テクスチャ合成によるオブジェクト除去

岡部 龍太郎† 服部 泰造† 千種 康民‡
東京国際大学 商学部情報システム学科† 東京工科大学‡

1 概要

本研究では，静止画像中の選択したオブジェクトをパターンマッチングを用いたテクスチャ合成により穴埋めをする．選択された領域の周囲の情報を参照しパターンマッチングにより同一画像内の似た領域を見つける．そこで一番誤差の小さかった領域の中心点の画素値で1ピクセルずつ順次穴埋めを行う．結果的に，周辺領域とよく似たテクスチャを合成するとともに境界を自然につなぐことができる．

本研究では，この一連の流れをユーザによる領域選択を除いて自動化し負担を軽減させること，テクスチャ合成の精度を上げることを目的とした．

2 研究背景

今日，デジタルカメラや携帯電話のカメラ機能の普及によりデジタルデータとして写真を保存する機会が多くなってきた．また，それらの画像をただ印刷するだけではなく加工を行ってから印刷をすることが多くなってきている．本研究で行う画像内の不要物除去は最もニーズの高い機能の一つである．しかし，現在提案されている手法の多くは手作業で塗りつぶしていく手法であり，ユーザの時間，体力，技術等の負担が大きく，非常に生産効率が悪いといえる．

そこで，本研究ではユーザが対象範囲を選択するだけで自動的に対象範囲を周囲と同化させることを目的とした．

3 提案手法の概要

本研究では，テンプレートマッチングを用いて穴埋めを行う点と周囲が似ている点を探し出し，その点の要素で穴埋めを行った．

Object Elimination by Texture Synthesis
Ryoutarou OKABE†, Taizoh HATTORI† and
Yasutami CHIGUSA‡
†Tokyo International University,
‡Tokyo University of Technology
E-mail zohzemi@tiu.ac.jp
URL http://www.tiu.ac.jp/~zohzemi/

3.1 テンプレートマッチングとは

特定形状の検索の代表的手法で，図1で示すように，テンプレート画像（以下パッチと呼ぶ）T を入力画像 F の探索領域内で動かし，最もよいマッチングの取れる場所を探し出すという処理である．テンプレート画像と入力画像との画素毎の差（残差，以降誤差と呼ぶ）がマッチングの尺度となる．[1]

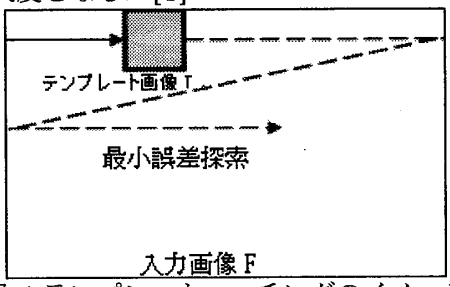


図 1テンプレートマッチングのイメージ

3.2 本研究の特徴

本研究の特徴として，既存のものと比べ対象を消す際のクローン部分をユーザが指定する必要がないので負担が大幅に軽減されている．また，ユーザが消したい範囲を指定しさえすればいいので技術的な負担も大幅に軽減されている．また，これを実現するために本研究では穴埋めの順序，パッチの形状，残差計算式において工夫をすることでテクスチャ合成の精度を上げることができた．

穴埋めを行う順序を縦横方向だけでなく塗りつぶした範囲を周囲から穴埋めを行う方法（輪郭追跡[2]）を加え，連続したパターンを出現しにくくした．

また，次に人工物の多くが直線で構成されていることから，不要と思われる四隅を除くことでマッチングの精度を上げることができた．

また，残差計算式においてただ平均を計算したのではマッチングに利用できる点(x)が少なくても最小であればその点を採用してしまう，しかし利用できる点が少ないということは多い場合に比べ，精度が低くなる要因になる，よって表1のように利用できる点の数を計算に加味す

ることで対応した.

x	分子	平均	残差
2	10	5	10
3	10	3.33	5
4	12	3	4
8	24	3	3.42

表 1　分母の-1 の必要性

3.3 残差計算式

3.1 の残差の計算式として式 1 を考案した.

式 1 の分子では，誤差計算を行っており，具体的には，w をパッチのサイズ（w*2+1 がテンプレート画像 T）とし，入力画像 F の(x,y)を中心としたパッチとの誤差を計算している．p はそれぞれの画素の RGB プレーンである．M ではその画素がユーザの選択した部分かどうかを 1 か 0（0 はユーザが選択した点）で判断し，それをかけることでそれを誤差として含むかどうかを判断している．分母部分では，塗りつぶし部分を含めない個数で割ることで平均をとるようにしている．また，そこで-1 をしているが，3.2 で挙げたようにマッチングに利用できる画素数の重要度を上げるために行っている.

$$E(x,y,u,v)=\frac{\sum_{i=-w}^{+w}\sum_{j=-w}^{+w}\left\{\sum_{p=0}^{2}[I(x+i,y+j,p)-I(u+i,v+j,p)]^2\right\}M(x+i,y+j)M(u+i,v+j)}{\sum_{i=-w}^{+w}\sum_{j=-w}^{+w}M(x+i,y+j)M(u+i,v+j)}-1$$

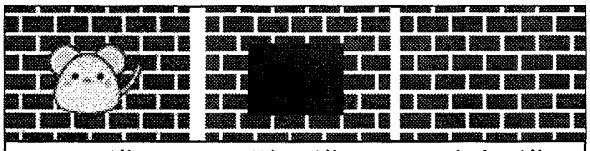
式 1　残差計算式

4　実験例

処理手順として，図 2 の対象となる画像(a)を選択し，(b)のように塗りつぶし処理を実行すると自動で(c)のような結果を出力する.

また，結果の画像に不満があれば，パッチサイズ等を手動で変更することでよりユーザの目的にあった出力を自動で行う.

以下の実験結果は輪郭追跡を用いて，パッチの形状をひし形，パッチのサイズを 10 とした.



(a)元画像　(b)選択画像　(c)出力画像

図 2 処理手順例

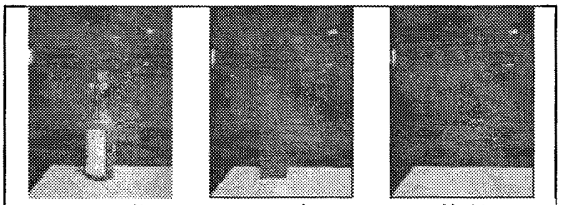
上記の実験例はテクスチャの頻度，形状共に一定で最も簡単な例である．次にテクスチャの頻度，形状共にランダムな場合の例を図 3 であげる．(a)が対象画像，(b)が処理後画像である.

(a)元画像　(b)出力画像

図 3 ランダムな場合の例

次に，一度ではうまくいかないが複数回実行することで自然な結果を得られる例を挙げる.

図 4 では対象画像(a)に対して一度処理をかけると(b)のようになる．その後(b)に対して選択範囲を一度目と変えて再度実行すると(c)のようになる．このように，一度の処理で消えなくても実行結果に対して再度選択範囲等を変えて実行することで適切な結果を得ることができる.



(a)元画像　(b)一度目　(c)複数回

図 4　複数回処理をかけると適切になる例

5　まとめ

本研究を用いることで，ユーザの手間を減らし，なおかつ高い精度でテクスチャを合成し，オブジェクトを除去することができた.

今後の課題は，システムの高速化とさらなる精度の向上である.

また，本研究を応用することで近年よく利用されるワイヤーアクション等で映りこむワイヤーや,経年劣化による映像のノイズ除去などで利用することができるだろう.

参考文献

[1] 伊東敏夫:“ VB.NET で学ぶ 画像処理アルゴリズム”,広文社，2002 年

[2] 酒井 理雄:“ C 言語による輪郭追跡について”,
http://homepage2.nifty.com/tsugu/sotuken/binedge/