Vol.2020-HCI-188 No.6
Vol.2020-EC-56 No.6
2020/6/1

GANによる光学的整合性が保たれたAR画像の生成手法の提案

池谷 駿弥[1,a)]　佐藤 正章[1]　井村 誠孝[1,b)]

概要：AR(Augmented Reality，拡張現実感) においてバーチャル物体を写実的に表現するには，光学的整合性の問題を解決する必要がある．本研究では実物体から光源情報等の推定を行わず，ディープラーニングを用いた生成モデルである GAN(Generative Adversarial Networks，敵対的生成ネットワーク) により，光学的整合性が保たれていない AR 画像を，光学的整合性が保たれた AR 画像に変換する End-to-End な手法を提案する．GAN による画像生成結果から，ドロップシャドウやバーチャル物体への周辺現実物体の映り込みなどの表現が付加され，GAN による実世界と整合した光学的整合性の表現を行うことが可能であると分かった．

キーワード：拡張現実，光学的整合性，敵対的生成ネットワーク，ディープラーニング，コンピュータグラフィックス

1. はじめに

スマートフォンの普及と性能向上により，AR(Augmented Reality，拡張現実感) 技術が身近なものになりつつある．AR 技術を利用したシステムやコンテンツにおいて，ユーザ体験を向上させるためには，時間的整合性，幾何学的整合性，光学的整合性を考慮する必要がある．時間的整合性とは，ユーザの視点移動などの現実環境における行動に対し，バーチャル物体の描画を遅延なく行うことである．時間的整合性を保つためには，コンピュータの性能向上や効率的なアルゴリズムの使用が重要である．幾何学的整合性とは，現実空間に対するバーチャル物体の位置合わせを行うことである．幾何学的整合性を保つ手法として，マーカーベースの手法と，マーカーレスの手法の大きく 2 つに大別される．マーカーベースの手法では AR マーカーを用いて，AR マーカーの位置，姿勢を基準とした視点位置，姿勢を推定することで幾何学的整合性を保っている．マーカーレスの手法では，視野内に存在する現実物体の自然特徴点を検出し，検出した自然特徴点を追跡することで幾何学的整合性を保っている．光学的整合性とは，現実環境の光を考慮してバーチャル物体をレンダリングすることで，バーチャル物体の陰影や写り込みなどの表現を違和感なく提示することである．光学的整合性を保つために，現実環境に実物体を配置し，実物体の見え方から光源情報を推定する手法 [1] が存在するが，実物体の利用や配布には制限が多い．幾何学的整合性の実現のためにマーカーレスの手法を利用する AR システムでは，AR マーカーを排除したにも関わらず，光学的整合性実現のために実物体を配置する必要があるという制約が残る．

本研究の目標は，現実環境の制約を排除し，多くの AR システムで容易に利用可能な，光学的整合性を実現するシステムの構築である．本研究では多くの AR システムで容易に利用可能とするため，AR システムの出力画像をディープラーニングにより変換することで光学的整合性を実現するアプローチを採る．ディープラーニングを用いた生成モデルの 1 つとして，GAN(Generative Adversarial Networks，敵対的生成ネットワーク)[2] が存在する．本研究では GAN を用いて，光学的整合性が保たれていない AR 画像を，光学的整合性が保たれた AR 画像に変換する手法を提案する．本手法により生成された画像を，光学的整合性が考慮されていない AR システムのレンダリング結果画像の代わりにユーザへ提示することで，光学的整合性を実現する．

2. 関連研究

2.1 実物体の観測による光学的整合法

安室ら [1] は，立体マーカーを用いて，幾何学的整合性と光学的整合性が保たれたバーチャル物体の表現を可能と

[1] 関西学院大学
Kwansei Gakuin University
[a)] eud16238@kwansei.ac.jp
[b)] m.imura@kwansei.ac.jp

している．立体マーカーは 2 次元 AR マーカーと鏡面球から構成されており，2 次元 AR マーカーにより視点カメラの位置姿勢を取得し，鏡面球からバーチャル物体重畳位置における照光条件を取得している．

Pilet ら [3] は，複数台のカメラと既知のテクスチャ付き平面物体を用いて，簡素な光源探査を行っている．平面物体から放射照度を取得し，バーチャル物体に写実的な陰影を付与している．

Gruber ら [4] は，実世界の任意の幾何形状を再構成することで，バーチャル物体の陰影の表現を可能としている．拡散反射を仮定することで，RGB-D カメラによりリアルタイムに再構成された幾何形状表面の輝度から環境マップの球面調和関数による展開の係数を求め，推定された環境マップからバーチャル物体の陰影の表現を可能としている．

2.2 GAN による画像生成と変換

ディープラーニングを用いた深層生成モデルの一種として，Goodfellow らによって発表された GAN(Generative Adversarial Networks，敵対的生成ネットワーク)[2] が存在する．GAN は与えられたデータの特徴を学習し，新規データの生成や，データ特徴に沿ったデータ変換を行うことができる．GAN は生成器と判別器と呼ばれる 2 つのニューラルネットワークから構成される．生成器は新規データの生成を行い，判別器は入力されたデータが学習データか，生成器が生成したデータなのかを推定する．生成器は判別器が誤判別するように，判別器は正しく判別できるように，お互いを敵対させながら学習を進めることで，最終的に生成器は与えられたデータ特徴に沿った新規データを生成できるようになる．

Isola ら [5] は, 画像から画像への変換を可能とした pix2pix と呼ばれる GAN を提案している．pix2pix の学習データ画像は，生成器への入力画像と生成器の出力として期待する出力画像の 2 枚 1 組のペア画像群である．生成器が学習データのペア画像間の写像を学習することで，汎用的な画像変換タスクを単一のネットワーク構造で実現している．また，生成器は画像特徴を畳み込み演算による低次元への圧縮後，逆畳み込み演算により元の解像度に復元するエンコーダデコーダ構造を採っている．畳み込み演算で得た画像特徴を逆畳み込み演算にも利用する U-Net[6] 構造を用いることで，入力画像のピクセル位置情報を考慮した画像変換を可能にしている．

Iizuka ら [7] は，画像の大域的かつ局所的な整合性を考慮した画像補完手法を提案している．補完ネットワークは欠損画像を入力として受け取り，欠損領域を補完した画像を出力する．大域識別ネットワークでは，補完ネットワークが生成した画像が画像全体として整合性のある構造になっているかを評価している．局所識別ネットワークでは，画像補間を行った領域のみの詳細な自然さについて評価を行っている．これにより，画像の大域的かつ局所的な整合性を考慮した画像補完を実現しており，複雑な画像補完を可能としている．

2.3 ディープラーニングを用いた光学的整合法

Georgoulis ら [8] は，実世界の鏡面反射物体の単一画像から照明条件と表面反射特性を推定する手法を提案している．鏡面反射物体の形状や反射率，現実環境の照明条件が未知の場合でも，複数の CNN(Convolutional Neural Network，畳み込みニューラルネットワーク) を用いて，鏡面反射物体の反射率マップを推定し，推定した反射率マップから別の CNN により BRDF(Bidirectional Reflectance Distribution Function，双方向反射分布関数) のパラメータと環境マップを推定している．

Mandl ら [9] は，バーチャル物体表面における球面調和関数のパラメータをリアルタイムに推定する手法を提案している．多数の球面調和関数パラメータでバーチャル物体を照らし，一様に分布した各視点カメラ位置姿勢でレンダリングを行った画像を複数の CNN の学習データとして利用している．学習後は，視点カメラの位置姿勢から該当する CNN を選択し，球面調和関数パラメータを推定している．

小川ら [10] は，陰影が付与されていない CG 物体に対して，陰影を付与する手法を提案している．陰影が付与された参照物体と陰影が付与されていない重畳物体をレンダリングした CG 画像から，pix2pix を用いて重畳物体に陰影を付与した画像を生成している．小川らが利用している学習データ画像およびテストデータ画像は，参照物体，重畳物体，周辺環境とも CG によって構成されているため，実世界に整合した陰影の付与はまだ確認されていない．

3. 提案手法

3.1 概要

本提案手法は GAN を用いて，光学的整合性が保たれていない AR システムのレンダリング結果画像を，光学的整合性が保たれた AR 画像に変換する手法である．提案手法の概要を**図 1** に示す．GAN の学習を行うため，光学的整合性が考慮された AR システムと，考慮されていない AR システムを用意し，2 つの AR システムのレンダリング結果画像群を GAN の学習データ画像として利用する．GAN の学習後，光学的整合性が考慮されていない AR システムによって生成された AR 画像をユーザへ提示せず，学習済みの GAN の生成器に入力する．AR システムのレンダリング結果画像の代わりに，GAN の生成器によって生成された画像をユーザに提示することで，光学的整合性が保たれた AR 体験を実現する．

3.2 GAN のニューラルネットワーク構造

提案する GAN のニューラルネットワーク構造を**図 2** に

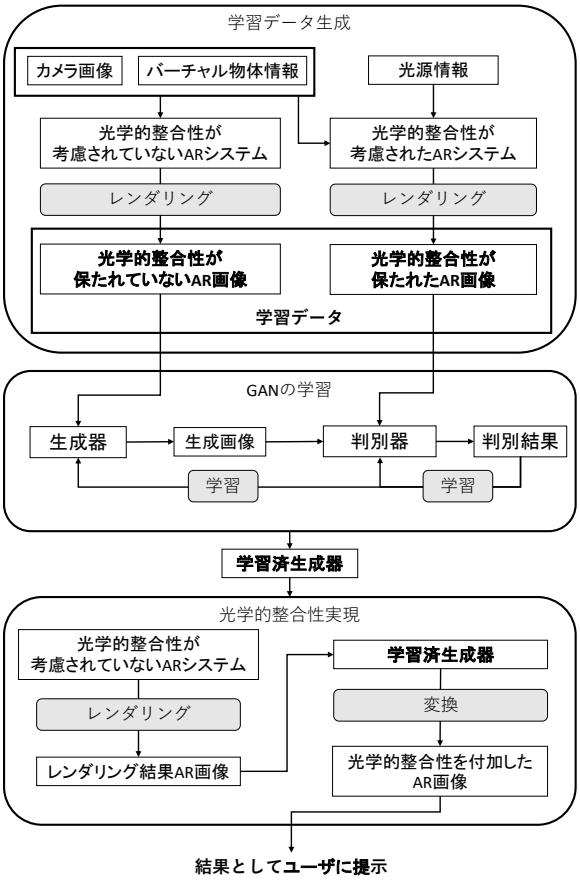


図 1　提案手法の概要

示す．提案する GAN は，Isola ら [5] の pix2pix と Iizuka ら [7] の手法を参考にしており，1 つの生成器と 2 つの判別器から構成される．

3.2.1　生成器

生成器は光学的整合性が保たれていない AR 画像を，光学的整合性が保たれた AR 画像に変換する．

生成器は光学的整合性が保たれていない AR 画像と，バーチャル物体の画像上での描画位置を示すマスク画像の 2 枚の画像を入力として受け取る．生成器はエンコーダデコーダ構造になっており，入力された画像は畳み込み層，拡張畳み込み層 [11]，逆畳み込み層を経て，光学的整合性が保たれた AR 画像 1 枚に変換され出力される．畳み込み層では，特徴マップのダウンサンプリングを行い，より小さな特徴マップへと次元圧縮を行う．拡張畳み込み層では，膨張係数の値を変え，フィルタの要素間の間隔を広げた畳み込み演算による効率的な受容野の拡大を行うことで，特徴マップの大域的な情報を集約する．逆畳み込み層では，特徴マップのアップサンプリングを行い，生成器への入力画像と同じ解像度の画像への復元を行う．

また，生成器は畳み込み演算で得た画像特徴を逆畳み込み演算にも利用する U-Net[6] 構造をもつ．逆畳み込み層において前層の出力のチャネルベクトルと，畳み込み演算で得た特徴マップのチャネルベクトルを結合させ，結合後の特徴マップに対して逆畳み込み演算を行う．

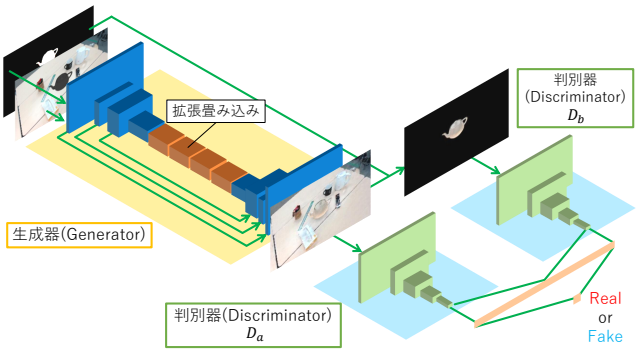


図 2　GAN のニューラルネットワーク構造

生成器は拡張畳み込み層と U-Net 構造により，画像の大域的整合性と入力画像のピクセル位置情報を考慮した画像変換が可能である．

3.2.2　判別器

判別器は生成器の学習の補助を行う．提案する GAN は，2 つの判別器をもつ．どちらの判別器も畳み込み層で構成されており，最終層のみ全結合層である．

判別器 D_a は，生成器の出力画像もしくは学習データのうち光学的整合性が保たれた AR 画像を入力として受け取り，画像の大域的整合性の評価を行う．判別器 D_b は，判別器 D_a への入力画像のバーチャル物体描画領域以外の画素値を 0 に置き換えた画像を入力として受け取り，バーチャル物体描画領域における局所的整合性の評価を行う．最終的に，判別器 D_a と判別器 D_b の出力を結合したベクトルを全結合層に入力し，判別器への入力画像が学習データ画像か生成器が生成した画像かの判別結果を取得する．

3.3　生成器と判別器の学習

光学的整合性が保たれていない AR 画像から，光学的整合性が保たれた AR 画像への変換を可能とするため，平均絶対誤差 (Mean Absolute Error，MAE) と GAN の損失関数 [2] を組み合わせた損失関数を利用する．平均絶対誤差による損失関数は式 (1) のように表される．

$$\mathcal{L}_{L_1}(G) = \mathbb{E}_{x,y,m}[\|y - G(x,m)\|_1] \qquad (1)$$

ここで，x は学習データの光学的整合性が保たれていない AR 画像を，y は学習データの光学的整合性が保たれた AR 画像を，m は学習データのマスク画像を，$G(x,m)$ は生成器の出力画像を表す．よって $\mathcal{L}_{L_1}(G)$ は，生成器によって生成された画像が，学習データの光学的整合性が保たれた AR 画像とどれだけ異なるかの誤差を表す．また，GAN の損失関数 [2] は式 (2) のように表される．

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,m}[\log(1 - D(G(x,m)))] \qquad (2)$$

ここで，$D(y)$ は判別器が y を学習データの光学的整合性が

保たれた AR 画像であると判断する確率を表し，$D(G(x,m))$ は判別器が $G(x,m)$ を生成器が生成した画像であると判断する確率を表す．学習データもしくは生成器が生成した画像を判別器が正しく判断できるようになると，$\mathcal{L}_{GAN}(G,D)$ の損失値は大きくなる．一方で，生成器が生成した画像を判別器が学習データであると誤判別すると，$\mathcal{L}_{GAN}(G,D)$ の損失値は小さくなる．式 (1) と式 (2) を組み合わせたネットワーク全体の損失関数は式 (3) のように表される．

$$G^* = \arg\min_G \max_D \mathcal{L}_{GAN}(G,D) + \lambda \mathcal{L}_{L_1}(G) \qquad (3)$$

ここで，λ は $\mathcal{L}_{L_1}(G)$ の重みを表すパラメータである．

生成器は，式 (1) により，学習データの光学的整合性が保たれた AR 画像に似た画像を生成できるように学習する．また，式 (2) より，判別器は出力した判別結果が正しくなるように学習を行い，生成器は生成した画像を判別器が学習データ画像だと誤判別するように学習を行う．結果，式 (3) において，判別器は G^* を最大化するように学習を行い，生成器は G^* を最小化するように学習を行う．このように，生成器と判別器を敵対させながら交互に学習を進めることで，最終的に生成器は高度に整合性が保たれた AR 画像を生成できる．

各学習段階において，生成器による画像生成後，生成画像 n 枚と学習データの光学的整合性が保たれた AR 画像 n 枚を判別器に与える．判別器の出力である判別結果と式 (2) により，判別器の損失値を計算し，誤差逆伝播により判別器ニューラルネットワークの重みパラメータを更新する．その後，判別結果と式 (1)，式 (2) により，生成器の損失値を計算し，判別器と同様に生成器ニューラルネットワークの重みパラメータを更新する．

4. 実装

4.1 データセット画像の生成方法

データセット画像の生成を行うため，現実環境下における撮影を行った．室内の机上に照明環境を取得するための全方位カメラを設置し，その周囲を web カメラとスポットライトを移動させながら撮影を行った．web カメラには Logicool C922 PRO STREAM WEBCAM を利用し，幅 1920px，高さ 1080px の解像度で撮影した．全方位カメラには RICOH THETA SC を利用し，幅 1920px，高さ 960px の解像度で撮影した．Python 3.6.8 によって web カメラと全方位カメラのシャッター開閉を同期し，どちらのフレームレートも約 30fps で撮影した．web カメラによるフレーム画像の例を**図 3** に，全方位カメラによるフレーム全天球画像の例を**図 4** に示す．

現実環境における撮影後，オープンソース 3DCG ソフトである Blender 2.8 を用いて，バーチャル物体を重畳した AR 画像をレンダリングした．幾何学的整合性のための視点カメラの位置姿勢推定には，Kanade-Lucas-Tomasi トラッキングを実装している Blender のモーショントラッキング機能を利用した．光学的整合性が保たれていない AR 画像は，モーショントラッキング機能による幾何学的位置合わせのみを考慮し，光源は環境光によるアンビエントオクルージョンによりレンダリングした．光学的整合性が保たれた AR 画像は，全方位カメラ画像による環境マッピングとイメージベースドライティング，レイトレーシング法によってレンダリングした．どちらの AR 画像でも，バーチャル物体は全方位カメラの位置に重畳し，バーチャル物体のマテリアルは拡散反射と鏡面反射を 1 対 1 で混合させたものを設定した．また，バーチャル物体描画領域の画素値を 255 とし，それ以外の領域画素値を 0 とするマスク画像もレンダリングした．どれも幅 1920px，高さ 1080px の解像度の画像として出力し，3 枚 1 組とするデータセットを生成した．生成したデータセットの例を**図 5** に示す．

図 3 web カメラによる撮影画像

図 4 全方位カメラによる撮影画像

4.2 GAN のニューラルネットワーク実装

提案した GAN のニューラルネットワークを実装するため，Python 3.6.8 とニューラルネットワーク実装ライブラリである Chainer を使用した．ニューラルネットワークと学習の実装に使用したライブラリ群を**表 1** に示す．また，GAN の実装と学習に使用した計算機の仕様を**表 2** に示す．

実装した生成器の構造を**表 3** に示す．畳み込み層，拡張畳み込み層，逆畳み込み層はそれぞれ 4 層であり，それぞれの層の出力に対してバッチ正規化を行った．また，各層の活性化関数には式 (4) に示す $\alpha = 0.2$ の LeakyReLU 関数

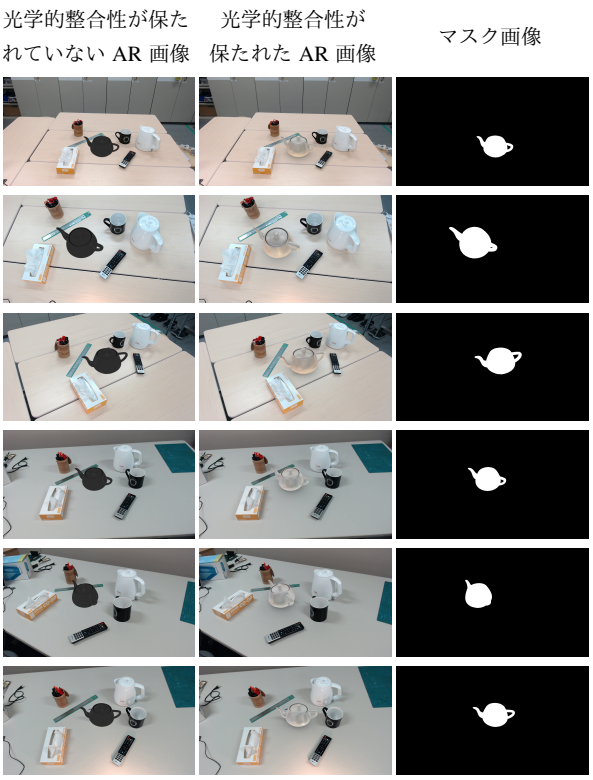


図 5　生成したデータセット画像の例

表 1　実装に使用したライブラリ

ライブラリ名	バージョン
Chainer	6.0.0
CUDA	10.0.130
cupy-cuda100	6.0.0
numpy	1.16.3
Pillow	5.1.0

表 2　実装に使用した計算機の仕様

項目	仕様
OS	Microsoft Windows 10 Home
CPU	Intel Core i7-9700K @ 3.60GHz (8 コア)
メモリ	32GB 2666MHz
GPU	NVIDIA GeForce RTX 2070 (2304CUDA コア，1620MHz，8GB メモリ)

を使用した．ただし，最終層の逆畳み込み層の出力はバッチ正規化を行わず，活性化関数は tanh 関数を使用した．

$$y = \begin{cases} x & (x \geq 0) \\ \alpha x & (x < 0) \end{cases} \tag{4}$$

実装した 2 つの判別器の構造を**表 4** に示す．2 つの判別器は共通の構造を持ち，どちらも畳み込み層 5 層，全結合層 1 層で構成した．1 層目の畳み込み層の出力に対してはバッチ正規化を行わず，活性化関数として $\alpha = 0.2$ の LeakyReLU 関数を使用した．2，3，4，5 層目の畳み込み層の出力に対してはバッチ正規化を行い，活性化関数として $\alpha = 0.2$ の LeakyReLU 関数を使用した．最終層の全結合層の活性化関数には式 (5) に示す sigmoid 関数を使用した．

$$y = \frac{1}{1 + e^{x}} \tag{5}$$

表 3　実装した生成器の構造

層	入力サイズ	出力サイズ	フィルタ	ストライド	膨張係数	活性化関数
Convolution	256 × 144 × 6	128 × 72 × 16	4 × 4	2 × 2	1	-
Batch Normalization	128 × 72 × 16	128 × 72 × 16	-	-	-	LeakyReLU
Convolution	128 × 72 × 16	128 × 72 × 32	3 × 3	1 × 1	1	-
Batch Normalization	128 × 72 × 32	128 × 72 × 32	-	-	-	LeakyReLU
Convolution	128 × 72 × 32	64 × 36 × 64	4 × 4	2 × 2	1	-
Batch Normalization	64 × 36 × 64	64 × 36 × 64	-	-	-	LeakyReLU
Convolution	64 × 36 × 64	64 × 36 × 128	3 × 3	1 × 1	1	-
Batch Normalization	64 × 36 × 128	64 × 36 × 128	-	-	-	LeakyReLU
Dilated Convolution	64 × 36 × 128	64 × 36 × 128	3 × 3	1 × 1	2	-
Batch Normalization	64 × 36 × 128	64 × 36 × 128	-	-	-	LeakyReLU
Dilated Convolution	64 × 36 × 128	64 × 36 × 128	3 × 3	1 × 1	4	-
Batch Normalization	64 × 36 × 128	64 × 36 × 128	-	-	-	LeakyReLU
Dilated Convolution	64 × 36 × 128	64 × 36 × 128	3 × 3	1 × 1	8	-
Batch Normalization	64 × 36 × 128	64 × 36 × 128	-	-	-	LeakyReLU
Dilated Convolution	64 × 36 × 128	64 × 36 × 128	3 × 3	1 × 1	16	-
Batch Normalization	64 × 36 × 128	64 × 36 × 128	-	-	-	LeakyReLU
Deconvolution	64 × 36 × 128	64 × 36 × 64	3 × 3	1 × 1	1	-
Batch Normalization	64 × 36 × 64	64 × 36 × 64	-	-	-	LeakyReLU
Concat	64 × 36 × 64, 64 × 36 × 64	64 × 36 × 128	-	-	-	-
Deconvolution	64 × 36 × 128	128 × 72 × 32	4 × 4	2 × 2	1	-
Batch Normalization	128 × 72 × 32	128 × 72 × 32	-	-	-	LeakyReLU
Concat	128 × 72 × 32, 128 × 72 × 32	128 × 72 × 64	-	-	-	-
Deconvolution	128 × 72 × 64	128 × 72 × 16	3 × 3	1 × 1	1	-
Batch Normalization	128 × 72 × 16	128 × 72 × 16	-	-	-	LeakyReLU
Concat	128 × 72 × 16, 128 × 72 × 16	128 × 72 × 32	-	-	-	-
Deconvolution	128 × 72 × 32	256 × 144 × 3	4 × 4	2 × 2	1	tanh

表 4　実装した判別器の構造

層	入力サイズ	出力サイズ	カーネル	ストライド	活性化関数
Convolution	256 × 144 × 3	128 × 72 × 16	4 × 4	2 × 2	LeakyReLU
Convolution	128 × 72 × 16	64 × 36 × 32	4 × 4	2 × 2	-
Batch Normalization	64 × 36 × 32	64 × 36 × 32	-	-	LeakyReLU
Convolution	64 × 36 × 32	32 × 18 × 64	4 × 4	2 × 2	-
Batch Normalization	32 × 18 × 64	32 × 18 × 64	-	-	LeakyReLU
Convolution	32 × 18 × 64	16 × 9 × 128	4 × 4	2 × 2	-
Batch Normalization	16 × 9 × 128	16 × 9 × 128	-	-	LeakyReLU
Convolution	16 × 9 × 128	8 × 4 × 256	4 × 4	2 × 2	-
Batch Normalization	8 × 4 × 256	8 × 4 × 256	-	-	LeakyReLU
Fully Connected	8192	512	-	-	sigmoid

表 5　2 つの判別器の出力結合層

層	入力サイズ	出力サイズ
Concat	512, 512	1024
Fully Connected	1024	1

2 つの判別器の出力を結合したものを全結合層に入力し，最終的な判別結果を得た．2 つの判別器の出力の結合と全結合層の構造を**表 5** に示す．

4.3　学習手法の実装

3.3 節で述べたように，生成器と判別器の損失値を交互に計算して誤差逆伝搬により学習を行った．生成器，判別器のニューラルネットワークの重みパラメータの更新手法には Adam[12] を用い，式 (3) の λ は 10 に設定し学習を行った．

また，安定した学習を行わせるために，Shrivastava ら [13] が提案している手法を利用した．Shrivastava らは GAN の学習過程において，判別器は生成器が過去に生成した画像に対しても正しく判別すべきであると仮定している．そのため，生成器によって直近に生成された画像だけではなく，過去に生成器が生成した画像も判別器に与えることで，判別器が過去の生成画像についても正しく判別できる手法を提案している．本研究で実装した判別器にも Shrivastava らが提案している手法を利用することで，GAN の安定した学習を実装した．

5. 結果と評価

5.1 学習済みの生成器による画像生成結果

2 つの異なる室内照明条件において 4.1 節で述べた手法により，2606 組，計 7818 枚の画像群を生成した．生成した画像の例を図 5 に示す．生成した画像を幅 256px，高さ 144px の解像度に縮小し，ランダムに選択した 2506 組を学習データ，残りの 100 組をテストデータとして利用した．2506 組の学習データ画像を用いて，32 ミニバッチで約 55000 エポック，約 2 週間の時間をかけ，GAN を学習させた．GAN の学習後，学習済みの生成器に対して，テストデータの光学的整合性が保たれていない AR 画像を入力し，画像変換を行った．

学習済みの生成器による画像変換結果を**図 6** に示す．学習済みの生成器による 1 回の画像変換には，約 83.0ms の時間を要した．図 6(a) は生成器への入力である光学的整合性が保たれていない AR 画像を，図 6(b) は生成器への入力であるマスク画像を，図 6(c) は生成器の出力画像として期待する光学的整合性が保たれた AR 画像を，図 6(d) は生成器の出力画像を示している．図 6(a) では，バーチャル物体のドロップシャドウや，バーチャル物体への周辺現実物体の映り込みの表現が存在しなかったが，図 6(d) ではこれらの表現が付加されていることが分かる．また，図 6(c) と図 6(d) の表現が非常に似ており，GAN によって実世界と整合した表現が可能であることが分かった．

(a) 入力画像

(b) 入力マスク画像

(c) 正解画像

(d) 生成画像

図 6 学習済みの生成器による AR 画像変換結果

5.2 提案手法と pix2pix の比較

提案手法の GAN のニューラルネットワークを評価するため，比較対象として Isola らの画像変換手法 pix2pix[5] を実装した．実装した pix2pix を 5.1 節で述べた学習データを用い，32 ミニバッチで約 55000 エポック，約 5 日の時間をかけ学習させた．その後，学習済みの提案手法の生成器と pix2pix の生成器を用いて，5.1 節で述べたテストデータに対する画像変換を行った．提案手法と pix2pix による，テストデータに対する画像変換結果を**図 7** に示す．図 7(c) ではバーチャル物体のマテリアル表現が図 7(a) と異なって見えるが，図 7(b) ではバーチャル物体のマテリアル表現は図 7(a) と非常に似ている．また，図 7(c) では画像上の現実物体領域への影響も確認されるが，図 7(b) では現実物体領域への影響が非常に小さいことがわかる．

また，同様のテストデータを用い，生成画像と正解画像の間の全画素に対する平均二乗誤差を計算した．生成画像と正解画像の平均二乗誤差による，提案手法と pix2pix の比較を**表 6** に示す．表 6 より，提案手法は pix2pix に比べ生成画像と正解画像間におけるピクセル値の差分が小さく，提案手法は pix2pix に比べ高度に光学的整合性が保たれた画像を生成することができると言える．

表 6 生成画像と正解画像の平均二乗誤差による比較

ネットワーク	平均	標準偏差
pix2pix	44.150	11.921
提案手法	3.121	1.180

5.3 提案手法の汎用性に関する検証

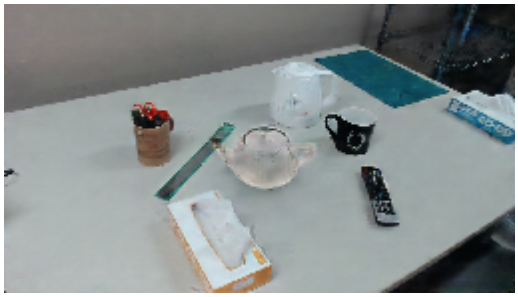
AR 体験では様々な要因が変動することが考えられる．光学的整合性を実現する AR システムにおいて，考慮すべき変動要因は以下が挙げられる．

- バーチャル物体の形状
- バーチャル物体のマテリアル
- 現実環境の光源環境
- 現実周辺物体とその配置

これらのうち，バーチャル物体の形状が異なる AR 画像

(a) 正解画像

(b) 提案手法により生成された画像



(c) pix2pix により生成された画像

図 7 提案手法と pix2pix により生成された画像の比較

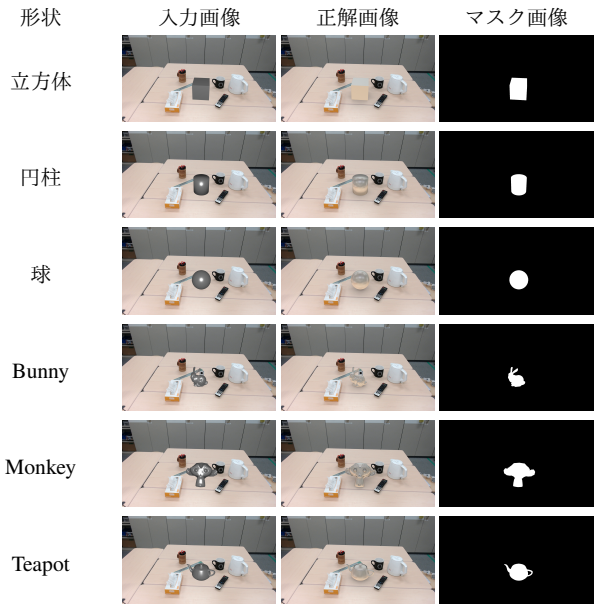


図 8 バーチャル物体形状が異なるデータセット

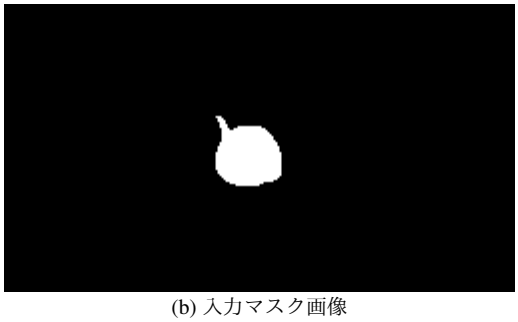
に対し，提案手法の対応可能性を検証した．4.1 節で述べた手法により，室内光源環境において撮影を行い，バーチャル物体の形状のみが異なる学習データとテストデータを生成した．ただし，光学的整合性が保たれていない AR 画像は 4.1 節で述べた手法とは異なり，バーチャル環境の光源として点光源をカメラの位置に追加した．学習データのバーチャル物体には立方体，円柱，球，Bunny，Monkey を使用し，それぞれ 1266 組，計 18990 枚の画像群を生成した．テストデータのバーチャル物体には Teapot を使用し，1266 組の画像群を生成した．生成した学習データ画像とテストデータ画像例を**図 8** に示す．

学習データ画像を幅 256px，高さ 144px の解像度に縮小し，32 ミニバッチで約 30000 エポック，約 2 週間の時間をかけ，GAN を学習させた．GAN の学習後，学習済みの生成器に対して，テストデータ画像を入力し，画像変換を行った．学習データに含まれない Teapot の AR 画像に対する画像変換結果を**図 9** に示す．図 9(a) は生成器への入力である光学的整合性が保たれていない AR 画像を，図 9(b) は生成器への入力であるマスク画像を，図 9(c) は生成器の出力画像として期待する光学的整合性が保たれた AR 画像を，図 9(d) は生成器の出力画像を示している．学習データに存在しないバーチャル物体形状でも，ドロップシャドウの表現は図 9(c) と図 9(d) で似ている．しかし，図 9(d) の局所的な形状の表現が消失している．これは，学習データに含まれるバーチャル物体の局所的形状に偏りがあったことが原因であると考えられる．また，損失関数に勾配が一定である平均絶対誤差を用いたことから，生成画像と正解画像の誤差が大きい時と小さい時で学習率が変わらず，局所領域における微小な最適化が行えなかったことも原因として考えられる．

6. おわりに

本研究では，現実環境の制約を排除し多くの AR システムで容易に光学的整合性を実現するために，GAN を用いて光学的整合性が保たれていない AR 画像を，光学的整合性が保たれた AR 画像に変換する手法を提案した．画像の大域的整合性とピクセル位置を考慮した画像変換を行う生成器と，画像の大域的整合性と局所的整合性をそれぞれ評価する 2 つの判別器により，光学的整合性を実現する GAN のニューラルネットワーク構造を提案し実装した．また，GAN を学習させるための画像データの生成を行い，生成した画像データを用いて GAN の学習を行い，画像変換を行った．さらに GAN の評価を行うため，pix2pix との比較，バーチャル物体形状の変更に対する対応可能性の検証を行った．学習済みの GAN の生成器による画像変換結果から，ドロップシャドウやバーチャル物体への周辺現実物体の映り込みなどの表現が付加され，GAN によって実世界と整合した光学的整合性の表現を行うことが可能であると分かった．また，pix2pix との比較から，現実物体領域への画像変換の影響が発生しにくく，pix2pix に比べ高度に光学的整合性が保たれた画像を生成できることが分かった．


(a) 入力画像


(b) 入力マスク画像


(c) 正解画像


(d) 生成画像

図 9 バーチャル物体形状に対する提案手法の汎用性に関する検証結果

一方，バーチャル物体形状の変更に対する対応可能性の検証から，学習データに存在しないバーチャル物体の局所的形状の表現が消失することが確認された．

今後の課題としては，バーチャル物体の形状やマテリアル，周辺光源環境，周辺現実物体を変えた多様な環境への対応を可能とするために，損失関数の工夫やニューラルネットワーク構造の改良，学習データの拡充化を行うことが挙げられる．また，本研究で生成した画像は比較的低解像度であるため，ユーザが違和感を感じないように高解像度化することや，高解像度でもバーチャル物体表現に違和感を感じないような画素値の微調整を行える工夫が必要であると考えられる．

参考文献

[1] 安室喜弘，石川　悠，井村誠孝，南　広一，眞鍋佳嗣，千原國宏：立体マーカを用いた実空間における仮想物体の調和的表現，映像情報メディア学会誌，Vol. 57, No. 10, pp. 1307–1313 (2003).

[2] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y.: Generative adversarial nets, *Advances in neural information processing systems*, pp. 2672–2680 (2014).

[3] Pilet, J., Geiger, A., Lagger, P., Lepetit, V. and Fua, P.: An all-in-one solution to geometric and photometric calibration, *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 69–78 (2006).

[4] Gruber, L., Richter-Trummer, T. and Schmalstieg, D.: Real-time photometric registration from arbitrary geometry, *2012 IEEE international symposium on mixed and augmented reality (ISMAR)*, pp. 119–128 (2012).

[5] Isola, P., Zhu, J.-Y., Zhou, T. and Efros, A. A.: Image-to-image translation with conditional adversarial networks, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134 (2017).

[6] Ronneberger, O., Fischer, P. and Brox, T.: U-net: Convolutional networks for biomedical image segmentation, *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241 (2015).

[7] Iizuka, S., Simo-Serra, E. and Ishikawa, H.: Globally and Locally Consistent Image Completion, *ACM Transactions on Graphics (Proc. of SIGGRAPH 2017)*, Vol. 36, No. 4, pp. 107:1–107:14 (2017).

[8] Georgoulis, S., Rematas, K., Ritschel, T., Gavves, E., Fritz, M., Van Gool, L. and Tuytelaars, T.: Reflectance and natural illumination from single-material specular objects using deep learning, *IEEE transactions on pattern analysis and machine intelligence*, Vol. 40, No. 8, pp. 1932–1947 (2017).

[9] Mandl, D., Yi, K. M., Mohr, P., Roth, P. M., Fua, P., Lepetit, V., Schmalstieg, D. and Kalkofen, D.: Learning lightprobes for mixed reality illumination, *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 82–89 (2017).

[10] 小川敬也，間下以大，浦西友樹，Ratsamee, P.，竹村治雄：拡張現実感における RGB カメラ画像に整合した陰影付け，情報処理学会研究報告，Vol. 2019-CVIM-217, No. 32, pp. 1–8 (2019).

[11] Yu, F. and Koltun, V.: Multi-scale context aggregation by dilated convolutions, *arXiv preprint arXiv:1511.07122* (2015).

[12] Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).

[13] Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W. and Webb, R.: Learning from simulated and unsupervised images through adversarial training, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2107–2116 (2017).