用し編集した結果と目標画像との差分画像をそれぞれ光源位置ごとに**図 2** に示す．

5.1.2 評価

CG 画像に対してシステムを適用した結果，差分画像からも分かるように大域照明部分はほぼ同様の画像が得られており，システムは有用であると言える．また，実際にシステムを利用した際，CG の元画像から最終結果が出るまで約 0.3 秒で処理が終わっており，インタラクティブな速度で動作していると言える．

目標画像との正規化相互相関を編集後画像・直接光のみ編集画像のそれぞれに対して求めたところ，**図 3** のように直接光のみのものと比較してどの画像についても 1 に近い値が得られた．このことからも提案システムは時間をかけて計算した大域照明と大差無い画像を短時間で合成することができると言える．エリア 4 の直接光のみの画像が 1 に近くなるのは，図 2 の 2 段目からわかるように元の光源



編集後　編集後と目標の差分画像　直接光のみ　直接光のみと目標の差分画像　目標画像

図 2　CG 画像に対する実験結果

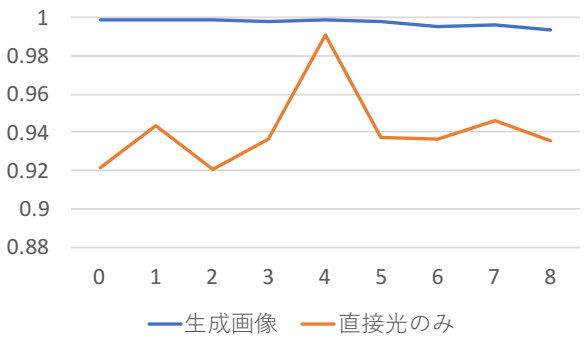


図 3　正規化相互相関

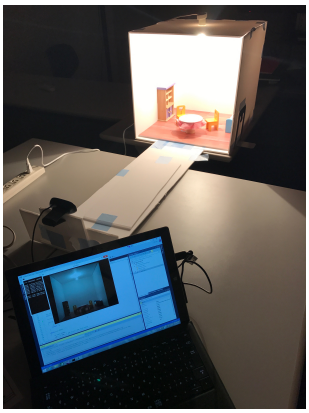


図 4　実験環境の様子

位置と同じ場所に光源を置いた場合を計算しようとしたため，シミュレーション同士で打ち消しあった結果ほとんどの部分で元の画像が描画されているためである．

5.2　実験 2：実画像による評価

実際に現実の環境を撮影した映像に対してシステムを使用し，評価を行った．実験は**図 4** のような環境で行う．

5.2.1　実験設定

現実の部屋を想定した環境を 40cm 四方のスチレンボードとミニチュア家具，100W 型ミニクリプトン電球で再現し，web カメラで動画を撮影する．そして撮影した動画に対して実際にシステムを適用する．部屋の内装・家具については，大域照明の効果が分かりやすい壁色のものと，より環境を複雑にするため複数の家具を置いたものの 2 パターンで行った．システムの適用結果と現実に光源位置を変えてみた結果の比較を行い，システムの有用性について評価を行う．

5.2.2　実験結果

はじめにシステムを適用した元の環境の画像を示し，実験を行った結果と光源を実際に編集した場合の画像，直接光成分のみで編集した結果を示す．まず，部屋の内装が単純なパターン（**図 5**）にシステムを適用した結果と目標とする現実画像を**図 6** に示す．また，部屋の内装が複雑なパターン（**図 7**）にシステムを適用した結果と目標とする現実画像を**図 8** に示す．

5.3　考察

5.3.1　計算時間についての評価

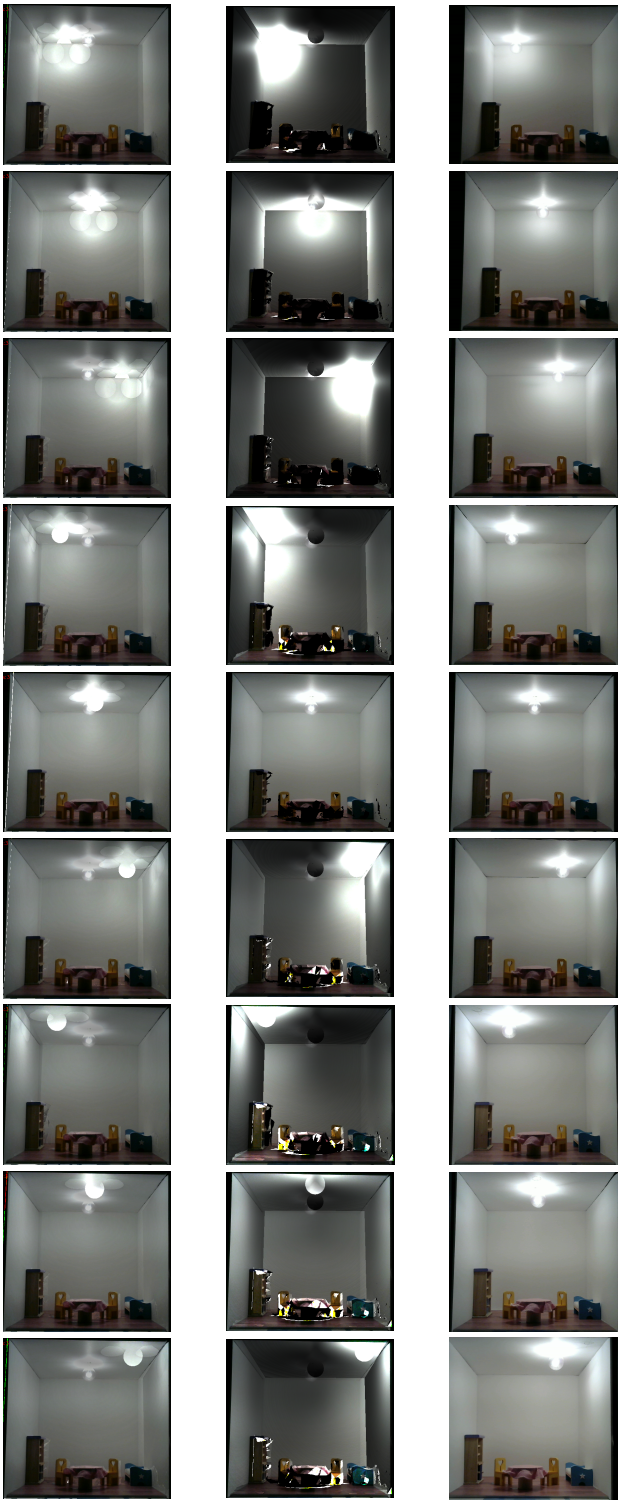
システムは入力の現実画像に対して平均 3fps で動いており，照明の移動の入力から約 0.3 秒で照明位置を変化させた画像を作成できている．このことからインタラクティ

図 5 現実画像（単純な内装）

編集後　直接光のみ　目標画像

図 6 現実画像（単純な内装）に対する実験結果

図 7 現実画像（複雑な内装）



編集後　直接光のみ　目標画像

図 8 現実画像（複雑な内装）に対する実験結果

ⓒ 2017 Information Processing Society of Japan

ブに動作していると言える．今回のシステムで利用したモデルのライトマップの生成には約 1,2 分はかかっていることから，提案手法は約 200 分の 1 の時間で光源環境の変化に対応した大域照明画像を生成できると言える．

5.3.2 光学的整合性問題についての評価

壁などを中心に大部分において，目標画像と比較してみても比較的光学的整合性の取れた画像が生成できていると言える．また，直接光成分のみを計算対象とした場合に比べると直接光成分のみのものは光が当たっていると白に，当たっていないと黒にくっきりとコントラストが現れたが，本システムの方では完全な黒ではなく柔らかい影の表現や壁の光の当たり具合など大域照明の表現が再現できており，より写実的になっていると言える．

本研究の手法の実現において最も問題となっているのが現実画像とシミュレーション画像の位置合わせである．これらの画像が合わない大きな原因としては以下のようなものがある．

- 3 次元再構築の精度
  この点に関しては使用しているソフトウェアに完全に依存するのでより精度の良い設定にするもしくはより正確に計測できるソフトウェアを使うことでしか解決できない．
- 手動によるキャリブレーション
  モノクロの CG 上の点とカラー実画像との間で特徴点の対応を自動検出するのは難しいと考えられ，手動で行うしかないので，指定する点を増やすことである程度は改善すると考えられる．
- Unity 内カメラと現実のカメラとのカメラ特性の差
  Unity のカメラコンポーネントの設定上ではレンズ歪みなどの設定は存在せず，完全に一致させることは難しいと考えられる．

6. おわりに

本研究では，対象の空間の 3 次元復元から算出した光源編集画像の事前計算と現実画像の掛け合わせにより，AR 環境において光学的整合性が比較的取れ，かつ計算時間が短く照明を編集することができるシステムを提案した．実験から，システムを適用した AR 環境の画像が比較的現実画像に近い結果がインタラクティブな速度で得られたことより，光学的整合性とリアルタイム性をある程度両立できる手法であると言える．今後の課題としては，対象の空間が動的であった場合へのシステムの対応や実環境の再現シミュレーションの精度の向上などが考えられる．

謝辞 本研究の一部は JSPS 科研費 JP16H02858 の助成を受けた．

参考文献

[1] Place IKEA furniture in your home with augmented reality, available from ⟨https://www.youtube.com/watch?v=vDNzTasuYEw⟩ (accessed 2017-02-14).
[2] Phan, V. T. and Choo, S. Y.: Interior Design in Augmented Reality Environment, *International Journal of Computer Applications*, Vol. 5, No. 5, pp. 16–21 (2010).
[3] 坂内大樹ほか：マジカル・ミステリー・ルーム–家具再配置を例とした実時間隠消現実感体験，第 19 回日本バーチャルリアリティ学会大会論文集，pp. 308–309 (2014).
[4] 森　磨美ほか：Pinch-n-Paste: AR 環境におけるテクスチャ転写用ジェスチャインタラクションシステム，研究報告コンピュータビジョンとイメージメディア（CVIM），Vol. 2014, No. 18, pp. 1–8 (2014).
[5] 上田将司，北原　格，大田友一：フリーハンド映像から抽出した壁領域における複合現実型壁紙シミュレーション，電子情報通信学会技術研究報告. PRMU, パターン認識・メディア理解， Vol. 111, No. 379, pp. 233–238 (2012).
[6] Gruber, L. and Ventura, J. and Schmalstieg, D.: Image-space illumination for augmented reality in dynamic environments, *2015 IEEE Virtual Reality (VR)*, pp. 127–134 (2015).
[7] Jachnik, J., Newcombe, R. A. and Davison, A. J.: Real-time surface light-field capture for augmentation of planar specular surfaces., *the 11th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 91–97 (2012).
[8] 安原広幸ほか：AR のための光源環境と物体の反射特性のオンライン推定，電子情報通信学会技術報告, パターン認識・メディア理解 (PRMU)， Vol. 112, No. 385, pp. 341–346 (2013).
[9] 倉地紀子：CG MAGIC：レンダリング，オーム社 (2007).
[10] 眞鍋知久：照明条件を変化させた画像の高速生成手法に関する研究，修士論文，広島大学 (2012).
[11] Iwanicki, M.: Lighting Technology of “The Last of Us”, *ACM SIGGRAPH 2013 talks*, p. 20:1 (2013).
[12] Mukaigawa, Y., Sumino, K. and Yagi, Y.: Rapid BRDF Measurement using an Ellipsoidal Mirror and a Projector, *IPSJ Transactions on Computer Vision and Applications*, Vol. 1, pp. 21–32 (2009).
[13] Marschner, S. R. et al.: Image-Based Bidirectional Reflectance Distribution Function Measurement, *Applied Optics*, Vol. 39, No. 16, pp. 2592–2600 (2000).
[14] 向川康博：反射・散乱の計測とモデル化，情報処理学会研究報告, コンピュータビジョンとイメージメディア (CVIM)， Vol. 172, No. 34, pp. 1–11 (2010).
[15] Unity - Game Engine, Unity (online), available from ⟨http://japan.unity3d.com⟩ (accessed 2017-02-14).
[16] Unity - マニュアル: グローバルイルミネーション，Unity（オンライン），入手先 ⟨https://docs.unity3d.com/jp/current/Manual/GIIntro.html⟩ (参照 2017-02-14).
[17] ReconstructMe — Real Time 3D Scanning Software, available from ⟨http://reconstructme.net⟩ (accessed 2017-02-14).
[18] Blender, available from ⟨https://blender.jp⟩ (accessed 2017-02-14).
[19] GML C++ Camera Calibration Toolbox — Graphics and Media Lab, available from ⟨http://graphics.cs.msu.ru/en/node/909⟩ (accessed 2017-02-14).
[20] OpenCV, available from ⟨http://opencv.org/⟩ (accessed 2017-02-14).

ⓒ 2017 Information Processing Society of Japan