CH-003

物理ベース微分可能レンダリングを用いた工業製品外観検査の検討

Investigation of Visual Inspection of Industrial Products Using Physics-Based Differentiable Renderings

森田 匠† 右田 剛史† 高橋 規一†
Takumi Morita Tsuyoshi Migita Norikazu Takahashi

1 まえがき

現在，国内における多くの工場が，工業製品の外観検査を人の目で行っている．しかしながらこの行為は多くの問題点を抱えている．まず，人の目で行う外観検査では，その人が外観検査をする際の不注意やコンディションの良し悪しによって，精度が大きく下がってしまう可能性がある．そのため，工業製品の傷や歪みを見落としてしまうリスクが存在する．また，外観検査をするために要員を割かなければならないので，人件費がかかってしまうという問題も存在する．さらに，人の目で行う外観検査において，長い期間で一定の精度を保っていくには，後継者の育成が必要である．しかしながら，近年後継者となる人材の不足が嘆かれており，長期的に見ても人の目で行っていくことは厳しくなっている．これらの問題に対し，近年，人工知能を用いた外観検査の自動化 [1][3] への期待が高まっている．この技術では，Deep Learning により適切な教師データを学習させることで，工業製品の不良個所と汚れの判別や，製品の形状や向き，色や光量の変化に対しても柔軟に対応することができる．これにより，どのような工業製品であっても，高精度を保ったまま外観検査をすることができる．また，人工知能による自動化により，人員を外観検査に割く必要がなくなるため，人件費の削減につながり，後継者の育成も不要となる．これらの利点により，外観検査を自動化することは人の目で行うことの多くのデメリットを解消することができる．しかしながら，人工知能の運用には多くの教師データを必要とし，工業製品一種類ごとに教師データを作成する労力や資金を払うことが難しい工場も存在する．よって多くの工場で外観検査の自動化を行うには，より少ないデータから工業製品のパラメータ推定をする手段が必要である．

そのパラメータ推定の手法の一つとして，近年コンピュータビジョン分野において微分可能レンダリング [2] に注目が集まっている．微分可能レンダリングとは，2 次元画像の 3 次元シーン情報に関する勾配を，自動微分を用いて計算できるようにしたレンダリングであり，生成画像と入力画像を比較することによって 3 次元シーン情報を推定していくことができる．微分可能レンダリングについてのより詳しい解説は，第 2 章で示すこととする．

上記のような背景により，本研究では外観検査の自動化を，微分可能レンダリングによるパラメータ推定を用いて行うことを目標とする．工業製品の写真と，工業製品の 3D モデル，光源，カメラを設置した仮想空間を入力し，仮想空間のパラメータ推定をするために，物理ベースの微分可能レンダラー Mitsuba3 [6] を利用する．Mitsuba3 に関しては，2.3 節で詳細な説明を記す．そして，製品の 2 次元画像（目標画像）から光源位置や製品の姿勢等のパラメータを，所定の Epoch 回推定して, 目標画像と見た目が同じになるような画像（推定後モデル画像）を生成する．また，今回の推定には異なる視点からなる複数枚の画像を使用しており，複数視点から推定をすることによって，3D データにおける物体や光源のパラメータの推定精度の向上を試みている．これにより，目標画像と推定後モデル画像を画素単位で比較することで，形状の違いや傷の有無の検出が期待できる．

† 岡山大学大学院環境生命自然科学研究科 Graduate School of Environmental, Life, Natural Science and Technology, Okayama University

2 微分可能レンダリング

2.1 微分可能レンダリングとは

まずレンダリングとは，3 次元シーン情報（物体の形状，光源，カメラ位置など）から 2 次元画像を生成することである．第 1 章でも示したが，微分可能レンダリングとは，2 次元画像の 3 次元シーン情報に関する勾配を自動微分を用いて計算できるようにしたレンダリングである．微分可能レンダリングによるパラメータ推定の詳細な流れ [2] は以下のようになっている (なお，本研究の作成プログラムでは，勾配降下法ではなくそれを発展させた最適化アルゴリズムである，Adam を用いて更新を行う).

1. 3 次元シーン情報（$\hat{\pi}$ で表す）の値を初期化する．
2. レンダリング関数 R を用いて画像 $\hat{I}$ を生成する．

$$\hat{I} = R(\hat{\pi})$$

3. 生成された画像 $\hat{I}$ と実際の画像 I の違いを，画像間の距離を測る適当な関数 D を用いて計算する．

$$L = D(I, \hat{I}) = D(I, R(\hat{\pi}))$$

4. L の値が減少するように $\hat{\pi}$ の値を勾配降下法で更新する（微分可能レンダリングを用いることで $\partial L/\partial \hat{\pi}$ が計算できる）.

$$\hat{\pi} \leftarrow \hat{\pi} - \eta \frac{\partial L}{\partial \hat{\pi}}$$

5. ステップ 2 から 4 を所定の回数繰り返す．

2.2 3 次元再構成

ここでは微分可能レンダリングの代表的な応用研究であり，本研究と大きくかかわる技術である「3 次元再構成」[2][5] について解説する．また，以下では仮想空間をレンダリングした画像のことをモデル画像とよぶ．

従来の多視点ステレオでは，複数の目標画像間での特徴点の対応を求め（対応点検出），特徴点座標の誤差に基づき画像内の物体を重ね合わせていく．しかしながら，微分可能レンダリングによる推定では対応点検出は

せずに，推定結果が正しいかどうかを，推定中パラメータを反映したモデル画像（以下では $\hat{I}$ とよぶ）と目標画像を見比べ（画素値の引き算），その誤差に基づき仮想空間の修正を繰り返していく．この処理によるパラメータ推定は 3 次元再構成とよばれる（3 次元再構成の手順は図 1 を参照）．この技術を使うことで，仮想空間上の物体，光源，カメラ等を動かしながら $\hat{I}$ を更新していき，目標画像と更新後のモデル画像を同じ画像にしていくことが可能である．

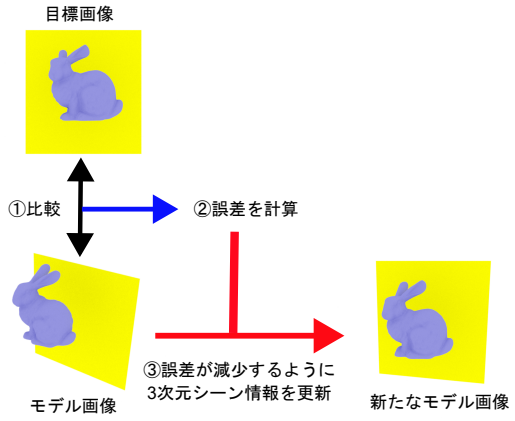

図 1　3 次元再構成の手順

また 3 次元再構成によって，物体の形状そのものを目標画像の物体に変化させていくこと [4] も可能である．今回の研究では仮想空間上の物体形状は既知のものとし，形状変化は加えないものとする．

2.3　Mitsuba3

ここでは，微分可能レンダリングを使用することのできるレンダリングシステムである，Mitsuba3 について解説していく [6].

Mitsuba3 は微分可能レンダラであり，カメラパラメータ，物体位置・姿勢，BSDF（双方向散乱分布関数），テクスチャなどの入力パラメータについて，生成した画像全体の導関数を計算することができる．また，Mitsuba3 は Python と深く統合されたレンダラである．そのため，PyTorch などの既存の Python ライブラリと組み合わせることが可能となっている．また，Mitsuba3 は現実で工業製品が光による影響をどのように受けるのか，画像上で再現することができる．Mitsuba3 では他にも，形状最適化，NeRF の推定，ボリュームレンダリング等様々なことに応用できるが，本研究の題意と反れてしまうため，省略する．

3　問題解決の流れ

この章では，本研究の問題設定および提案手法について説明する．

3.1　問題設定

ここでは，本研究の問題設定について示す．第 1 章でも示したが，本研究では，国内における多くの工場で，工業製品の外観検査がいまだに人の目で行われていることを問題視している．その現状を打開すべく，本研究では外観検査の自動化を目標としている．外観検査の自動化をする手段として，本研究では微分可能レンダリングを採用している．本研究は，目標画像とパラメータ推定したい 3D モデルと適切な初期値があれば問題なくプログラムを運用できるというメリットがある．このメリットは，AI などの膨大な教師データを必要する自動化形式ではできないことである．様々な傷のある工業製品を必要とする教師あり学習では多くのコストや労力を支払わなければならず，工場によってはこれを用意できない場合が多い．このことから本研究では微分可能レンダリングを自動化の手段として採用している．

3.2　提案手法

本研究の大目標を，微分可能レンダリングに基づく外観検査手法の開発としている．本研究ではその目標達成のために，以下の手順で外観検査が行われることを想定している．

1. 実際の工業製品の 2 次元画像（目標画像）から光源位置や製品の姿勢等のパラメータを微分可能レンダリングを用いて所定の Epoch 回反復して，推定後モデル画像を生成
2. 目標画像と推定後モデル画像を比較して工業製品の形状の違いや傷の有無を検出

また，図 2 は，提案手法の流れを簡略化して説明したものである．ここからは各手順についての説明を行う．まず手順 1 について説明する．前提として，この手順では工業製品の 3D モデルを所持しているものとする．また，微分可能レンダリングをするために，Mitsuba3 を使用している．まず，実際の工業製品の 2 次元画像（目標画像）を複数枚（この数を X と置く）取得する．次に工業製品の 3D モデル，光源，カメラを設置した仮想空間を用意する．次に仮想空間内の工業製品の物体位置・姿勢パラメータ，光源の位置パラメータ，カメラの位置・向きパラメータを最適化アルゴリズムの Adam による推定の対象として登録する．そして，X 台のカメラから仮想空間内の工業製品をレンダリングし，X 枚の目標画像のそれぞれに対応する $\hat{I}$ を取得した後，目標画像と $\hat{I}$ を比較することによるパラメータ推定を Adam で行う．最後に前文の処理を所定の Epoch 数繰り返す．カメラの台数，および目標画像の枚数を複数にすることのメリットとして，実際の工業製品における傷や歪みを様々な視点から検知できるようになること，仮想空間内での物体位置・姿勢の推定精度が向上することが挙げられる．

手順 2 では，手順 1 で出力した推定後モデル画像と目標画像を画素ごとに比較することによって，色彩のパラメータの違いから製品の傷や歪みなどを判別し，工業製品の外観検査を行うことができる．

本研究で作成するプログラムは，手順 1 を実装したものとする．

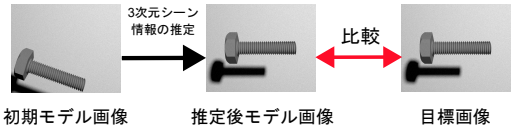

図 2　目標達成の手順

3.3　パラメータ推定プログラムの作成手法

ここでは，微分可能レンダリングによるパラメータ推定をするプログラムを，どのような手法で作成するのかを示す．パラメータ推定プログラムは，自動微分を用い

たものと差分商を用いたものの 2 種類作成するため，以下の各項でその手法を解説する．

また，今回のプログラムでは 4 つのカメラ視点からモデルパラメータを最適化していく（そのためモデル画像，目標画像はそれぞれ 4 枚となる）．また，微分可能レンダリングでモデル画像と目標画像を比較する際，両方の画像に平滑化を適用した後比較を行っている．これにより，画像をぼかして画像内の色の変化を連続的にし，オブジェクトの境目などにみられる外れ値の影響を低減することができるため，結果的に自動微分および step によるパラメータ推定を実行する際の精度が上がる．

3.3.1 自動微分を用いたパラメータ推定プログラムの作成手法

ここでは，Mitsuba3[1)]ライブラリ内にある backward を用いた微分可能レンダリングプログラムの作成手法について解説する．backward とは出力から入力に向かって勾配を自動計算する手法である．このプログラムは，適切な初期値設定と 3D オブジェクトや光源，カメラを設置した仮想空間を入力することで，3.2 節で紹介したパラメータ推定を行うことができる．しかしながら，テクスチャを付与したり，オブジェクトの個数がある基準を超えると backward の処理が重くなってしまい，プログラムが実用的に動作しなくなってしまう．そのため，仮想空間内のテクスチャ，オブジェクトの数が制限される．

パラメータ推定の大まかな手順は以下のようになる．

1. 目標画像，初期モデル画像の作成
2. 物体位置・姿勢，光源位置，カメラパラメータの推定
3. 手順 2 を所定の Epoch 回繰り返す
4. 推定後モデル画像出力，Epoch 毎の誤差をグラフ化

本稿では，目標画像は入力した仮想空間をレンダリングすることで作成し，実際の工業製品を撮影した写真は使用しない．初期モデル画像は仮想空間のパラメータに誤差を加え，それをレンダリングすることで作成する．また，物体位置・姿勢，光源位置，カメラ位置・向きの値を推定させるために，それぞれのパラメータを Adam に登録する．手順 2 では Adam を適用した微分可能レンダリングによるパラメータ推定を行う．この工程は 1 画像ずつ行われ，1 回の Epoch で各カメラパラメータは 1 回，物体位置・姿勢，光源位置は各視点から計 4 回の推定が行われる．そして手順 3 により $\hat{I}$ が更新されていく．このプログラムでは推定結果の収束度合いから，Epoch 数は 50 回としている．手順 4 では，推定後モデル画像出力，Epoch 毎の誤差をグラフ化を行う他，真値との最終的な誤差も出力する．

3.3.2 差分商によるパラメータ推定プログラムの作成手法

ここでは，差分商によるパラメータ推定プログラムの作成手法について解説する．3.3.1 項ではオブジェクトの数を増やしたり，オブジェクトの材質に複雑な BSDF（金属材質）を適用した状態では backward 中にプログラムの動作が極めて遅くなって（場合によっては止まって）しまい，実行することができなかった．これは backward の部分を，forward 微分の処理に変更しても同様の結果だった．しかしながらこの節で解説するプログラムでは，以上の問題点を解消し，物体の色や材質（このプログラムで扱う仮想空間内の工業製品には，表面の目が粗い金属材質を付与している）などを扱うことができる．このプログラムでは backward による自動微分の代わりに差分商を算出することで推定を行っている．差分商とは，微分を差分によって近似する手法のことである．これにより，各パラメータに対する勾配を自動微分に頼らず算出することができ，backward と比較してより速い処理速度で推定を進めることが可能であり，初期値の許容誤差も大きい．

このプログラムの手順は 3.3.1 項で紹介したプログラムと似ているが，異なるものである．Mitsuba3(drjit) には，自動微分以外の方法での最適化を柔軟に行う方法が用意されていないため，推定パラメータを Adam で更新する際や，差分商算出の際に PyTorch を組み込んでいる．パラメータ推定の手順は以下のようになる．

1. 初期パラメータを Mitsuba3 形式で取り出す
2. 初期パラメータを PyTorch 形式に変換し，PyTorch の Adam に登録する
3. PyTorch 形式の推定パラメータを Mitsuba3 レンダラの仮想空間に反映させる
4. 仮想空間をレンダリングし，$\hat{I}$ を作成する
5. 各パラメータ変化における目標画像と $\hat{I}$ の誤差 $(D(I, R(\hat{\pi})))$ の差分商を計算する
6. 手順 5 の結果をもとに PyTorch の Adam の step により推定を行う
7. 手順 3 から 6 を所定の Epoch 数繰り返す

また，3.3.1 項のプログラムからの変更点として，特定の Epoch 数まで達した時，各パラメータに適用している Adam の学習率をより小さな値となるように変更している．これによって，所定の学習率で推定を進めきった状態からより細かい推定を進めることができる．また，推定結果の収束度合いから，プログラム 2 の Epoch 数は 80 回としている．

4 評価実験

ここでは，第 3 節で示した提案手法を実際に Python プログラムで実現し，それがどの程度の精度になったのかを解説していく．今回のプログラムで使用した工業製品の 3D データは，株式会社クレファクト様から提供していただいたメッシュデータである．

4.1 自動微分を用いたパラメータ推定

ここでは，3.3.1 項のプログラムの実験結果について解説する．

プログラム 1 から出力された各画像は図 3, 4, 5，Epoch 毎の誤差の推移は図 6 のとおりである．横軸は Epoch，縦軸は損失関数の値（誤差）であり，各曲線はそれぞれのカメラ視点と対応している．図 3 と図 4 を比較すると画像内の工業製品の位置や影の伸び方が違う部分が多々見られる．しかし図 5 では，図 3 より見た目が図 4 に近くなっており，正しい方向にパラメータ推定ができていることがわかる．また，図 6 からわかるが，Epoch 数が 30 を超えた部分で誤差が再び増加していってしまっている．これは Adam の，動作毎に過去の勾配が蓄積されていく，という規則が原因となっている．この規則によ

1) 厳密には misuba3 の自動微分部分は drjit という別のパッケージである．

り，誤差が小さくなる方向が変わってしまった際にその方向が変更されず，推定が悪い方向（誤差が大きくなってしまう方向）に進んでしまうという現象が実験中多々見られた．また，工業製品と床を近づけ，影を多く反映させるとそうでない場合より推定精度が落ちてしまうことが確認できた．これは，影の配色が背景と同化してしまい，backward で影と配色の区別をつけることができないのが原因だと考えられる．他の原因としては，床に工業製品を近づけすぎると，物体の位置推定の途中で工業製品が床を貫通してしまうことがあり，見かけの物体形状が変わってしまうことから推定結果が悪くなってしまうパターンも存在する．

プログラム 1 の各推定パラメータと真値は表 1 のとおりである．物体姿勢以外のパラメータは仮想空間内のワールド座標によって定義されている．物体姿勢はワールド座標の任意の軸を基準として，表 1 の物体姿勢の値 ×100 度で回転させることで反映している．各カメラの位置パラメータは真値の誤差 −0.4 から+0.4 を乱数で生成している．その他のパラメータは誤差の絶対値が推定できる範囲の最大値になるように設定している（これ以上の誤差になると推定が失敗する）．推定結果を見ると，カメラの位置パラメータに推定値が真値に近づいた後超えて遠ざかってしまっている値が見られる（例としてカメラ 2 の推定値の z 軸の値があげられる）．これは物体位置の推定値との兼ね合いにより，微分可能レンダリングの「画像の見え方からどの方向に更新するのか決める」という手法で，画像の見え方だけ目標画像と同じようにしているのが理由の一つとして挙げられる．これは，カメラの数を増やし，より多視点から物体を観測できるようにする，または背景オブジェクトを増やして，それらを物体観測地点の目印とすることで解決する．しかしプログラム 1 の手法では処理速度の問題により，オブジェクトを増やしたり，複雑な材質を適用することができなかった．

またこのプログラムでは，学習率の大きさを固定としている．そのため誤差の値が推定によってある程度小さくなってきた場合，その誤差以上に推定を進めることができなくなってしまう場合がある．この問題は，Epoch 回数や誤差の大きさに応じて学習率を変えることができるようにすることで解決する．

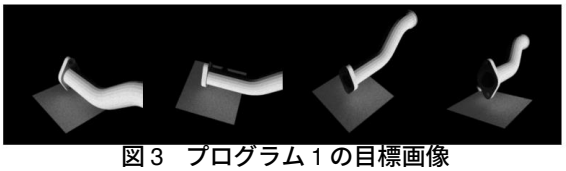

図 3　プログラム 1 の目標画像

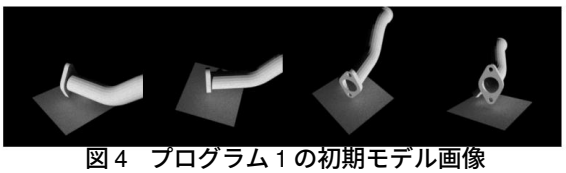

図 4　プログラム 1 の初期モデル画像

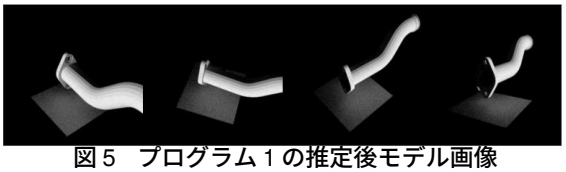

図 5　プログラム 1 の推定後モデル画像

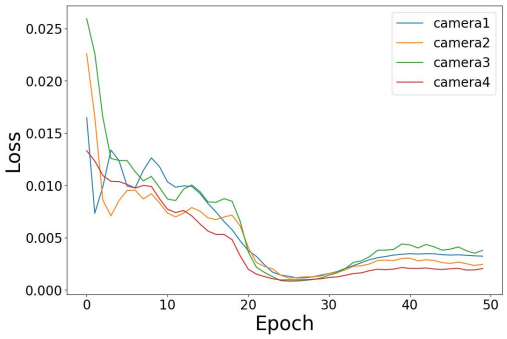

図 6　プログラム 1 の目標画像と $\hat{I}$ の Epoch 毎誤差グラフ

表 1　プログラム 1 の各推定パラメータ比較

	真値	初期値	推定値
カメラ 1	[7.9609, 6.6800, 5.9999]	[7.5673, 6.2981, 5.8763]	[7.9769, 6.7398, 5.9266]
カメラ 2	[1.8046, 10.2344, 5.9999]	[2.2009, 10.1839, 5.6633]	[1.9204, 10.2017, 6.3341]
カメラ 3	[−5.1961, 9.0, 5.9999]	[−5.5031, 9.1600, 5.7257]	[−5.43036, 8.6957, 5.9286]
カメラ 4	[−9.7655, 3.5543, 5.9999]	[−9.4159, 3.5551, 6.2907]	[−9.7701, 3.4759, 6.0402]
物体姿勢	0.0	0.2	−0.0315
物体位置	[0.0, 0.0]	[0.08, −0.2]	[−0.0909, 0.1161]
光源位置	[0.0,100]	[0.0,90]	[2.5004, −3.8799, 97.4399]

4.2　差分商によるパラメータ推定

ここでは，3.3.2 項のプログラムの実験結果について解説する．

プログラム 2 から出力された各画像は図 9, 10, 11, Epoch 毎の誤差の推移は図 12 のとおりである．横軸は Epoch，縦軸は損失関数の値（誤差）であり，各曲線はそれぞれのカメラ視点と対応している．図 9 と図 10 を比較すると画像内の工業製品の位置や金属光沢，影など違う部分が多々見られる．しかし図 11 では，図 9 より見た目が図 10 に近くなっており，正しい方向にパラメータ推定ができていることがわかる．出力結果からもわかるとおり，各画像に色彩や素材等のテクスチャを反映させた状態で推定・出力を行うことが可能となった．そして，仮想空間内により多くのオブジェクトを配置できるようになったため，入力する仮想空間内に部屋を作り，その部屋の内部に工業製品，光源のオブジェクトを配置している．これにより，部屋内での撮影という，より実践的なシチュエーションでのパラメータ推定が可能となった（今回の部屋では，推定の目印となるように，部屋の各壁を異なる色にしている）．またプログラム 2 では，仮想空間のすべてのオブジェクトに対するパラメータ推定中の SPP（あるピクセルに対してのサンプル使用数）を増やしている．例として，$\hat{I}$ の SPP はプログラム 1 が 4 だったのに対し，プログラム 2 では 32 に増やしている．これにより，仮想空間内で光の影響をより現実に近い形で再現することが可能である（SPP による見た目の違いの例として，図 7（SPP=4）と図 8（SPP=32）を示す）．また，図 12 を見ると，Epoch が 60 回付近の誤差が急に増加してしまっていることがわかる．これは，微分可能レンダリングの最中，学習率の更新を行う際に Adam をリセットしたことが原因となっている．これにより Adam 内に蓄積した勾配が一度消去され，再度 1 から推定を始めることになるため，このようなグラフとなる．

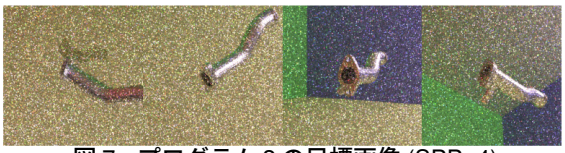

図 7　プログラム 2 の目標画像 (SPP=4)

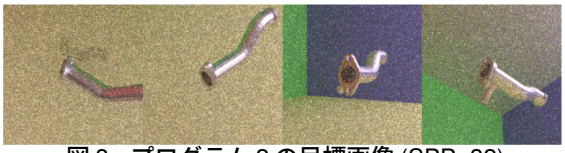

図 8　プログラム 2 の目標画像 (SPP=32)

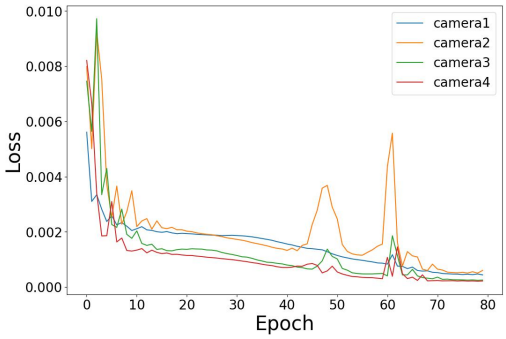

図 12　プログラム 2 の目標画像と $\hat{I}$ の Epoch 毎誤差グラフ

プログラム 2 の各推定パラメータと真値は表 2 のとおりである．各カメラの位置パラメータは真値の誤差 −0.5 から+0.5 を乱数で生成している．その他のパラメータは誤差の絶対値が推定できる範囲の最大値になるように設定している（これ以上の誤差になると推定が失敗する）．プログラム 1 と比較すると，初期値の誤差がより大きい状態から推定を進めることができる（例としてプログラム 1 のカメラの位置パラメータの誤差は −0.4 から+0.4 となっておりプログラム 2 の方が許容誤差が大きい）．さらに推定値と真値の値を比較すると，プログラム 1 と比べて物体姿勢・位置，光源位置の最終的な誤差が小さくなっていることがわかる．ただ，カメラ 1 の推定値は他カメラの推定値と比べて真値との誤差が大きくなっている．これは，他視点に比べて背景色のバリエーションが少ないことや，プログラム 1 と同様に影が関与した推定の精度が少し落ちてしまうことが原因だと考えられる．また，光源の位置によってはオブジェクトに白飛びが発生してしまう場合があり，その部分が外れ値となることがある．その場合には推定結果の精度が落ちてしまっていた．

表 2　プログラム 2 の各推定パラメータ比較

	真値	初期値	推定値
カメラ 1	[5.1961, 9.0000, 5.9999]	[5.1844, 8.7470, 5.7784]	[5.3372, 8.7104, 5.9904]
カメラ 2	[−5.1961, 9.0000, 5.9999]	[−5.4930, 9.1797, 6.3375]	[−5.1546, 9.0327, 6.0583]
カメラ 3	[−10.3923, −0.0000, 5.9999]	[−10.1503, 0.1447, 6.2921]	[−10.3970, 0.0008, 6.0086]
カメラ 4	[−5.1961, −9.0000, 5.9999]	[−5.4460, −9.1479, 5.9499]	[−5.2187, −9.0963, 6.0245]
物体姿勢	0.0	0.2500	0.0015
物体位置	[0.0, 0.0]	[0.1200, −0.3000]	[−0.0097, 0.0111]
光源位置	[50,50,50]	[44, 58, 44]	[49.3592, 49.6239, 49.9194]

また，このプログラムの普遍性を確かめるために，別の工業製品モデル [8] を使った推定も行った．出力された各画像は図 13, 14, 15，Epoch 毎の誤差の推移は図 16 のとおりである．横軸は Epoch，縦軸は損失関数の値（誤差）であり，各曲線はそれぞれのカメラ視点と対応している．各推定パラメータと真値は表 3 のとおりである．表 3 を見ると，表 2 と同じくカメラ 1 のパラメータ推定精度が落ちてしまうが，物体位置・姿勢の推定は高精度で推定を行えていることがわかる．しかしながら，光の当たり方の違いにより光源位置の推定結果は表 2 のものと比較して落ちてしまっているので，物体ごとに光源位置初期値の許容誤差を見直さなければならない．

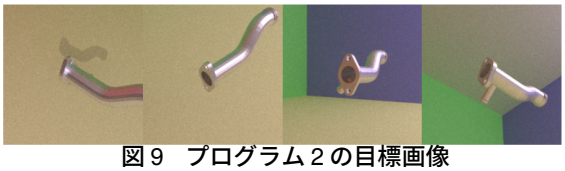

図 9　プログラム 2 の目標画像

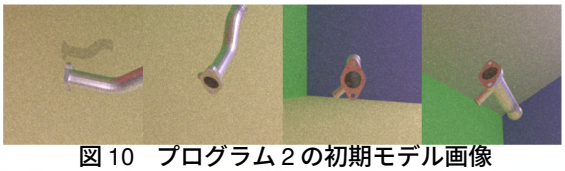

図 10　プログラム 2 の初期モデル画像

図 11　プログラム 2 の推定後モデル画像

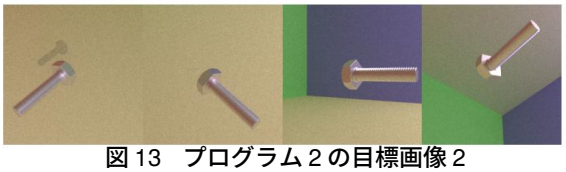

図 13　プログラム 2 の目標画像 2

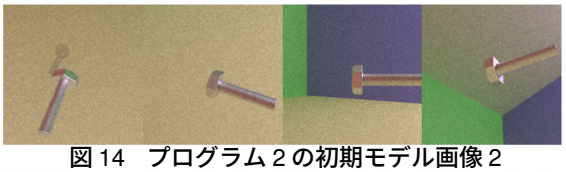

図 14　プログラム 2 の初期モデル画像 2

図 15　プログラム 2 の推定後モデル画像 2

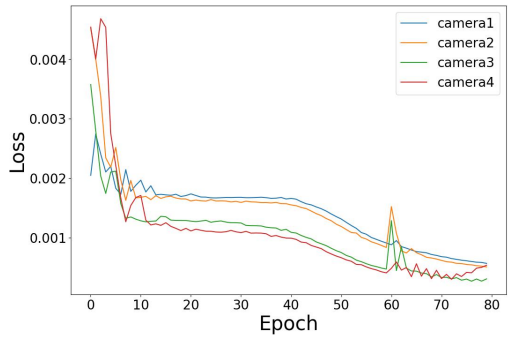

図 16　プログラム 2 の目標画像と $\hat{I}$ の Epoch 毎誤差グラフ 2

表 3　プログラム 2 の各推定パラメータ比較 2

	真値	初期値	推定値
カメラ 1	[5.1961, 9.0000, 5.9999]	[4.6991, 9.4740, 5.7606]	[5.0711, 9.4393, 5.6615]
カメラ 2	[−5.1961, 9.0000, 5.9999]	[−5.6203, 8.6204, 6.2042]	[−5.1715, 8.9094, 6.0909]
カメラ 3	[−10.3923, −0.0000, 5.9999]	[−10.2721, 0.4788, 5.7182]	[−10.4120, −0.0018, 6.0314]
カメラ 4	[−5.1961, −9.0000, 5.9999]	[−5.2114, −8.6815, 6.2049]	[−5.2078, −9.0365, 5.9916]
物体姿勢	0.0	0.2500	0.0073
物体位置	[0.0, 0.0]	[0.1200, −0.3000]	[−0.0041, −0.0038]
光源位置	[50,50,50]	[44, 58, 44]	[48.5524, 51.4451, 50.5642]

5　おわりに

本論文では，微分可能レンダリングを用いて工業製品の外観検査を行うための手法を提案した．プログラム 1 では，適切な初期値設定と 3D オブジェクトや光源，カメラを持つ仮想空間を入力することで，パラメータ推定を行うことができた．しかし，工業製品の金属光沢を再現できず，かつ初期値の許容誤差が狭い状態でしか推定を行うことができなかった．プログラム 2 では，工業製品の材質や光の当たり方による影響を考慮し，プログラム 1 より初期値の誤差を大きくした状態でも問題なく機能し，この推定手段の有用性を示した．しかしながら，誤差をより大きくした状態からの推定や影の多い状態での推定の苦手なシチュエーションも存在し，初期値設定を柔軟に行うことができないなど，未だ改善点は多くある．より実用性を得るには，初期値をより大きな誤差範囲で設定したものをプログラム 2 のような最終精度で推定できる必要がある．このような結果を得るための一例として，微分可能レンダリングとニューラルネットワークを併用することが挙げられる．ニューラルネットワークに工業製品の 3D モデルから作成した画像と，姿勢パラメータの組を教師データとして学習させることで，姿勢パラメータを画像から推定することが可能となる．この手法を使い，ニューラルネットワークでより大きな誤差からの推定を行うことで，誤差が小さくなった状態から微分可能レンダリングでより細かな推定を行うことができる．今後の研究ではニューラルネットワークと微分可能レンダリングを併用した外観検査の手法を確立していきたい．

参考文献

[1] S. Kubota, “Application of Deep Learning to Visual Inspection and Research Trends,” Journal of the Japan Society for Precision Engineering, vol.85, no.1, pp.27–30, 2019.

[2] 加藤大晴, “ニュウモン微分可能レンダリング − 3 次元ビジョンの新潮流！ 3 次元再構成から NeRF まで −,” コンピュータビジョン最前線 Autumn 2022, 共立出版, pp.78–114, 2022.

[3] 田中 拓哉, 笠原 亮介, “画像を用いた自動外観検査技術,” 日本画像学会誌,vol.55, no.3, 2016.

[4] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis, “Learning to estimate 3d human pose and shape from a single color image,” in CVPR, 2018.

[5] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka,Wadim Kehl and Adrien Gaidon, ”Differentiable Rendering: A Survey,” arXiv:2006.12057 [cs.CV], 2020.

[6] W. Jakob, S. Speierer, N. Roussel, M. Nimier-David, D. Vicini, T. Zeltner, B. Nicolet, M. Crespo, V. Leroy and Z. Zhang, Mitsuba 3 renderer, version 3.0.1, https://mitsuba-renderer.org, 2022.

[7] Mitsuba 3 Tutorials, Gradient-based optimization, https://mitsuba.readthedocs.io/en/latest/src/inverse_rendering/gradient_based_opt.html（2024 年 1 月 26 日アクセス）

[8] cgtrader, RH UNC 2A Bolt free 3D print model, 2019. https://www.cgtrader.com/free-3d-print-models/hobby-diy/mechanical-parts/unc-2a-bolt