

# コンピュータビジョン

担当: 井尻 敬

## ○講義資料 :

講義で紹介した資料・ソースコードは可能な限りWeb上に公開します.

<https://takashijiri.com/classes>

<https://github.com/TakashiIjiri/PythonOpenCVPractice>

## ○場所 :

対面受講する方

- ・豊洲キャンパス 6階 PC室1 (人数に応じてPC室2も可能に)

オンライン受講する方

- ・ネットワークがあればどこからでも

## ○ 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。2年生時のデジタルメディア処理では、画像処理の基本である『画像データ構造・画像撮影方法・各種フィルタ・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、本コンピュータビジョンでは、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について解説します。また、講義後半では、深層学習を用いた画像処理についても紹介します。

本講義にて解説する技術に関して、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 ([takashijiri.com](http://takashijiri.com)) に事前にアップロードします。

## ○ 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 (25%) , プログラミング課題 (50%) , 小テスト(25%)に基づき評価します.

## ○実施方法 :

講義の座学部分は、『事前学習動画配信&小テスト』『当日の解説』にて実施します

- ・事前学習動画はMicrosoft stream, 小テストはscombを通じて配信します

- ・好きなタイミングで動画を視聴し、関連する小テストに解答してください

- ・小テストの締め切りは、実際の講義直前までとします

- ・当日の解説では講義内容全体を俯瞰・補足を行いつつ、個別の質問に答えます

- ・デジタルメディア処理では小テストの解説のみを行いましたが、もう少し講義内容の補足もしたいと考えています。

- ・講義室より配信するため、講義室・学内・自宅など、好きなところから受講してください

※ 第1回目のみ、事前動画でなく講義室にて対面 + 配信形式で実施します

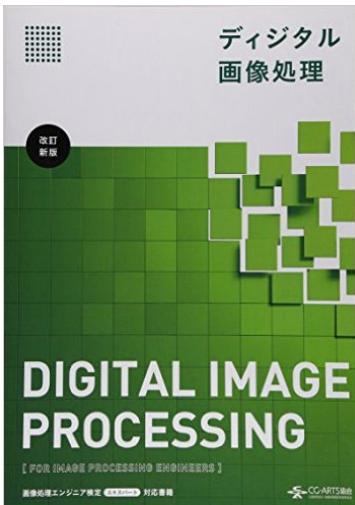
プログラミング演習は、PC室にて実施します

- ・対面参加の方 → PC室にて受講してください

- ・オンライン参加の方 → zoomにてブレークアウトルームを作成します。

※TAは主にオンライン側に配置する予定です。

※質問等ある場合はPC室よりzoom ブレークアウトルームへ行ってTAへ質問してください。



## 参考書

- 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反面、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

※基本的に、スライド資料を中心に講義を進めます

## 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。  
※きちんと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

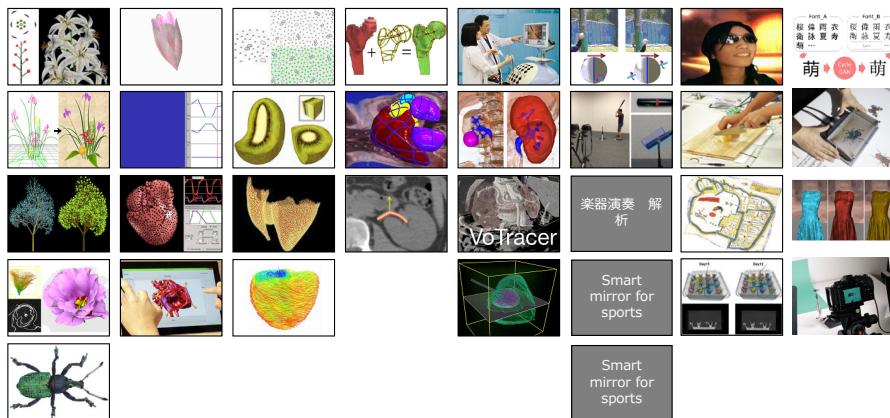
- 本講義紹介する手法はなるべくソースコードも合わせて提供します
  - [takashijir.com/classes](http://takashijir.com/classes)
  - [github.com/TakashiIjiri/PythonOpenCVPractice](https://github.com/TakashiIjiri/PythonOpenCVPractice)
- Python & OpenCV または MFC & C++ 環境で書いてあります
- インストール方法・コーディングの基本に関する資料も用意します
  - ただし詳細は講義中には触れません
  - 興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

## Contents

- 序論 : イントロダクション
- 特徴検出1 : テンプレートマッチング、コーナー検出、エッジ検出
- 特徴検出2 : ハフ変換、DoG, SIFT特徴
- 領域分割 : 領域分割とは、閾値法、領域拡張法、グラフカット法,
- オプティカルフロー : 領域分割残り、Lucas-Kanade法 ※この部分シラバス更新遅れの可能性あり
- 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
- パターン認識基礎2 : ニューラルネットワーク、深層学習
- パターン認識基礎3 : 主成分分析、オートエンコーダ
- 総括と筆記試験
- プログラミング演習 1 : zoom実施 ※ 講義時間中zoomを開設、TAに自由に質問可
- プログラミング演習 2 : zoom実施
- プログラミング演習 3 : zoom実施
- プログラミング演習 4 : zoom実施
- プログラミング演習 5 : zoom実施

# 井尻敬 - takashijiri.com

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



講義を理解するために前提とする数学知識

- ・高校までの数学
- ・線形代数全般 : ベクトル, ベクトルの内積, 行列, 行列の積, 固有値固有ベクトル
- ・最適化数学 (主に最急降下法を利用)
- ・畳み込みとフーリエ変換

ひととおり復習をお願いします。

## 復習: デジタルメディア処理

本講義はデジタルメディア処理（2年後期）の知識を前提とする

- ・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

※未履修の方は適宜独習してください

※過去資料は, <http://takashijiri.com/classes/index.html>

## 画像処理技術に関する研究紹介

目的

- ・プログラミング課題の紹介
- ・教科書レベルの画像処理技術が研究への応用されている例を紹介

Contents

- ・Texture合成
- ・Seam Carving
- ・野球のボールの追跡と回転推定

# テクスチャ合成

## 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 (←課題として出題)

• 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

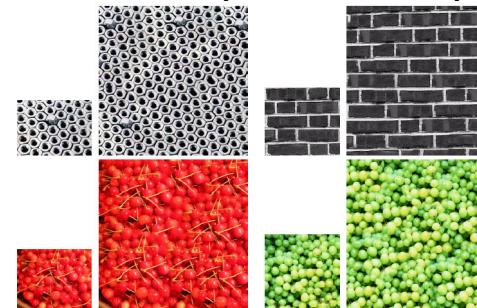
• Image quilting

• Graph Cut textures

- 20年前に出たとても有名な論文です。
- 論文の実装と聞くと難しそうに感じるかもしれません  
が実装自体はとても簡単です。
- 卒研が始まると既存手法の実装が大切になります

- 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・手触り・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

IEEE International Conference on Computer Vision, Corfu, Greece, September 1999

## Texture Synthesis by Non-parametric Sampling

Alexei A. Efros and Thomas K. Leung

Computer Science Division

University of California, Berkeley

Berkeley, CA 94720-1776, U.S.A.

{efros,leungt}@cs.berkeley.edu

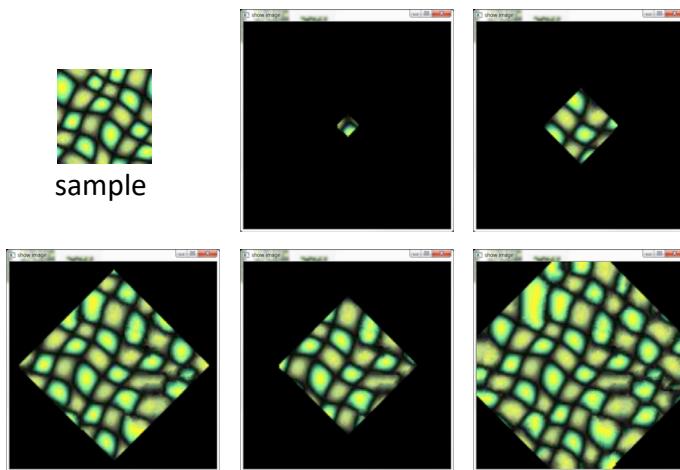
### ICCV 1999

背景：幅広い応用があるため『テクスチャ合成』は需要の高い技術。

課題：当時既存のテクスチャ合成技術は、Markov random fieldや周波数解析に基づくものだったが、多様なテクスチャに対してよい合成は難しかった。

提案：中心から徐々に合成を進めるアルゴリズムを提案。新しい画素値を埋めると  
きに入力画像から似た近傍領域を持つ画素をサンプリングする

## 実装例 - プログラミング課題として出題する予定



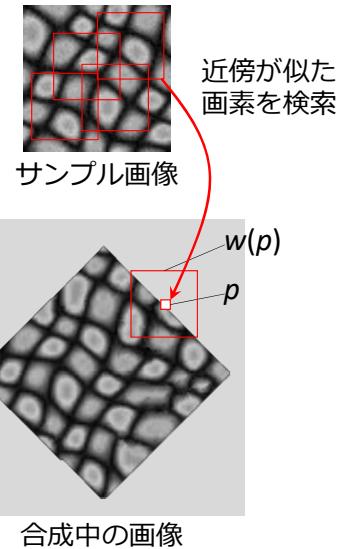
## アルゴリズム (簡易版)

- 中央からテクスチャを "grow" させる

入力 : サンプル画像  $I_{smp}$ , 窓サイズ  $k$   
出力 : 合成画像  $I$

1. 画像  $I$  の中心  $3 \times 3$  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

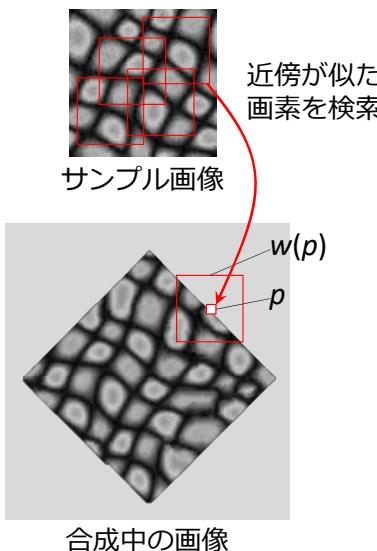
本当はもう少し複雑 (次頁へ)



## アルゴリズム

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  $3 \times 3$  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

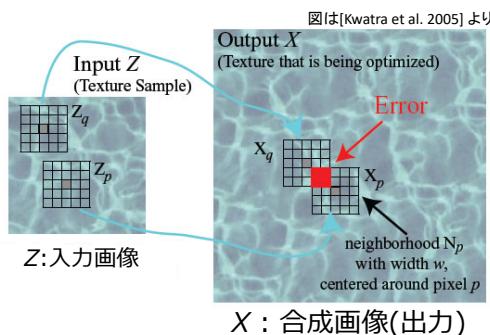


## テクスチャ合成法

- 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
- ここから先は本講義で求める範囲を少し超えます  
• . . . が論文紹介の例として話をします。

## 画像の最適化

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



$N_p$ : 画素  $p \in X$  の近傍領域 (窓サイズ  $w=16$  など)

$x_p$ :  $N_p$  のベクトル表現

$z_p$ :  $x_p$  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)

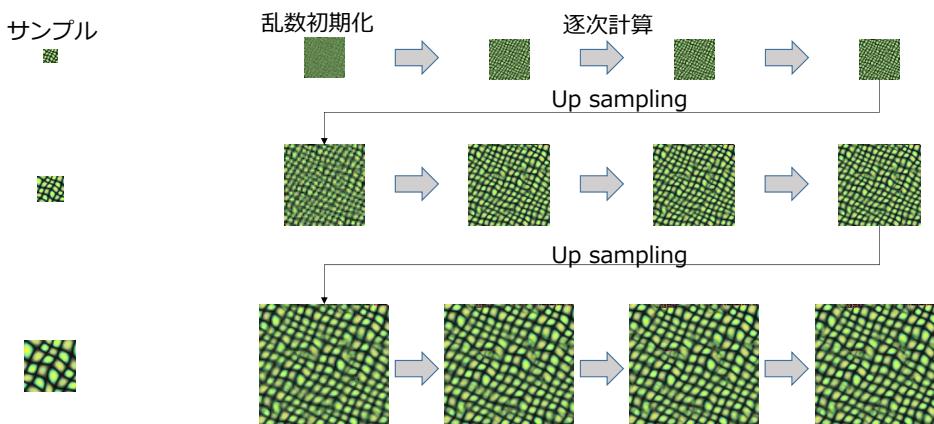
SKIP

最適化により画像  $X$  を求める

$$\operatorname{argmin}_X \sum_{p \in X^*} \|x_p - z_p\|^2$$

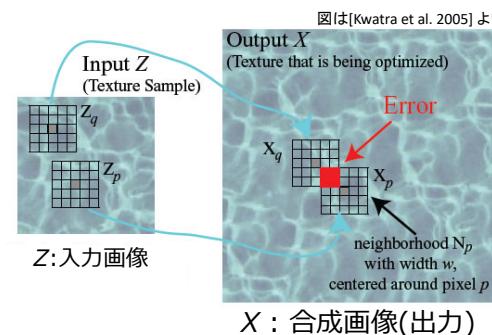
→ 逐次処理で解く

## 多重解像度を考慮した合成



## 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)



$N_p$ : 画素  $p \in X$  の近傍領域 (窓サイズ  $w=16$  など)

$x_p$ :  $N_p$  のベクトル表現

$z_p$ :  $x_p$  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)

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,  $x_p$  に最も似たパッチ  $z_p$  を検索

2-2. 発見したパッチ  $z_p$  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とる

※ 論文では、平均ではなく、重み付き平均をとる手法なども議論されている

## 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

# PatchMatch

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

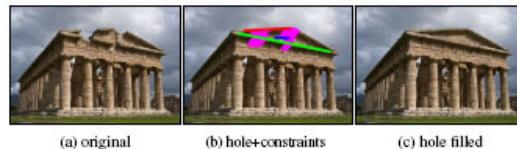
ここまでに紹介したTexture合成は、  
以下2ステップよりなる

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

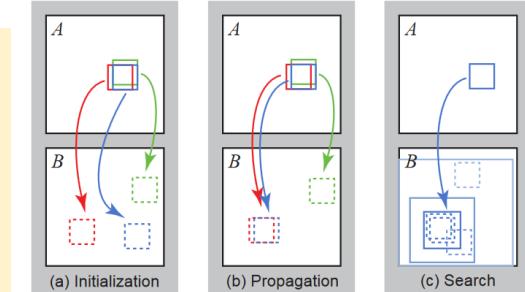
# PatchMatch

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探用
  - 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。

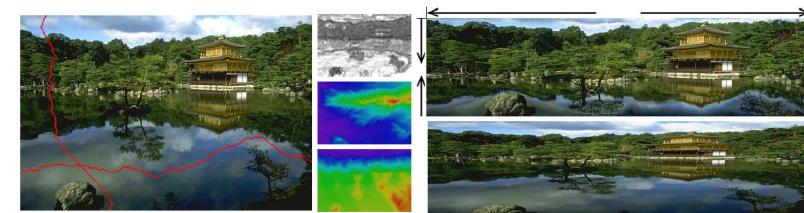
Search: ランダムにパッチを選択し、現在よりも良い対応が見つかれば更新。パッチ選択の窓を徐々に小さくする。

# Seam Carving

## Seam Carving for Content-Aware Image Resizing

Shai Avidan  
Mitsubishi Electric Research Labs

Ariel Shamir  
The Interdisciplinary Center & MERL



背景：コンテンツの縦横比を変化させたいことがある（image retargetingと呼ばれる）

問題：画像のretargetingにおいて単純に横方向に伸縮させると、撮影されたものの縦横比も変化してしまう

提案：画像全体の縦横比を変化させながら映ったものの縦横比を変化させないretargeting手法

## アイディア

- 各画素に重要度を定義（例： $I_x^2 + I_y^2$  or  $|I_x| + |I_y|$ ）
- 横方向に縮める場合は、以下の条件を満たすseamを計算しそれを間引く
  - 画像の上辺から下辺へ画素をひとつずつないだもの
  - ひとつ下に行く際、一画素分横方向に移動できる
  - 重要度の総和が最小となる

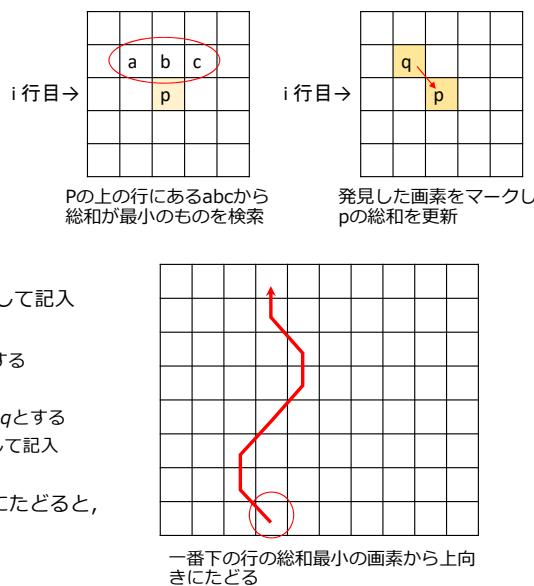
重要度マップ



[図は論文より]

## 解き方

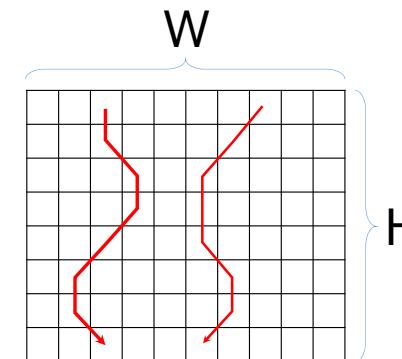
- 各画素に以下の情報を書き込む
  - その画素に到達するSeamの画素値の総和
  - その画素に到達する直前の画素
- Seamの検索アルゴリズム
  - 一行目の画素について、その画素値を総和として記入
  - 2行目からH行目まで以下を繰り返す
    - 今  $i-1$  行目までは計算が終わっているものとする
    - $i$  行目のある画素  $p$  に着目する
    - $p$  の左上、上、右上のうち総和が最小のものを  $q$  とする
    - $q$ までの総和と  $p$  の画素値を、 $p$ までの総和として記入
    - 画素  $p$  の直前の画素として、 $q$ をマーク
  - $H$ 行目において総和が最小の画素から上向きにたどると、Seamが得られる



## 解きたい問題

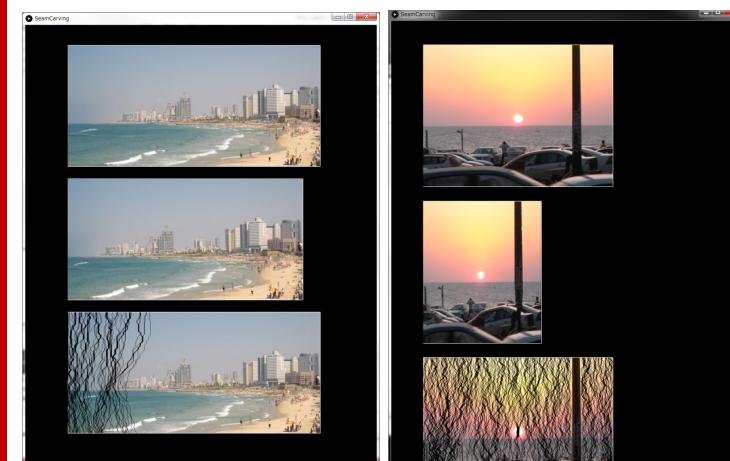
- サイズ  $W \times H$  の画像に対して『通る画素値の総和が最小となるSeam』を求めよ
- ただし…
  - Seamは画素をつなぐものとする
  - Seamの始点は画像の上端、終点は画像の下端の画素とする
  - 一行分上から下に移動する際、左右に1画素分移動してもよい

※画素にはその画素の重要度が入っている



- Seamの数は非常に多いので、すべてのseamを作成し、画素値の総和が最も小さいものを見つけるのは非現実的
- 動的計画法 (Dynamic Programming) が利用できる

## 実装例 - プログラミング課題として出題します



## 野球ボールの回転速度・回転軸推定

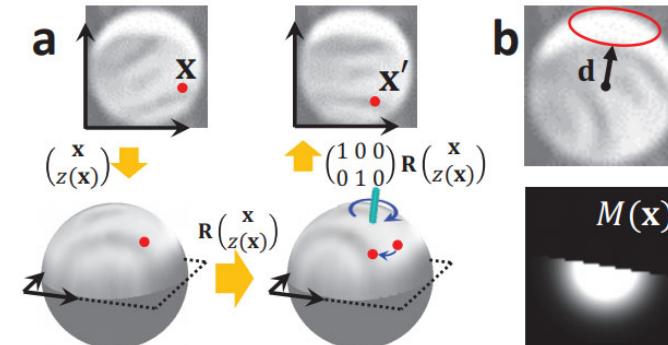


Automatic spin measurements for pitched Baseballs via consumer-grade high-speed cameras, Signal, Image and Video Processing 2017.

背景：野球は多くのプロ・アマ競技者が楽しんでいる国民的スポーツ

課題：変化球の回転軸・回転速度を手軽に知ることは難しい

提案：消費者レベルの高速度カメラにより回転推定を



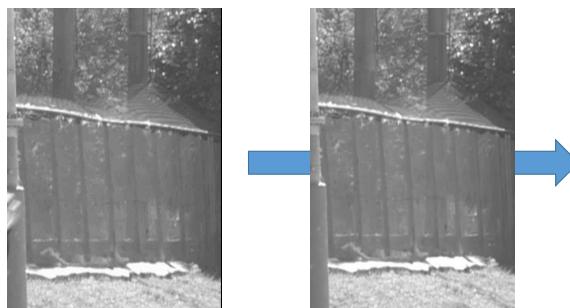
## 動画中の動く物体の抽出

- 三脚による固定を仮定
- 動画内で動く物体に関する画素のみを抽出したい・・・

### →背景差分法

背景画像（動画の平均画像など）を作成

背景画像との差が閾値以上なら動きのある画素として抽出



## 動画中のボール(円)の抽出

- 動画から円状の領域を抽出したい・・・

方法1 → テンプレートマッチング – テンプレートと似た局所領域を検索

方法2 → Hough変換 – パラメータ空間における投票を行う



※それぞれ講義にて解説します  
※Hough変換についてはプログラミング課題として出題します

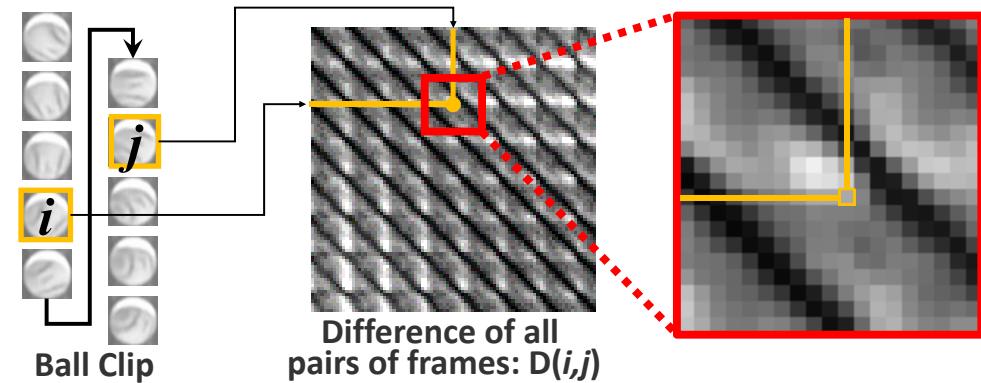
## ボール動画から回転速度を推定

- ・ボール追跡によりボール領域を切り出した動画を作成できた
- ・回転速度をどうやって求める？？



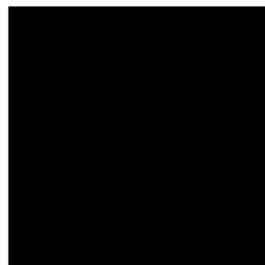
ボールは回転しているので、周期的に似た画像が現れるはず！

- ・フレーム  $i$  とフレーム  $j$  の差分をプロットする
  - ・斜め方向に縞模様が見える → この間隔が回転周期
- ※画像同士の差分についてプログラミング課題に出題します



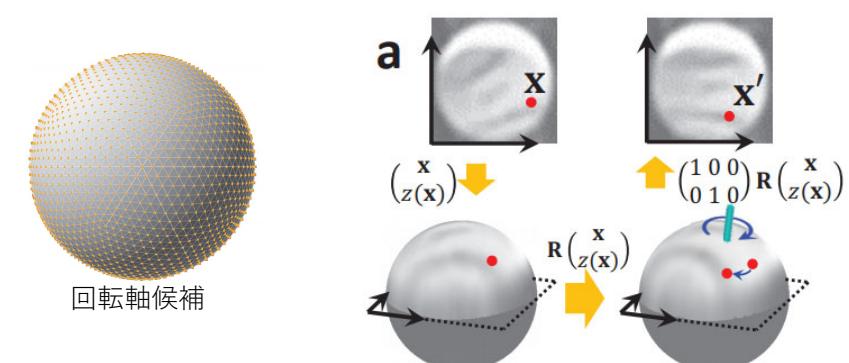
## ボール動画から回転軸を推定

- ・ボール追跡によりボール領域を切り出し、回転速度も取得した
- ・回転軸はどうやって求める？？



## ボール動画から回転軸を推定

- ・回転軸候補を用意（2k本）
- ・ある回軸  $a$  に対して、動画のあるフレーム 3 次元的に回転した画像を作成し隣のフレームと比較
- ・回転した画像が最も隣のフレームとマッチする軸を最適解として出力



# まとめ

- 本科目の実施方法

- 座学部分：事前動画配信 + 当日の解説
- 演習部分：pythonによる画像処理（認識に関するもの多め）課題を出題  
※ 当日解説は、小テスト + 質問 + その他の補足を行う  
※ 対面受講・オンライン受講どちらも可

- 画像処理・画像認識に関する研究紹介

- テクスチャ合成
- シームカービング
- 野球ボールの回転解析

前期14回の間、よろしくおねがいします！