デジタルメディア処理2

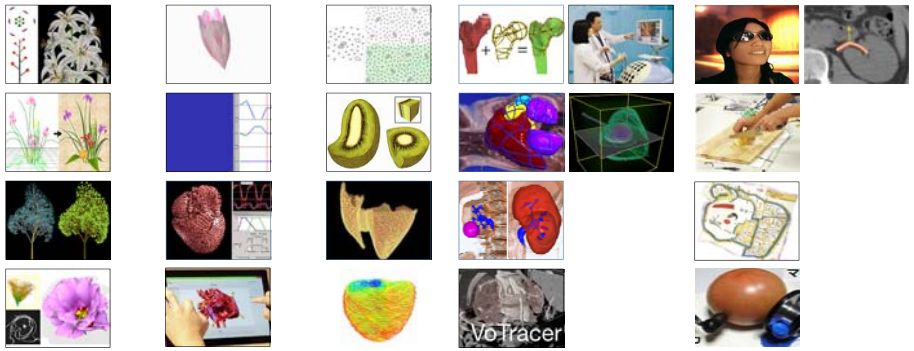
担当: 井尻 敬

井尻敬 – takashiijiri.com

2017 - 現在 ：芝浦工大 准教授
2017 - 現在 ：慶應義塾大学 SFC 客員研究員
2016 - 現在 ：国立循環器病研究センター 客員研究員
2015 - 現在 ：理化学研究所 客員研究員
2015 – 2017：立命館大学 講師
2013 – 2016：北海道大学 客員准教授
2009 – 2015：理化学研究所 研究員
2004 – 2009：東京大学 修士/博士
2000 – 2009：東京工業大学 学士

井尻敬 – takashiijiri.com

専門：Computer Graphics / 画像処理 / ユーザインタフェース



モデリング　アニメーション　テクスチャ　領域分割&応用　その他

○ 講義の概要:

画像処理は，産業・自然科学・エンタテインメントなど，多種多様な分野の発展に関わる非常に重要な技術です．デジタルメディア処理１では，画像処理の基本である，画像データ構造・画像撮影方法・線形フィルタ・非線形フィルタ・フーリエ変換・拡大縮小・補間などについて紹介しました．この内容をさらに発展させ，本デジタルメディア処理２では，計算機が画像を認識する手法について紹介します．具体的には，画像から目的部分を切り抜く領域分割，画像の局所領域の特徴を計算機が理解できる形で記述する特徴抽出，および，抽出した特徴を用いて画像を識別するパターン認識を紹介します．また，講義の後半では，深層学習を用いた画像処理についても紹介します．

それぞれの技術に関して，コーディング可能な深さで理解できるよう，ソースコードを交えながら詳細な技術解説を行ないます．また，Pythonを用いたプログラミング演習を通して画像処理手法のより深い理解を目指します．講義資料は井尻のweb pageに事前にアップロードします．講義動画は，受講者に限りscombより閲覧可能です．比較的高度な内容に的を絞って話すので，理解が難しい時は利用してください．

○ 達成目標：

1. 領域分割 – 画像の領域分割法について主要なアルゴリズムを説明・実装できる
2. 特徴抽出 – 画像認識に必要な特徴抽出の基礎を説明・実装できる
3. パタレコ – 画像に対するパターン認識（顔認識など）の基礎やアルゴリズムを説明・実装できる

○ 成績評価：

筆記試験（50%），プログラミング課題（50%）に基づき評価します．

○講義資料：

講義において用いた資料・ソースコードは可能な限りWeb上に公開します．以下のURLを参考にしてください．

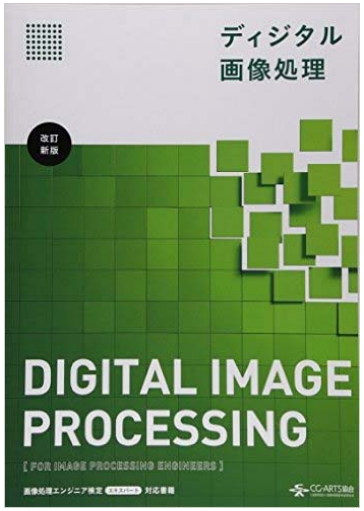
URL：takashiijiri.com/classes

○質問など：

講義に関する質問があれば，講義後またはメールにてご連絡ください．

オフィスアワーは金曜日2限．

takashi.ijiri80 AtMark gmail.ac.jp



教科書

- CG-Arts協会（画像情報教育進行委員会）
- ディジタル画像処理[改訂新版] 大型本

- 日本語で読める画像処理の教科書です
- 画像や例が多く入門者には最適だと思います

- 網羅性が非常に高い，初学者にとっては説明が少ない部分もある
  → 講義中に丁寧に解説します

ある手法を『理解する』とは？

- 教科書をおぼえた：×
- 人にその手法を説明できる：△
- 例を挙げて人に説明できる：○
- プログラムとして記述できる：◎

→ コードを書こう！

※井尻の偏見に基づきます．異論は認めます．
※理解できていない手法も離散化された数式さえあればコードに落とせることも結構あります。。

ソースコードについて

- 本講義紹介する手法はなるべくソースコードも合わせて提供します
  - takashiijiri.com/classes
  - github.com/TakashiIjiri/PythonOpenCVPractice
- Python & OpenCV または MFC & C++ 環境で書いてあります
- インストール方法・コーディングの基本に関する資料も用意します
  - ただし詳細は講義中には触れません
  - 興味のある人だけ自由に勉強を進めてください
  - **学内環境ではインストールの必要がありません（学情の人ありがとう！）→ 説明**

デジタルメディア処理２、2019（前期）

4/19　序論　　：イントロダクション，テクスチャ合成
4/26　特徴検出1　　：テンプレートマッチング、コーナー・エッジ検出
5/10　特徴検出2　　：DoG特徴量、SIFT特徴量、ハフ変換
5/17　領域分割　　：領域分割とは，閾値法，領域拡張法，動的輪郭モデル
5/24　領域分割　　：グラフカット，モーフォロジー処理，Marching cubes
5/31　パターン認識基礎1：パターン認識概論，サポートベクタマシン
6/07　パターン認識基礎2：ニューラルネットワーク、深層学習
6/14　パターン認識基礎3：主成分分析, オートエンコーダ
6/21　筆記試験（50点満点）
6/28　プログラミング演習　1 (基礎的な課題30点, 発展的な課題 20点)
7/05　プログラミング演習　2
7/12　プログラミング演習　3
7/19　プログラミング演習　4
7/26　プログラミング演習　5

復習: デジタルメディア処理1

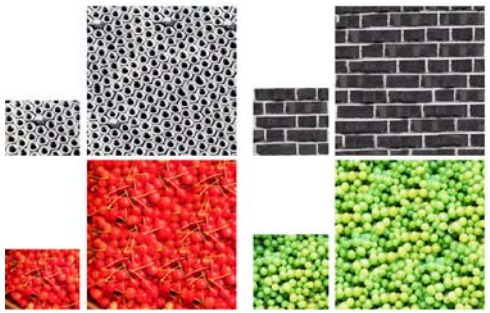
画像とは
画像の変形とアファイン変換
フーリエ変換 (FFT)
フィルタ処理
畳み込み
逆畳み込み

テクスチャ合成

- **テクスチャとは**，ここでは『物体表面に現れる模様』のことを指す
※分野によっては，触感・歯ざわりなどもテクスチャと呼ばれる

- **テクスチャ合成とは，**例となるテクスチャから新たなテクスチャを生成する技術のこと.

図は[Kwatra et al SIGGRAPH 2005]より



小さなテクスチャから大きなテクスチャを生成

画像は[Simakov et al. CVPR 2008]より

画像リサイズ

テクスチャ合成法

- 似た局所領域を検索
  - 画素毎にコピーする
    - Alexei A. Efros and Thomas K. Leung, Texture Synthesis by Non-parametric Sampling, ICCV 1999
  - 画像最適化(片方向類似度)
    - Vivek Kwatra et al. Texture Optimization for Example-based Synthesis, SIGGRAPH 2005
  - 画像最適化(双方向類似度)
    - Simakov et al. Summarizing Visual Data Using Bidirectional Similarity, CVPR 08.
    - Wei et al. Inverse texture synthesis, SIGGRAPH 08.
  - 高速近傍計算

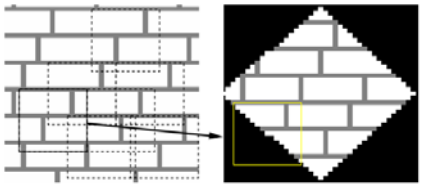
- 目立たないシームを計算

Image quilting
Graph Cut textures

※以降では，実装の詳細を省略して解説しています．実装する方は論文を読んで下さい.

画素ごとに合成

Alexei A. Efros and Thomas K. Leung, Texture Synthesis by Non-parametric Sampling, ICCV 1999
左図はこの論文より



example　　合成中の画像

- 中央からテクスチャを”grow”させる
- 注目画素 p の近傍 w(p) と似た領域 w′ を exampleから検索

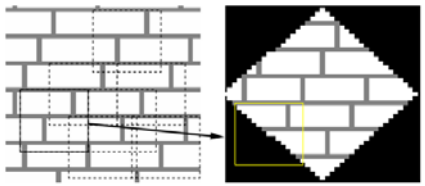
入力：サンプル画像 I_{smp} ， 近傍サイズk
出力：合成画像 I

1. 画像I の中心3x3画素をランダム初期化
2. 以下を繰り返す
   - 2.1 既合成部分の隣接画素 p を選択
   - 2.2 pの近傍 $w(p)$ と最も似た領域 w_{best} を I_{smp} より検索
   - 2.3 w_{best}の中央画素値を p に代入
   - 2.4 全画素の合成がなされたら終了

※ $w_{best} = \underset{w' \in I_{smp}}{\operatorname{argmin}}\, d(w(p), w')$
※類似度$d(w(p), w')$には平均二乗誤差(SSD)を利用
※$w(p)$の欠損部分は無視

画素ごとに合成

Alexei A. Efros and Thomas K. Leung, Texture Synthesis by Non-parametric Sampling, ICCV 1999
左図はこの論文より



example　　合成中の画像

- 中央からテクスチャを”grow”させる
- 注目画素 p の近傍 w(p) と似た領域 w′ を exampleから検索

入力：サンプル画像 I_{smp} ， 近傍サイズk
出力：合成画像 I

1. 画像I の中心3x3画素をランダム初期化
2. 以下を繰り返す
   - 2.1 既合成部分の隣接画素 p を選択
   - 2.2 pの近傍 $w(p)$ と最も似た領域 w_{best}を I_{smp} より検索
   - 2.3 $d(w(p), w') \leq 1.1 * d(w(p), w_{best})$ を満たすすべての w'をI_{smp} より検索
   - 2.4 発見した複数のw'の中央画素値からヒストグラムを作成し，最も頻度が高いものを p に代入
   - 2.4 全画素の合成がなされたら終了

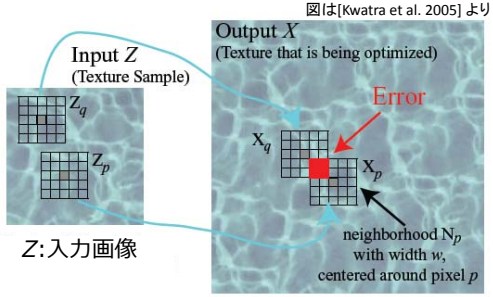
※ 論文で紹介されているアルゴリズムはもう少し複雑

実装例 – プログラミング課題として出題します

画像の最適化

Vivek Kwatra et al. Texture Optimization for Example-based Synthesis, SIGGRAPH 2005

図は[Kwatra et al. 2005] より



Z:入力画像

X：合成画像(出力)

最適化により画像Xを求める

$$\underset{X}{\operatorname{argmin}} \sum_{p \in X^*} ||\mathbf{x}_p - \mathbf{z}_p||^2$$

→ 逐次処理で解く

N_p：画素 $p \in X$ の近傍領域 (窓サイズw=16など)

$\mathbf{x}_p$：N_pのベクトル表現

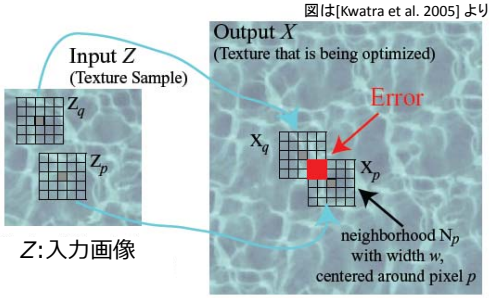
$\mathbf{z}_p$：$\mathbf{x}_p$に最も似た Z 内のパッチ

X^*：出力画像の一部分（論文では縦横ともにw/4 間隔でサンプルした画素群）

画像の最適化

Vivek Kwatra et al. Texture Optimization for Example-based Synthesis, SIGGRAPH 2005（左図はこの論文より）

図は[Kwatra et al. 2005] より



Z:入力画像

X：合成画像(出力)

入力：サンプル画像 Z

出力：合成画像 X

1. Xを乱数で初期化
2. 収束まで以下を繰り返す
   - **2-1. 任意の画素 $p \in X^*$について，$\mathbf{x}_p$ に最も似たパッチ$\mathbf{z}_p$を検索**
   - **2-2. 発見したパッチ$\mathbf{z}_p$をXにコピー**
   - ※２つ以上の領域z_p, z_q重なっている画素は平均を撮る

※論文では，平均でなく、重み付き平均をとる手法なども議論されている

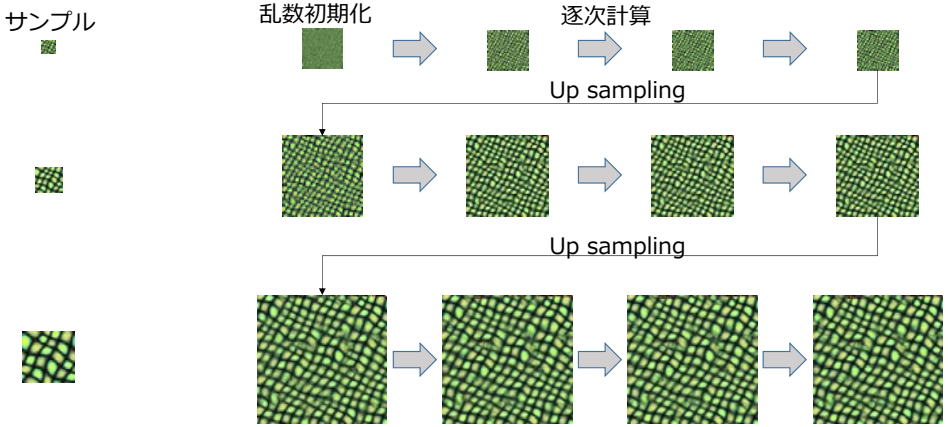
N_p：画素 $p \in X$ の近傍領域 (窓サイズw=16など)

$\mathbf{x}_p$：N_pのベクトル表現

$\mathbf{z}_p$：$\mathbf{x}_p$に最も似た Z 内のパッチ

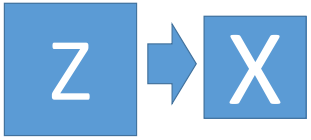
X^*：出力画像の一部分（論文では縦横ともにw/4 間隔でサンプルした画素群）

多重解像度を考慮した合成



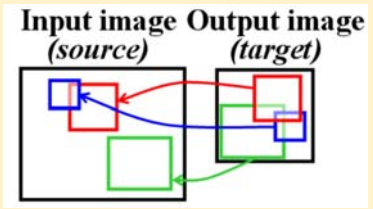
画像の最適化
双方向の類似度

Simakov et al. Summarizing Visual Data Using Bidirectional Similarity, CVPR 08.
Wei et al. Inverse texture synthesis, SIGGRAPH 08.



画像Zから画像Xを合成する問題を考える
上記論文は，画像縮小への応用を紹介

$$\underset{X}{\operatorname{argmin}} \sum_{p \in X^*} ||\mathbf{x}_p - \mathbf{z}_p||^2$$



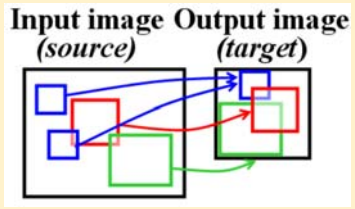
Output画像のパッチ $\mathbf{x}_p$ について似たパッチを探してコピー！
→ 入力画像内で使われない部分も多い

画像は[Simakov et al. 2008]より

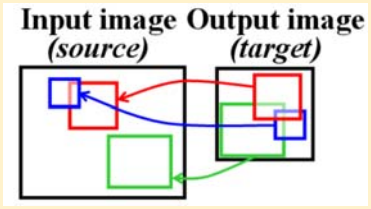
画像の最適化
双方向の類似度

Simakov et al. Summarizing Visual Data Using Bidirectional Similarity, CVPR 08.
Wei et al. Inverse texture synthesis, SIGGRAPH 08.

$$\underset{X}{\operatorname{argmin}} \sum_{q \in Z^*} ||\mathbf{z}_q - \mathbf{x}_q||^2 + \sum_{p \in X^*} ||\mathbf{x}_p - \mathbf{z}_p||^2$$



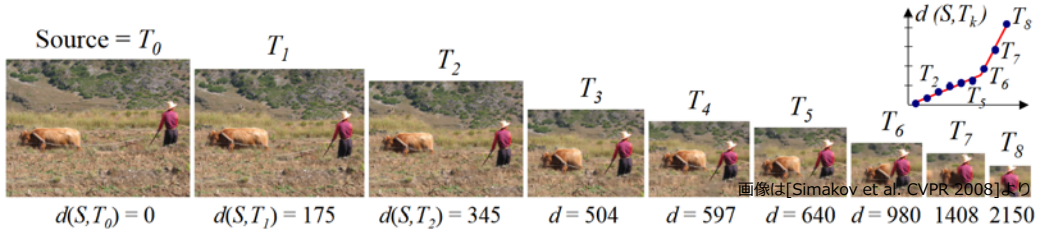
Input画像のパッチ$\mathbf{z}_q$ に最も似たパッチ$\mathbf{x}_q$をoutput画像内から探し，そこへ$\mathbf{z}_q$をコピー
→ 満遍なく入力画像が使われる



Output画像のパッチ $\mathbf{x}_p$ について似たパッチを探してコピー！
→ 入力画像内で使われない部分も多い

画像は[Simakov et al. 2008]より

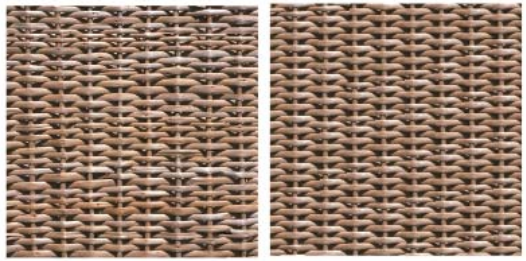
結果 [Simakov et al. CVPR 2008]より



画像は[Simakov et al. CVPR 2008]より

元画像 T_0 から以下を繰り返す
1. サイズを0.95倍し，これを初期解に
2. 先の目的関数を最適化（近傍探索とパッチコピーを繰り返す）

結果 [Wei et al. SIGGRAPH 08.]より



入力画像・合成画像・縮小合成画像
合成画像は元画像よりも一様になる

双方向類似度を考慮する事で
一部のみが使われる事を避けられる

PatchMatch

Connelly Barnes, et al. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing, SIGGRAPH 2009.

ここまでに紹介したTexture合成は，
以下2ステップよりなる
1. 最類似Patch検索
2. Patchの混合

特に1が計算のボトルネックに

→ PatchMatch

隣接画素のmatchingを利用した近似的な最類似Patch検索手法

図は[Connelly Barnes, et al, SIGGRAPH 2009]より

PatchMatch

Connelly Barnes, et al. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing, SIGGRAPH 2009.

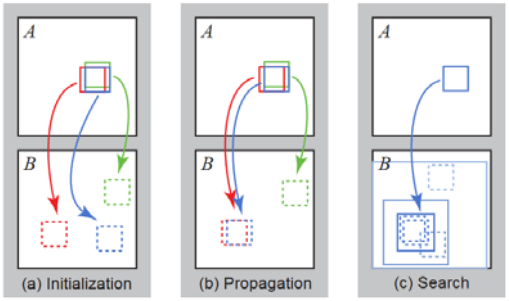
図は[Connelly Barnes, et al, SIGGRAPH 2009]より

問題 画像Aの全パッチについて，画像B内の最も類似するパッチを検索する

1. 初期化: ランダムに対応付け
2. 更新: 画像Aのパッチをラスタスキャンし…

2-1. Propagate, 上隣・左隣のパッチの対応と現在の対応を比べ，最も類似したものを採用

2-2. Search, ランダムに数個のマッチングを作成し，もし現在の対応よりも類似していれば採用



(b,c)では青いパッチ対応付けの更新を行なう．

Propagate: 左のパッチ(赤)の対応先を確認し，その右隣（青破線）と青パッチを比較．現在の対応よりも類似性が高ければ対応付けを更新．上のパッチ(緑)についても同様の処理をする．

Search : ランダムにパッチを選択し，現在よりも良い対応が見つかれば更新．パッチ選択の窓を徐々に小さくする．

高山先生(NII)の講義資料も分かりやすい

p. 6 in http://research.nii.ac.jp/~takayama/teaching/utokyo-iscg-2016/slides/iscg-2016-09-image2.pdf

井尻のC++/CLIのコードは以下のURLへ（そこまで高速化できてないです。。。）

https://github.com/TakashiIjiri/PatchMatch_CppCli