芸術科学会論文誌 Vol. 15, No. 2, pp. 86-97 (2016)

物体輪郭を再現する境界拡張テクスチャ合成法

越後谷勇介（学生会員）　**横山俊希**　**藤本忠博**（正会員）

岩手大学大学院工学研究科

Boundary Expansion Texture Synthesis for Reconstructing Object Contours

Yusuke Echigoya[1] (Student Member)　**Toshiki Yokoyama**

Tadahiro Fujimoto[2] (Member)

Graduate School of Engineering, Iwate University

1) h23j013@eecs.iwate-u.ac.jp　2) fujimoto@cis.iwate-u.ac.jp

アブストラクト

ユーザがテクスチャ合成を制御する手法として提案されたユーザ制御テクスチャ合成は, ユーザが描いた目的画像に合わせて入力テクスチャ（入力画像）から出力テクスチャ（出力画像）を合成する．入力画像に物体と背景が含まれている場合，通常，ユーザ制御テクスチャ合成では，入力画像上の物体と背景の画素を同等に扱い，物体の輪郭を意図的には考慮しない．そのため，入力画像上の物体の画素領域だけを対象として異なる背景に対して合成を行いたい場合, ユーザが目的画像上におおまかに描いた物体の輪郭に合わせて入力画像上の物体の特徴的な輪郭を出力画像上に再現することは難しい．そこで，本研究では，境界画素と境界距離を用いた最適な入力画素の評価により，目的画像上の物体輪郭を適切に拡張し，入力画像上の物体輪郭を意図的に再現する手法を提案する．

Abstract

A user controlled texture synthesis technique was proposed to allow a user to control texture synthesis. The technique creates an output texture (output image) from an input texture (input image) by using a target image painted by a user. When an object and its background exist in the input image, the user controlled texture synthesis technique usually treats the pixels of the object and those of the background equally without treating the contour of the object intentionally. Therefore, if we want to synthesize only the input pixels of the object onto a different background, it is difficult to reconstruct the characteristic contour of the object on an output image by giving a simple contour roughly painted on a target image. In this paper, we propose a texture synthesis technique to reconstruct a characteristic object contour intentionally by expanding the boundary of a painted region on a target image suitably and selecting optimal input pixels to synthesize by the evaluation using boundary pixels and boundary distances.

1. はじめに

主にコンピュータグラフィックスやコンピュータビジョンの分野において，元となる入力画像から目的に応じた様々な出力画像を生成するイメージエディット（image editing）と呼ばれる画像合成技術の研究が活発に行われている．多くのイメージエディット手法の基礎となるテクスチャ合成（texture synthesis）は，入力画像（入力テクスチャ）の特徴を維持する大きさの異なる新たな出力画像（出力テクスチャ）を合成する技術である[1-15]．ユーザ制御テクスチャ合成[3]は有用な応用技術の一つであり，ユーザが自由にペイント描画した目的画像の色分布に合わせて入力画像から出力画像を合成する．このとき，入力画像に物体と背景が含まれている場合，物体と背景の画素を同等に扱い，物体の輪郭は意図的には考慮されない．そのため，異なる背景を扱うために，ユーザが物体だけを意図して描いたペイント領域に対して入力画像上の物体の画素だけを合成したい場合であっても，ユーザがおおまかに描いたペイント領域の輪郭に合わせて入力画像上の物体の特徴的な輪郭を再現した出力画像を生成することは難しく，通常，ペイント領域の輪郭に沿って入力画像上の物体が不自然に途切れてしまう．

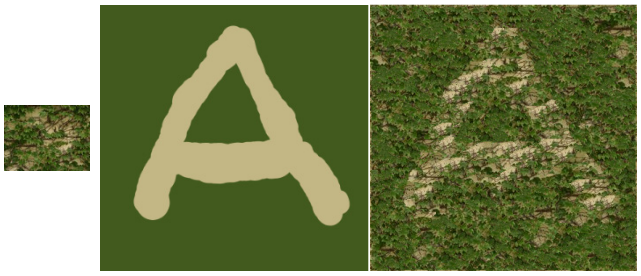
そこで，この問題を解決するため，本研究では，入力画像上の物体の輪郭，すなわち，物体と背景の境界に合わせてペイント領域の輪郭から外側に向けて合成領域の境界を適切に拡張し，入力画像上の物体輪郭を意図的に再現する境界拡張テクスチャ合成法を提案する．本手法は，ユーザ制御テクスチャ合成[3]とk-候補探索[4]を利用し，最適な合成画素を決定するための「境界距離」と「境界画素」を用いた近隣画素群の新たな評価方法に基づいて実現される．

以下，２節では関連研究の紹介，さらに，本手法と類似する既存手法との違いについて述べる．３節ではテクスチャ合成の基本アルゴリズムを説明する．４節では本手法の詳細について述べる．その後，５節で実験結果を示し，６節で結論と今後の課題について述べる．

2. 関連研究

2.1 テクスチャ合成

テクスチャ合成の方法は，局所的逐次合成法と大域的反復合成法に大別される．前者は入力画像と出力画像の間で局所的に近隣画素群の類似性を評価することで逐次的に出力画像の合成領域を拡大していく方法である．この方法は，さらに，ピクセルベース手法[1-4]とパッチベース手法[5-9]に分類される．前者は画素（ピクセル）単位で合成を行い，後者は隣接する画素群からなるパッチ単位で合成を行う．また，これらを融合した手法[10]も提案されている．一方，大域的反復合成法[11-15]は，出力画像の全画素を同時に反復的に更新することで，出力画像全体を大域的に最適な画像へと収束させていく．この方法は並列処理に向いており，フラグメントシェーダ等を利用してGPU上で高速に動作するような実装が可能である．なお，上記のそれぞれについて非常に多くの手法が提案されているため，ここでは主に初期の代表的なものだけを参考文献として挙げている．



入力画像　　目的画像　　出力画像

図 1．ユーザ制御テクスチャ合成の例

本研究で提案する手法は，ピクセルベース手法に基づく合成を行う．ピクセルベース手法では，通常，走査線順で出力画像の画素を合成する．出力画像上の処理画素について，その画素に隣接する合成済みの近隣画素群を考え，それと最も類似する色パターンの近隣画素群を持つ入力画像上の画素を探索し，その色を処理画素に合成する．最もシンプルな全探索法（full search）では入力画像上の全ての画素を探索するが，探索を効率化する候補探索法（coherence search）[3]があり，さらに，より良好な結果を得るために候補の数を増やすk-候補探索法（k-coherence search）[4]も提案されている．また，最近では，近似的に最類似の近隣画素群を高速に探索するANN探索（Approximate Nearest Neighbor Search）の方法として，PatchMatch法[16]やrandom walk法[17]が提案されている．

テクスチャ合成の応用技術として，サンプル画像を入力画像として扱い，ユーザが自由に描いた図形や画像に対してサンプル画像の特徴を与えた出力画像を生成するなど，サンプル画像を実例として用いることで新たな出力画像を生成する実例ベーステクスチャ合成（example-based texture synthesis）の様々な手法が提案されている．ユーザ制御テクスチャ合成（user controlled texture synthesis）[3]は，ピクセルベースの方法により，ユーザが描いた目的画像の色分布に合わせて入力画像から出力画像を合成する．図1にユーザ制御テクスチャ合成の例を示す．Image Analogies [18]は，ピクセルベースのテクスチャ合成を利用して，入力画像の雰囲気や作風を変える画像変換フィルタを実現する．実例として与えられた元画像Aと変換画像A’を用いて，入力画像Bに同等の変換を与えた出力画像B’を生成する．Texture-by-numbers [17-19]はImage Analogiesの応用であり，サンプル画像を構成要素（例えば，風景画像中の森林，草原，川，空など）ごとにラベリング色でセグメント分割した後，ラベリング色を用いて描いたペイント画像に合わせてサンプル画像に基づく出力画像を生成する．また，テクスチャ合成をベースとした高度なアプリケーションを目指した様々なイメージエディット技術が提案されている[12,13,16,20]．

2.2 物体輪郭を再現するテクスチャ合成

ユーザが描いた図形に対してサンプル画像に基づく出力画像を生成する実例ベーステクスチャ合成にとって，描いた図形の輪郭に沿ってサンプル画像上の物体やテクスチャパターンの特

徴的な輪郭を適切に再現することは重要である．しかし，通常のテクスチャ合成の手法は，それらの再現を意図的には考慮しない．合成過程において，前景（物体やテクスチャパターン）と背景を分離せず，前景が映る画素（前景画素）と背景が映る画素（背景画素）を同等に扱うことで，結果的にサンプル画像上の前景と背景の特徴的な境界が出力画像上に再現されることに頼るのが普通である．一方，前景の輪郭を意図的に考慮する手法として，Ritter らの“Painting with Texture”[21]と Lukac らの“Painting by Feature”[22]が提案されている．Ritter らは，後述のように，近隣画素群の評価に前景画素と背景画素の区別を導入することで，ユーザのペイント描画に合わせてサンプルテクスチャの輪郭を考慮した出力画像を生成するインタラクティブシステムを提案した．Lukac らは，１次元の特徴を持つ前景輪郭と２次元の特徴を持つ前景内部を分けて扱い，それぞれを描画するブラシ（brush）ツールとフィル（fill）ツールによってサンプル画像に基づく出力画像をペイント描画するインタラクティブシステムを提案した．

2.3 既存手法と提案手法との関連

本研究で提案する手法は，Lukac らの手法とは異なり，前景の輪郭と内部を分けることなく描画したペイント画像を扱い，Ritter らの手法に類似する．Ritter らのインタラクティブシステムは，まず，特徴的な境界を持つテクスチャの断片をサンプルテクスチャとし，幾つかのサンプルテクスチャをパレット上に読み込む．その後，ユーザが目的とするサンプルテクスチャを選択し，キャンバス上にブラシツールでインタラクティブに図形等をペイント描画することで，ペイントされた領域上にサンプルテクスチャの色パターンと境界形状を再現したテクスチャ合成がリアルタイムで行われ，出力画像が生成される．彼らのアルゴリズムは，前述した Image Analogies [18]と同様のピクセルベースの方法により，全探索と候補探索を用いた最適な近隣画素群の選択を行うことでサンプルテクスチャから出力画像への画素の合成を行う．ただし，境界を考慮するため，近隣画素群の評価に各画素の色だけではなく前景画素と背景画素の区別も導入している．具体的には，サンプルテクスチャと出力画像の間で評価する近隣画素群中の対応する画素どうしについて，(a)両方とも前景画素の場合，(b)両方とも背景画素の場合，(c)一方が前景画素で他方が背景画素の場合，という３つの場合を考える．(a)の場合は，各前景画素が持つ色値による通常の相違度を与える．(b)の場合は，サンプルテクスチャと出力画像の境界形状が合致することに貢献するため，高評価とする固定の相違度を与える．逆に，(c)の場合は，境界形状が合致しないため，低評価とする固定の相違度を与える．このとき，境界形状の合致をより重視するため，(a)によって評価される色の違いよりも(c)によって評価される境界形状の違いのほうが近隣画素群全体の評価に大きな影響を与えるような相違度の大きさとする．

本研究で提案する手法（以下，本手法）では，Ritter らの手法と同様，ユーザがキャンバス上に描いたペイント領域に対してピクセルベースの方法による近隣画素群の評価によって画素単位の合成を行い，サンプル画像から出力画像を生成する．最適な近隣画素群の選択には k-候補探索[4]を用いる．そして，境界を考慮するため，Ritter らの手法と同様，近隣画素群の評価に前景画素と背景画素の区別を導入し，上記の(a)，(b)，(c)の場合に分けて相違度を求める．ただし，本手法は次の点で Ritter らの手法と異なる．

(1) Ritter らの手法による近隣画素群の評価における(c)の場合について，本手法では，複雑な境界形状に安定して対応するため，調整可能な重みを与える．

(2) Ritter らの論文では，近隣画素群の評価において境界までの距離を考慮することに触れてはいるが，詳細な記述がない．一方，本手法では，サンプル画像の境界付近の特徴（色と形状）を出力画像に自然に再現するため，各画素から境界までの距離（「境界距離」）を導入した近隣画素群の評価方法を明確に提案する．これにより，キャンバス上の処理画素がペイント領域の境界に近づくにつれて，サンプル画像上でその境界と類似した向きを持つ境界に対して類似した境界距離を持つ画素が選択され，出力画像に合成される．

(3) Ritter らの手法ではユーザが描いたペイント領域内の画素だけに合成が行われる．そのため，複雑な大きな凹凸を持つ境界形状を再現することが難しい．一方，本手法では，複雑な大きな凹凸を持つ境界形状を再現するため，キャンバス上の処理画素を合成キューに格納して管理することで，ペイント領域の外側に向けて合成される画素領域の境界を拡張させることを可能とする．そして，近隣画素群の評価によって適切な「境界画素」を決定していくことで境界の拡張を停止する．また，境界画素を合成キューに再格納することによる隙間領域の埋め合わせによって，大きな境界拡張に伴って出力画像上に起きやすい不自然なひび割れの発生を軽減する．関連して，キャンバス上の画素の処理順序について，Ritter らの手法はペイント領域内の画素を走査線順に処理する．一方，本手法では，走査線順の他に，より自然な境界の拡張を促進するため，ペイント領域の中央から周囲に向かう順序での処理を可能とする．

(4) Ritter らのシステムでは，ユーザが目的とするサンプルテクスチャを選択してキャンバス上に図形等をペイント描画するが，サンプルテクスチャの色分布を考慮して描画した図形等に色を与えて出力画像の色分布を意図的に制御することは実現されていない．一方，本手法では，ユーザ制御テクスチャ合成を利用し，ユーザが意図的に色を与えて描いたペイント領域を目的画像として扱うことで，ユーザが望む色分布を持つ出力画像を生成する．

3. テクスチャ合成の基本アルゴリズム

ここでは，本研究で提案する境界拡張テクスチャ合成法で必要となる，これまでに提案されてきたテクスチャ合成の基本アルゴリズムについて述べる．本研究の提案手法は，ピクセルベース手法に基づく合成を行う．ピクセルベース手法では，図 2 に示すように，通常，走査線順で出力画像の画素を合成する．出力画像上の処理画素 P_o について，合成済みの M 個の画素

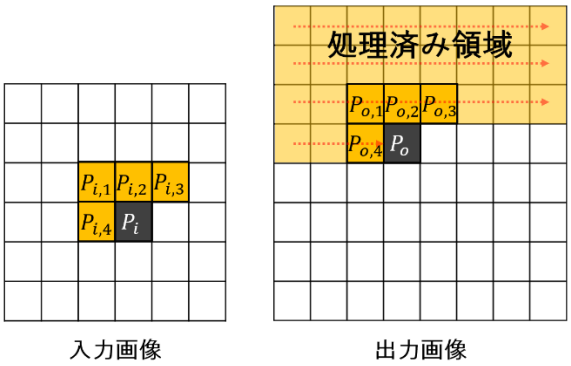


図 2. ピクセルベース手法によるテクスチャ合成

からなる左上 L 型の近隣画素群 N_o を考え，それと最も類似する色パターンを持つ入力画像上の画素 P_i の近隣画素群 N_i を探索し，画素 P_i の色を画素 P_o に合成する．なお，図 2 の近隣画素群は $M = 4$ 個の画素からなるが，任意の近隣画素群のサイズが可能である．２つの近隣画素群の色パターンの類似性は，通常，対応する位置の画素どうしが持つ色値の差の二乗の和（Sum of Squared Difference : SSD）を用いた次式による相違度 F を用いて評価を行う．

$$F = \sum_{m=1}^{M} \left| \boldsymbol{C}_{i,m} - \boldsymbol{C}_{o,m} \right|^2$$

$$= \sum_{m=1}^{M} \left(R_{i,m} - R_{o,m} \right)^2 + \left(G_{i,m} - G_{o,m} \right)^2 + \left(B_{i,m} - B_{o,m} \right)^2 \quad (1)$$

ここで，$\boldsymbol{C}_{i,m} = [R_{i,m}, G_{i,m}, B_{i,m}]$ と $\boldsymbol{C}_{o,m} = [R_{o,m}, G_{o,m}, B_{o,m}]$ は近隣画素群 N_i と N_o 内の対応する画素 $P_{i,m}$ と $P_{o,m}$ の色であり，相違度 F が小さいほど２つの近隣画素群の類似性が高い．最もシンプルな全探索法は，出力画像上の各画素 P_o に対して，入力画像上の全ての画素 P_i に関する評価を行い，最も相違度 F が小さくなる画素を選択する．また，探索を効率化する候補探索法[3]は，出力画像上の処理画素 P_o について，近隣画素群 N_o 内の各画素に合成済みの入力画像上の画素を利用して，入力画像上の隣接する画素群がなるべくまとまって出力画像上に合成されるように，画素 P_o に合成する候補となる入力画像上の画素 P_i を限定する．この方法は，入力画像の色パターンをなるべく維持するように合成を行う結果となり，特に，エッジ成分が多い入力画像に対して良好な出力画像を生成する．さらに，前処理で入力画像上の各近隣画素群について類似した k-1 個の近隣画素群を探索して記憶しておくことで，候補探索法の候補の数を k 倍に増やす k-候補探索法[4]がある．

実例ベーステクスチャ合成の一つであるユーザ制御テクスチャ合成[3]は，ピクセルベースの方法により，図 1 のように，ユーザが描いた目的画像の色分布に合わせて入力画像から出力画像を合成する．目的画像は出力画像と同じ解像度を持つ．図 3 に示すように，出力画像上の処理画素 P_o に隣接する合成済み画素からなる左上 L 型の近隣画素群 N_o に加えて，目的画像上の右下 L 型の近隣画素群 N_u も用い，それらを合わせた正方型

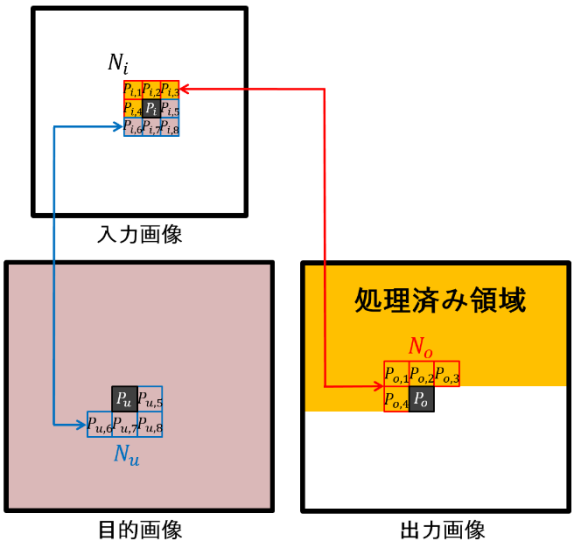


図 3. ユーザ制御テクスチャ合成

の近隣画素群と最も類似する色パターンを持つ入力画像上の画素 P_i の正方型の近隣画素群 N_i を探索し，画素 P_i の色を画素 P_o に合成する．近隣画素群の色パターンの相違度は式(1)で求める．なお，図 3 中の目的画像上の画素 P_u は出力画像上の処理画素 P_o と同じ画素位置であり，式(1)において，目的画像上の近隣画素群 N_u 内の画素 $P_{u,m}$ の色を $\boldsymbol{C}_{o,m}$ とする．図 3 の場合，近隣画素群内の画素数は $M = 8$ である．

4. 境界拡張テクスチャ合成法

4.1 概要

本手法は，ユーザ制御テクスチャ合成に基づき，サンプル画像（入力画像）の特徴的な物体輪郭に合わせて，ユーザが描いたペイント画像（目的画像）上のペイント領域の外側まで合成領域の境界を適切に拡張させるテクスチャ合成法である．近隣画素群の探索には k-候補探索を用いる．そして，境界距離に関する相違度，ならびに，色と前背景画素の区別に関する相違度によって，最適な合成画素を決定する．また，合成キューと境界画素によって，適切な合成領域の境界拡張を行う．

4.2 サンプル画像とペイント画像

はじめに，ユーザが描きたい物体を含むサンプル画像を用意する．そして，その物体が映る画素を前景画素，それ以外の画素を背景画素として分離する．この作業は手動で行ってもよいが，最近の一般のペイントツールでは簡単に前景と背景を分離する機能を持つものもあり，それらを用いることもできる．また，アルファマッティング等のソフトウェアを用いて自動で分離することも可能である．続いて，その物体の配色を考慮した色を用いて，ユーザがキャンバス上にペイントツールによって自由に物体を描いたペイント画像を用意する．このとき，あくまでサンプル画像上の元の物体はサンプルであり，ユーザは元の物体とは異なる外形や大きさで自由に描いてよい．また，色や形状を詳細に描く必要はなく，おおまかなラフな描画でよい

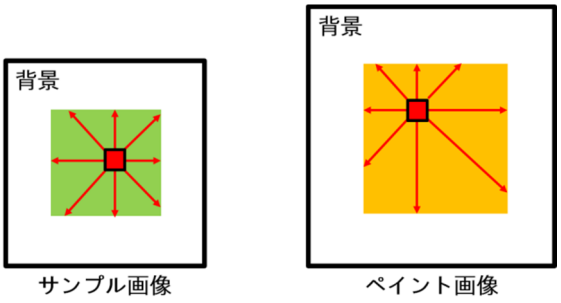


図 4. 境界距離

（図 7 から図 14 の実験例を参照）．ペイント画像上の画素は，ペイントした領域内の画素が前景画素，それ以外の画素が背景画素となる．本手法は，ユーザ制御テクスチャ合成に基づき，サンプル画像は入力画像として，ペイント画像は目的画像として扱われる．これらをもとにして，ペイント画像上のペイント領域の輪郭に合わせてサンプル画像上の元の物体の特徴的な輪郭を適切に再現した出力画像が生成される．

4.3 境界距離

続いて，合成の前処理として，サンプル画像とペイント画像の全ての前景画素について 8 方向の境界距離を求める．これらは，図 4 に示すように，前景画素から上下左右の 4 方向と斜め 45 度の 4 方向を合わせた 8 方向に直線を延ばし，それぞれの方向について境界まで，すなわち，背景画素に達するまでの距離である．ただし，ペイント画像の前景画素の境界距離は後述する合成領域の境界拡張によって動的に変化するため，合成処理の最中にも，適宜，更新する．

4.4 画素の処理順序

合成処理では，ピクセルベースの方法に従い，画素単位の合成を行う．通常のテクスチャ合成，そして，先述の Ritter らの手法では走査線順に処理を行うが，本手法では，後述するように，元のペイント領域の境界を内側から外側へ向けて拡張しながら合成を行うため，走査線順の処理はあまり望ましくない．そこで，ペイント領域の中心付近，具体的には，ユーザが指定した開始位置からペイント領域の境界に向けて内側から外側へ合成処理を行う．この処理を行うため，キャンバス上の画素の処理順序を記憶しておくための合成キューを用意し，合成処理の開始時にペイント領域内の全ての画素を上記の順序で合成キューに格納する．この合成キューは後述する合成領域の境界拡張を実現する役割も果たす．合成キューから格納されている順序で処理画素を取り出して合成処理を行い，合成キューが空になった時点で全体の処理を終了する．

4.5 近隣画素群の探索と評価

キャンバス上の処理画素に合成するサンプル画像上の最適な画素を決定するための近隣画素群の探索には k-候補探索[4]を用いる．k-候補探索では，合成の前処理として，サンプル画像の全ての画素 P_i に対して，式(1)と同様の色値による類似性の評価により，サンプル画像内の他の画素のうちで画素 $P_i = P_i^1$ と近隣画素群の類似性が高い k-1 個の画素 $P_i^2, \cdots, P_i^{\mathrm{k}}$ を探索して対応付け，それらの座標値を記憶しておく．これらの k 個の画素 $P_i^1, \cdots, P_i^{\mathrm{k}}$ を k-候補と呼ぶ．候補の数である k はユーザが任意に決定でき，特に k = 1 のときには k-候補探索は通常の候補探索となる．なお，k-候補はサンプル画像に対して不変であるため，前処理で求めた以降は更新する必要はない．合成処理の際には，キャンバス上の処理画素にサンプル画像上の画素を合成するたびに，合成画素の座標を記憶しておく．そして，合成すべき画素の探索の際には，現在の処理画素 P_o の近隣画素群 N_o 内の各処理済み画素 $P_{o,m}$ に合成された（座標が記憶されている）画素 $Q_{i,m}$ に関連する候補画素 $R_{i,m} = R_{i,m}^1$，ならびに，その各々に対応付けられた候補画素 $R_{i,m}^2, \cdots, R_{i,m}^k$ からなる k-候補のみを探索の対象とする[4]．このとき，合成処理が進むにつれてサンプル画像上で選択される合成画素が境界に近付いていくことで，候補画素が背景画素となる場合がある．この場合にも，後述するように，その背景画素の近隣画素群が最類似となる場合には「境界画素」として合成するため，除外せずに候補画素として残す．

ペイント画像の色分布に合わせた出力画像を生成するため，近隣画素群の評価には，サンプル画像を入力画像，ペイント画像を目的画像として用いることで，ユーザ制御テクスチャ合成を行う．ただし，画素の処理順序が走査線順ではないため，一般に，正方形を構成する出力画像とペイント画像（目的画像）の近隣画素群のそれぞれの形状が図 3 に示すような L 型ではなくなる．画素の処理順序に依存して，キャンバス上の現在の処理画素を囲む正方型の近隣画素群内で，処理済み画素は出力画像に属するものとし，その他の画素はペイント画像に属するものとする．なお，説明の都合上，出力画像とペイント画像を分けているが，実際の実装ではキャンバス上のペイント画像に合成画素を上書きして出力画像を生成していく．

相違度の算出には，式(1)による色に関する相違度に加えて，境界付近の特徴を再現するために，境界距離，ならびに，前景画素と背景画素の区別を利用する．まず，キャンバス上の処理画素がペイント領域の境界に近づくにつれて，サンプル画像上でその境界と類似した向きを持つ境界に対して類似した境界距離を持つ画素が選択されるようにする．これを実現するため，境界距離の閾値 D_{max} を設定する．そして，処理画素の 8 方向の境界距離 D_t，$t = 1, \cdots, 8$，のうちで $D_t \le D_{max}$ の条件を満たす方向だけが境界に近い方向であるとみなし，境界を考慮することにする．さらに，相違度の計算に含める境界距離の最大数 T_{max} を設定し，上記の条件を満たす方向のうち，境界距離が小さい方から最大で T_{max} 個の方向を選択する．選択された方向 t の集合を S とすると，境界距離に関する相違度は次式となる．

$$F_d = \begin{cases} \dfrac{1}{|S|}\displaystyle\sum_{t \in S} D_t & |S| > 1 \text{ のとき} \\ 0 & |S| = 0 \text{ のとき} \end{cases} \qquad (2)$$

ここで，値 $|S| \le T_{max}$ は集合 S に含まれる方向の数である．

表 1. 対応画素間の相違度 F_m

		サンプル画像（入力画像）	
		前景	背景
出力画像・ペイント画像（目的画像）	前景	$\left\|\boldsymbol{C}_{i,m}-\boldsymbol{C}_{o,m}\right\|^2$	W_1F_{max}
	背景	W_1F_{max}	0

一方，式(1)による色に関する相違度であるが，本手法はサンプル画像上の前景画素だけを合成の対象とするため，色が利用できるのは前景画素だけであり，背景画素の色は利用できない．そのため，Ritter らの手法[21]に類似した前景画素と背景画素の区別を導入する．キャンバス上の処理画素を囲む正方型の近隣画素群が M 個の画素を含むものとする．上述のように，これらは出力画像に属する処理済み画素とペイント画像に属する未処理画素からなる．これらをサンプル画像上で探索する正方型の近隣画素群内の M 個の画素と比較して類似性を評価する相違度の式を次に示す．

$$F_c=\frac{1}{M}\sum_{m=1}^{M}F_m \tag{3}$$

値 F_m は m 番目の対応する画素間の相違度であり，サンプル画像と出力画像・ペイント画像の間の対応する画素が前景画素と背景画素のいずれの組み合わせかによって，表 1 に示す値が与えられる．まず，(a) 両方とも前景画素の場合，式(1)と同様，通常の色値による評価を行う．表 1 中の $\boldsymbol{C}_{i,m}$ はサンプル画像の前景画素 $P_{i,m}$ の色であり，$\boldsymbol{C}_{o,m}$ は出力画像・ペイント画像の前景画素 $P_{o,m}$ の色である．次に，(b) 両方とも背景画素の場合には，サンプル画像と出力画像・ペイント画像の境界形状が合致することに貢献するため，高評価とする．この場合について，Ritter らの手法では相違度 0 を与えていたが，本手法でも相違度 0 を与えることとする．最後に，(c) 一方が前景画素で他方が背景画素の場合，境界形状の合致に反するため，低評価とする．この場合，Ritter らの手法では，境界形状の合致をより重視するため，(a) の色の評価よりも (c) の境界形状の評価のほうが近隣画素群全体の評価に大きな影響を与えるように大きな固定の相違度を与えていた．本手法でも大きな相違度を与えることとするが，過度な制約は軽減して形状の多少の違いを許容するほうが安定した良好な結果が得られる場合もあると考えられるため，境界形状の複雑さや特徴に応じて任意の相違度に調整できるようにする．具体的には，色値の差の二乗 $|\boldsymbol{C}_{i,m}-\boldsymbol{C}_{o,m}|^2$ の最大値 F_{max} に対して調整可能な重み W_1 をかけた値とする．色 $\boldsymbol{C}=[R,G,B]$ の各要素が $0\le R,G,B\le 255$ の値をとる場合，最大値 $F_{max}=3\cdot 255^2$ とする．最終的な相違度は，式(2)と式(3)を用いて，以下のようになる．

$$F_{all}=a\,F_d+(1-a)\,F_c \tag{4}$$

係数 a は境界距離に関する相違度 F_d と色と前背景画素の区別に関する相違度 F_c の比率を調整する重みである．そして，k-候補探索の候補画素のうち，相違度 F_{all} が最小となるものを選択して処理画素に合成する．先述のように，候補画素が背景画素となる場合があるが，その背景画素の相違度が最小となる場合には，処理画素に背景画素を合成する．すなわち，その処理画素には色を与えず，背景画素であるという属性だけを与える．この合成された背景画素を「境界画素」と呼ぶ (図 5 参照)．境界画素の合成は，境界付近の近隣画素群の評価の結果として，ペイント領域の輪郭に合わせてサンプル画像上の物体輪郭を最も適切に再現するように選択されたものである．これは本手法にとって重要な効果をもたらす．

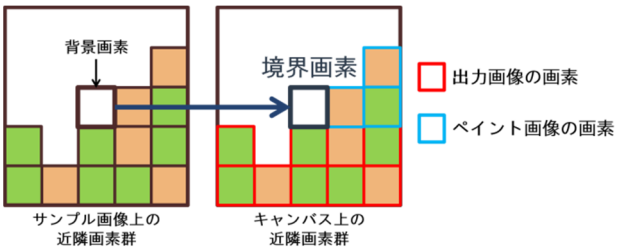


図 5. 境界画素の合成

4.6 合成領域の境界拡張

本手法では，ペイント領域の輪郭に合わせてサンプル画像上の物体の特徴的な輪郭を適切に再現した出力画像を生成する．特に，ペイント領域の外側に向けて画素が合成される領域の境界を適切に拡張することで，複雑な大きな凹凸を持つ境界形状の再現を可能とする．この実現のため，キャンバス上の現在の処理画素への合成を終えた時点で，その処理画素の上下左右の 4 近傍の画素のうちでペイント領域外のもののうち，未処理で合成キューに格納されていないものを合成キューに格納することで合成領域を拡張する．ただし，近隣画素群の評価の結果として，その処理画素に合成された画素が境界画素（背景画素）である場合（図 5 参照）には，この 4 近傍の画素の合成キューへの格納は行わず，合成領域の拡張を止める．つまり，前景画素が合成された場合のみ，合成領域の拡張を行う．これにより，複雑な境界形状の場合でも，サンプル画像の前景画素と背景画素の境界に合わせた適切な合成領域の拡張が行われる．

合成領域の境界の拡張による一つの問題として，２つの境界が境界画素群を挟んで接してしまう状況，すなわち，ある境界の大きな拡張が他の部分のすでに合成済みの境界画素群に外側から達してしまう状況が起きうる．この場合，結果として，その境界画素群が出力画像上でひび割れのような隙間を生じてしまう．そこで，合成領域の拡張のために処理画素の 4 近傍の画素を合成キューに格納するかどうかを判定する際，合成済みの境界画素がある場合には，その画素を合成キューに再格納し，全ての合成処理が終了した後，再度，近隣画素群の評価による合成処理を行うこととする．これにより，通常，前景画素が合成されることで隙間が埋められ，ひび割れが補完される (図 6 参照)．

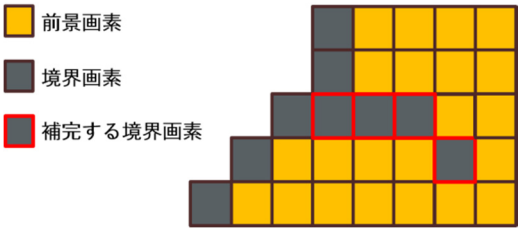


図 6. ひび割れの補完

5. 実験

複雑な凹凸を持つ特徴的な物体輪郭を含む様々なサンプル画像に対して本手法を適用した実験結果を図 7 から図 14 に示す. 各図中のサンプル画像はペイントツール（Adobe Photoshop CC）を使って元の画像から手動で物体領域を切り抜いて作成した. それぞれ，適切な近隣画素群サイズと k-候補数を用い，式(4)の係数 a は 0.5 とした．そして，調整可能な重み W_1 の値を変えて実験を行った. 使用した PC は, CPU: Intel Core i7-4790K, メモリ: 8.00 GB，OS: Windows 8.1 Pro である．各図中に処理時間（秒）も示す．実験結果から，ユーザがおおまかに描いたペイント画像に対して，その色分布を反映した出力画像が生成されていることが分かる．また，ペイント領域の境界の拡張により，サンプル画像上の複雑な大きな凹凸を持つ境界形状が出力画像上に適切に再現されており, 本手法の有効性が確認できる.

5.1 重み W_1 の効果

実験結果から, 重み W_1 の値を大きくするほど大きな境界拡張が生じ，出力画像上の境界形状がサンプル画像上の境界形状に類似していくことが分かる．そして，大きな W_1 に対しては過度な境界拡張が生じている．逆に，小さな W_1 に対しては境界拡張が小さく，出力画像上の境界形状がサンプル画像上の境界形状に至る前に拡張が止まってしまう箇所が見られる．これは，本手法の近隣画素群の評価方法によると考えられる.

この評価方法では，サンプル画像と出力画像・ペイント画像の間の対応画素について，一方が前景画素で他方が背景画素の場合，境界形状の合致に反するため，低評価のペナルティとして（基本的には大きな）相違度 W_1F_{max} を与える．また，背景画素どうしが対応する場合には, 高評価の相違度 0 を与える. ここで，全体の相違度をなるべく小さくするという観点から，ペイント領域の境界付近の近隣画素群 N_o に対してどのようなサンプル画像上の近隣画素群 N_i が選択される傾向にあるか，それぞれの近隣画素群中の対応画素 $P_{o,m} \in N_o$ と $P_{i,m} \in N_i$ の関係を考える.

まず，ペイント領域内の前景画素 $P_{o,m}$ に対しては，画素 $P_{i,m}$ は，背景画素で相違度 W_1F_{max} を得るよりも，なるべく色パターンが類似して相違度を小さくするような前景画素であるほうが望ましい.

一方，ペイント領域外の背景画素 $P_{o,m}$ に対しては，画素 $P_{i,m}$ は背景画素で相違度 0 を得るのが望ましい. しかし, 特に, サンプル画像上の物体輪郭が複雑な場合，上記のペイント領域内での色パターンの類似という条件を満たしながらペイント領域外では全ての画素 $P_{i,m}$ が背景画素となるような近隣画素群 N_i は希有である．ここで，重み W_1 が小さい場合には，背景画素 $P_{o,m}$ に対する画素 $P_{i,m}$ が前景画素であっても相違度 W_1F_{max} が大きなペナルティにはならない．よって，ペイント領域外の背景画素群 $\{P_{o,m}\}$ に対して，前景画素と背景画素の対応はあまり重視せず，なるべく多くの背景画素を含む画素群 $\{P_{i,m}\}$ を持つ近隣画素群 N_i が選択される傾向となり，結果的に境界拡張が抑えられる．一方，重み W_1 が大きい場合には，ペイント領域外の背景画素 $P_{o,m}$ に対する前景画素 $P_{i,m}$ による相違度 W_1F_{max} が大きなペナルティとなる．よって，可能な限り，対応画素 $P_{o,m}$ と $P_{i,m}$ が前景画素どうし，背景画素どうしのいずれかとなるような近隣画素群 N_i が選択される傾向となる．これは，出力画像上の境界形状を可能な限りサンプル画像上の境界形状に近付けるように境界拡張を進めることにつながる．重み W_1 が大きくなるにつれて，より正確な物体輪郭を再現しようとするため，境界拡張が停止せず，過度な拡張を生じることになる.

以上から, 重み W_1 は境界の拡張度を制御する役割を果たすものと言える．そして，図 7 から図 14 の比較から，サンプル画像上の物体輪郭の特徴によって適切な W_1 の値が異なることが分かる．この拡張度の調節も本手法の有用な機能の一つと考えられる.

5.2 ペイント画像とサンプル画像の境界色の不整合

図 11 から図 14 の実験では，ペイント画像とサンプル画像を比較した場合，両画像上で互いに類似した向きの境界部分どうしが必ずしも類似した色を持たない．言い換えれば，サンプル画像上の物体の色について，境界部分ごとの色分布までは考慮せず，サンプル画像上の物体が全体的に持つ色を意図したペイント色で自由にペイント画像を描画している．例えば，図 11 では，ペイント画像上のペイント領域の右端の境界部分は葉を意図した暗色で描かれているが，一方，サンプル画像上では右端の境界部分には葉は存在せずに紫色の花だけが存在する．このような場合，本手法によって生成される出力画像では，境界から離れたペイント領域の内部にはペイント色に合わせたサンプル画像上の物体色が合成されるが，ペイント領域の境界部分にはサンプル画像上の類似した向きの境界部分の物体色が優先的に合成され，その色は必ずしもペイント色と合致するわけではない．例えば，図 11 の出力画像では，中央から右側に向かってペイント領域の暗色に合わせて葉が合成されていくが，境界付近になるとサンプル画像の右端の境界部分を再現するように花が合成されている. 図 12, 13, 14 でも同様の現象が見られる. 図 12 では, サンプル画像上の物体の境界部分はどの向きでも黄色の花であるため，ペイント画像上のペイント領域の境界部分のうちで葉を意図した緑色の部分に対しても葉は合成されず，全ての境界部分に花が合成されている．図 13 と図 14 についても，ペイント領域の内部ではペイント色に合うように物体色が合成されているが，境界部分ではペイント色と物体色が必ずしも合致せず，それぞれの向きの境界部分をサンプル画像に合わ

せて再現することが優先されている．

5.3 重み a の効果

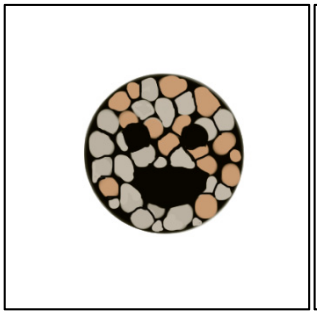
図 15 は，式(4)による相違度 F_{all} を構成する２つの相違度 F_d と F_c の効果を検証するため，それらの比率を調整する重み a の値を変化させた結果である．図 7 のサンプル画像を用い，W_1 を 0.8 とした．この結果から，以下のことが分かる．重み a の値が小さい場合，すなわち，色と前背景画素の区別に関する相違度 F_c の影響が大きい場合には，より明瞭にペイント画像上のペイント色に合わせてサンプル画像上の物体色が合成されている（ペイント画像の中央の３つの黒色の部分に注目）が，一方，境界部分の再現性が低い．逆に，重み a の値が大きい場合，すなわち，境界距離に関する相違度 F_d の影響が大きい場合には，ペイント色に対して合成される物体色の合致性が低くなるが，一方，境界部分の再現性は高い．これは，相違度 F_d による境界距離の評価によって，ペイント領域の境界部分に対して，より類似した境界距離を持つサンプル画像上の境界部分の画素が選択される効果によるものである．相違度 F_c も前背景画素の区別による評価によって境界部分を再現する効果を持つが，ペイント領域の各境界部分に対してサンプル画像上の類似した向きの境界部分の画素を積極的に選択することはないため，相違度 F_d に比べて境界部分の再現を促進する働きは低い．２つの相違度 F_d と F_c の比率を調整する重み a は合成色の合致性と境界部分の再現性を制御する役割を果たすものと言える．

5.4 ひび割れの補完

図 16 は, 境界画素によって発生するひび割れを補完することによる効果を示す（図 6 参照）．図 7 の出力画像(W_1 ＝1.0)について，ひび割れの補完の有無による結果を示す．右上の出力画像上の境界画素を示す図では，青色が最終的な境界画素，赤色がひび割れの補完のために境界画素に再合成を行った画素である．これらにより，元のペイント領域から境界画素が合成される位置まで境界の拡張が行われていること，ひび割れの原因となる合成領域内部の境界画素が適切に補完できていることが分かる．

6. おわりに

本研究では，ユーザが描いたペイント画像に合わせてサンプル画像上の物体の色分布と特徴的な輪郭を再現した出力画像を生成する境界拡張テクスチャ合成法を提案した．提案手法は，複雑な大きな凹凸を持つ輪郭に対応でき，調整可能な重みによる拡張度の制御が可能である．今後の課題として処理速度の向上が挙げられる．現状で利用している k-候補探索は候補画素を探索する前処理に時間を要する．また，動的に変化する境界距離の再計算の処理時間が大きい．そこで，random walk 法の利用など, より効率的な探索方法を検討する必要がある. そして，ユーザのペイント描画に対してリアルタイムで合成処理を行うインタラクティブシステムの実現を検討したい．また，別の課題として，アルファマッティングの利用がある．特に，複雑な境界形状に対しては，アルファ値を考慮することで高品質な画像の表現が可能となるため，現在，サンプル画像上の各画素のアルファ値も含めた新たな評価方法によるテクスチャ合成法を検討している．


ペイント画像


出力画像 (W_1 ＝0.5) 16.51 秒

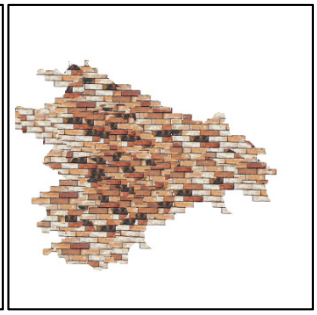

出力画像 (W_1 ＝0.8) 18.83 秒

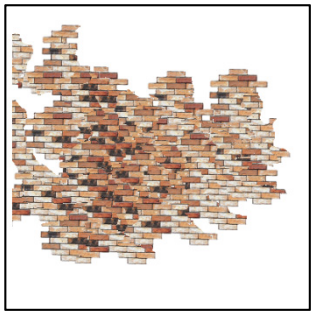

出力画像 (W_1 ＝1.0) 38.49 秒

図 7. 実験結果 1
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k＝3


サンプル画像
（石）


ペイント画像


出力画像 (W_1 ＝0.3) 20.21 秒


出力画像 (W_1 ＝0.5) 31.01 秒

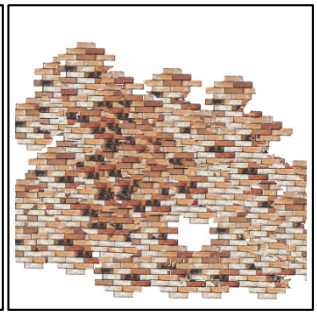
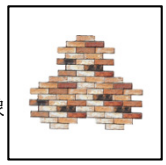
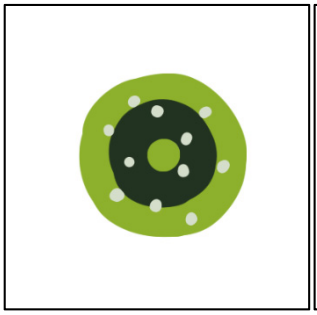
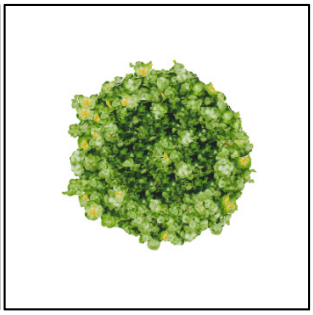
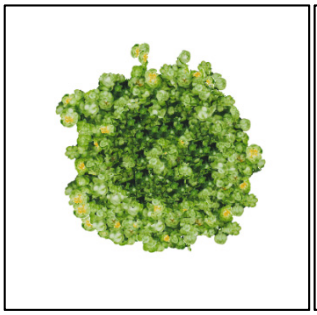

出力画像 (W_1 ＝0.8) 42.58 秒

図 9. 実験結果 3
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k＝1


サンプル画像
（レンガ）


ペイント画像


出力画像 (W_1 ＝0.5) 19.76 秒


出力画像 (W_1 ＝0.8) 23.58 秒

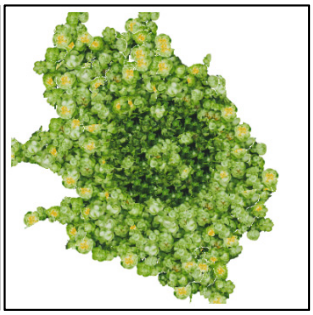
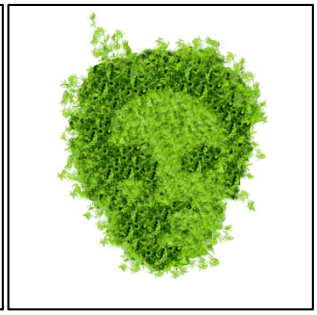
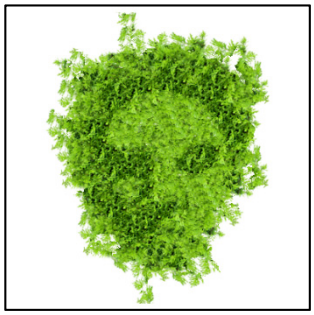

出力画像 (W_1 ＝1.0) 39.45 秒

図 8. 実験結果 2
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k＝5


サンプル画像
（花と葉）


ペイント画像


出力画像 (W_1 ＝0.5) 27.79 秒


出力画像 (W_1 ＝0.8) 33.21 秒

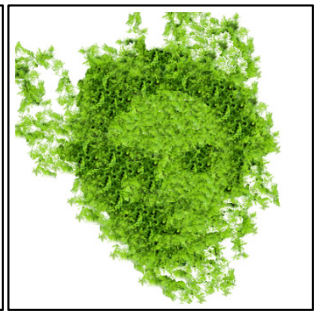
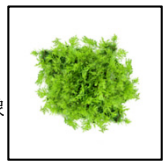
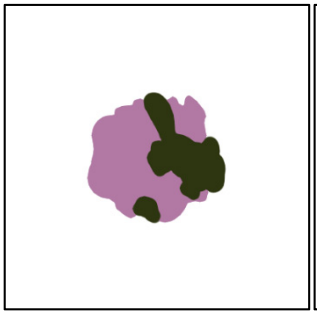
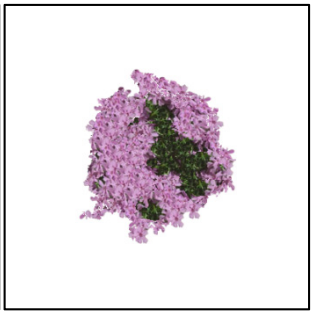
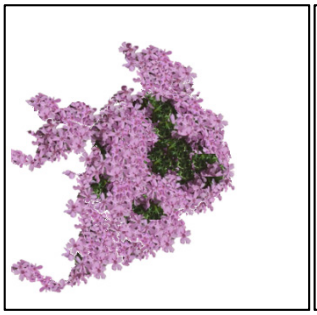

出力画像 (W_1 ＝1.0) 40.67 秒

図 10. 実験結果 4
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k＝3


サンプル画像
（葉）


ペイント画像


出力画像 (W_1 = 0.3) 13.87 秒


出力画像 (W_1 = 0.5) 24.08 秒

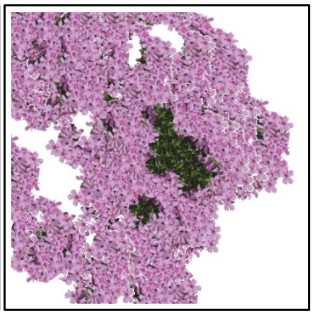
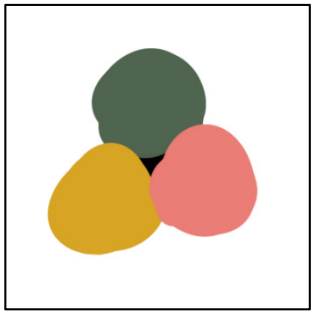
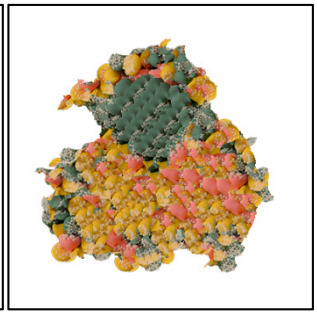

出力画像 (W_1 = 0.8) 53.23 秒

図 11. 実験結果 5
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k = 5


サンプル画像
（花と葉）


ペイント画像


出力画像 (W_1 = 0.5) 25.88 秒


出力画像 (W_1 = 0.8) 36.52 秒

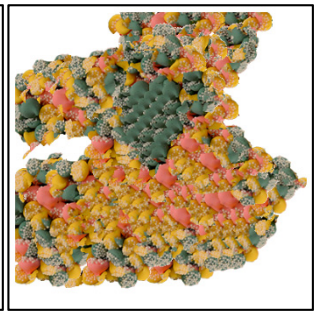
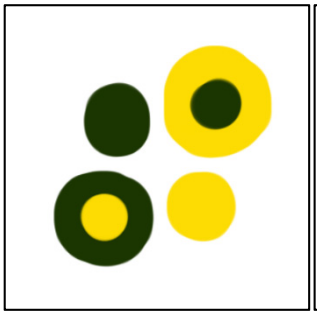
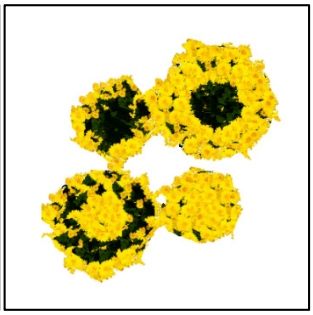
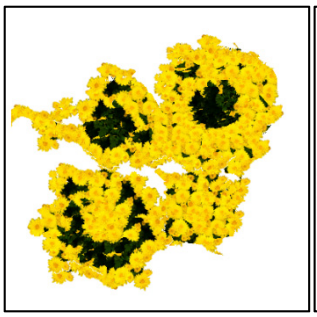

出力画像 (W_1 = 1.0) 45.85 秒

図 13. 実験結果 7
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k = 1


サンプル画像
（菓子）


ペイント画像


出力画像 (W_1 = 0.5) 22.64 秒


出力画像 (W_1 = 0.8) 29.90 秒

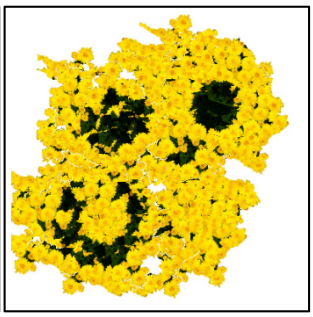
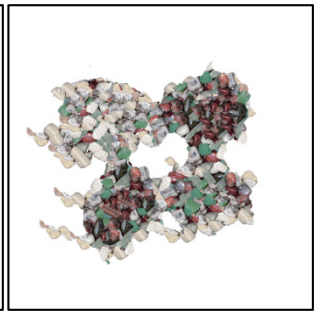
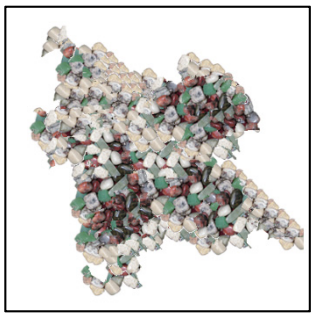

出力画像 (W_1 = 1.0) 40.45 秒

図 12. 実験結果 6
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k = 1


サンプル画像
（花と葉）


ペイント画像


出力画像 (W_1 = 0.3) 20.52 秒


出力画像 (W_1 = 0.5) 33.36 秒

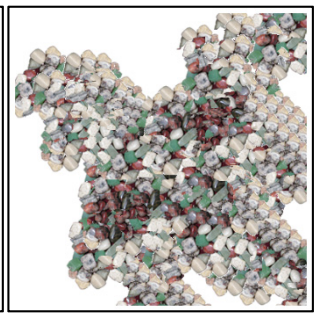
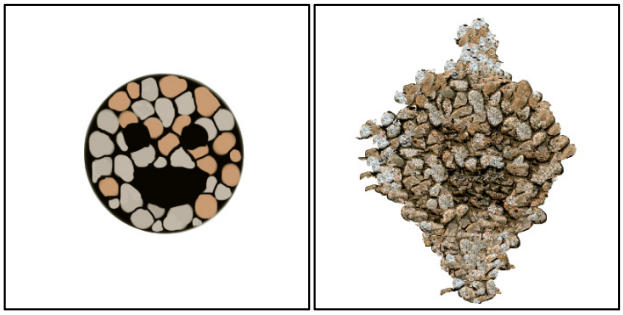

出力画像 (W_1 = 0.8) 49.11 秒

図 14. 実験結果 8
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k = 3


サンプル画像
（石）

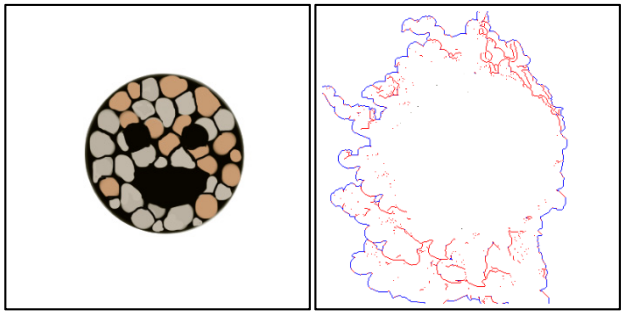


ペイント画像　出力画像（α ＝0.1）24.58 秒

出力画像（α ＝0.5）20.23 秒　出力画像（α ＝0.9）20.19 秒

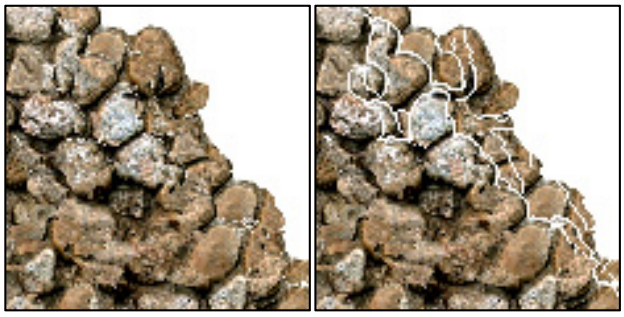
図 15. 重み α の効果
サンプル画像：200x200 画素
ペイント画像：400x400 画素
近隣画素群サイズ：15x15 画素
k-候補数：k＝3

サンプル画像（右）



ペイント画像　出力画像上の境界画素

補完あり出力画像　補完なし出力画像



補完あり出力画像の右上部分　補完なし出力画像の右上部分

図 16. ひび割れの補完の効果

サンプル画像（右）

参考文献

[1] A. A. Efros and T. K. Leung, Texture Synthesis by Non-parametric Sampling, Proc. of IEEE International Conference on Computer Vision 1999, vol. 2, pp. 1033-1038, 1999.

[2] L. Wei and M. Levoy, Fast Texture Synthesis using Tree-Structured Vector Quantization, Proc. of ACM SIGGRAPH 2000, pp. 479-488, 2000.

[3] M. Ashikhmin, Synthesizing Natural Textures, Proc. of ACM Symposium on Interactive 3D Graphics 2001, pp. 217-226, 2001.

[4] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, H. Shum, Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces, Proc. of ACM SIGGRAPH 2002, pp. 665-672, 2002.

[5] B. Guo, H. Shum and Y. Xu, Chaos Mosaic: Fast and Memory Efficient Texture Synthesis, Microsoft Research TechReport, MSR-TR-2000-32, 2000.

[6] A. A. Efros and W. T. Freeman, Image Quilting for Texture Synthesis and Transfer, Proc. of ACM SIGGRAPH 2001, pp. 341-346, 2001.

[7] L. Liang, C. Liu, Y. Xu, B. Guo and H. Shum, Real-time Texture Synthesis by Patch-based Sampling, ACM Transactions on Graphics, vol. 20, no. 3, pp. 127-150, 2001.

[8] V. Kwatra, A. Schodl, I. Essa, G. Turk and A. Bobick, Graphcut Textures: Image and Video Synthesis Using Graph Cuts, Proc. of ACM SIGGRAPH 2003, pp. 277-286, 2003.

[9] Q. Wu and Y. Yu, Feature Matching and Deformation for Texture Synthesis, Proc. of ACM SIGGRAPH 2004, pp. 364-367, 2004.

[10] C. Wu, Y. Lai and W. Tai, A Hybrid-based Texture Synthesis Approach, The Visual Computer, vol. 20, no. 2-3, pp. 106-129, 2004.

[11] V. Kwatra, I. Essa, A. Bobick and N. Kwatra, Texture Optimization for Example-based Synthesis, Proc. of ACM SIGGRAPH 2005, pp. 795-802, 2005.

[12] S. Lefebvre and H. Hoppe, Parallel Controllable Texture Synthesis, Proc. of ACM SIGGRAPH 2005, pp. 777-786, 2005.

[13] S. Lefebvre and H. Hoppe, Appearance-space Texture Synthesis, Proc. of ACM SIGGRAPH 2006, pp. 541-548, 2006.

[14] C. Han, E. Risser, R. Ramamoorthi and E. Grinspun, Multiscale Texture Synthesis, Proc. of ACM SIGGRAPH 2008, pp. 51(1)-(8), 2008.

[15] W. Dong, N. Zhou and J. Paul, Perspective-aware Texture Analysis and Synthesis, The Visual Computer, vol. 24, pp. 515-523, 2008.

[16] C. Barnes, E. Shechtman, A. Finkelstein and D. B Goldman, PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing, Proc. of ACM SIGGRAPH 2009, pp. 24(1)-(12), 2009.

[17] P. P. Busto, C. Eisenacher, S. Lefebvre and M. Stamminger, Instant Texture Synthesis by Numbers, Vision, Modeling and Visualization, pp. 81-85, 2010.

[18] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless and D. H. Salesin, Image Analogies, Proc. of ACM SIGGRAPH 2001, pp. 327-340, 2001.

[19] J. Fan, X. Shi, Z. Zhou and Y. Tang, A Fast Texture-by-Numbers Synthesis Method based on Texture Optimization, Proc. of the 5th International Symposium on Visual Information Communication and Interaction, pp. 1-10, 2012.

[20] S. Darabi, E. Shechtman, C. Barnes, D. B Goldman and P. Sen, Image Melding: Combining Inconsistent Images using Patch-based Synthesis, Proc. of ACM SIGGRAPH 2012, pp. 82(1)-(10), 2012.

[21] L. Ritter, W. Li, B. Curless, M. Agrawala and D. Salesin, Painting with Texture, Proc. of the 17th Eurographics Conf. on Rendering Techniques, pp. 371-376, 2006.

[22] M. Lukac, J. Fiser, J. Bazin, O. Jamriska, A. Sorkine-Hornung and D. Sykora, Painting by Feature: Texture Boundaries for Example-based Image Creation, Proc. of ACM SIGGRAPH 2013, pp. 116(1)-(8), 2013.

[23] 越後谷勇介，横山俊希，藤本忠博，物体輪郭を考慮した境界拡張テクスチャ合成法，NICOGRAPH 2015，F-04，pp. 1-8，2015.

越後谷 勇介

2015年岩手大学工学部卒業．現在，同大学大学院工学研究科博士前期課程．知的映像編集に関する研究に従事.

横山 俊希

2012年岩手大学工学部卒業．2014年同大学大学院工学研究科博士前期課程修了．在学中は知的映像編集に関する研究に従事．現在，(株)NTTデータ・フィナンシャルコア勤務.

藤本 忠博

1990年慶應義塾大学理工学部卒業．1992年同大学大学院理工学研究科前期博士課程修了．同年(株)三菱総合研究所入社．1995年同研究所退職．同年慶應義塾大学大学院理工学研究科後期博士課程入学. 1999年同大学院単位取得退学. 同年岩手大学助手. 2000年博士(工学)(慶應義塾大学)．2002年岩手大学講師．2005年助教授．2007年准教授．2016年教授．コンピュータグラフィックス, コンピュータビジョン, 画像処理に関する研究に従事．ACM，IEEE，芸術科学会，他会員.