

コンピュータグラフィクス論

－ 画像処理(2) －

2018年7月19日

高山 健志

テクスチャ合成による画像処理

シナリオ 1：画像内物体の消去

元画像



マスク

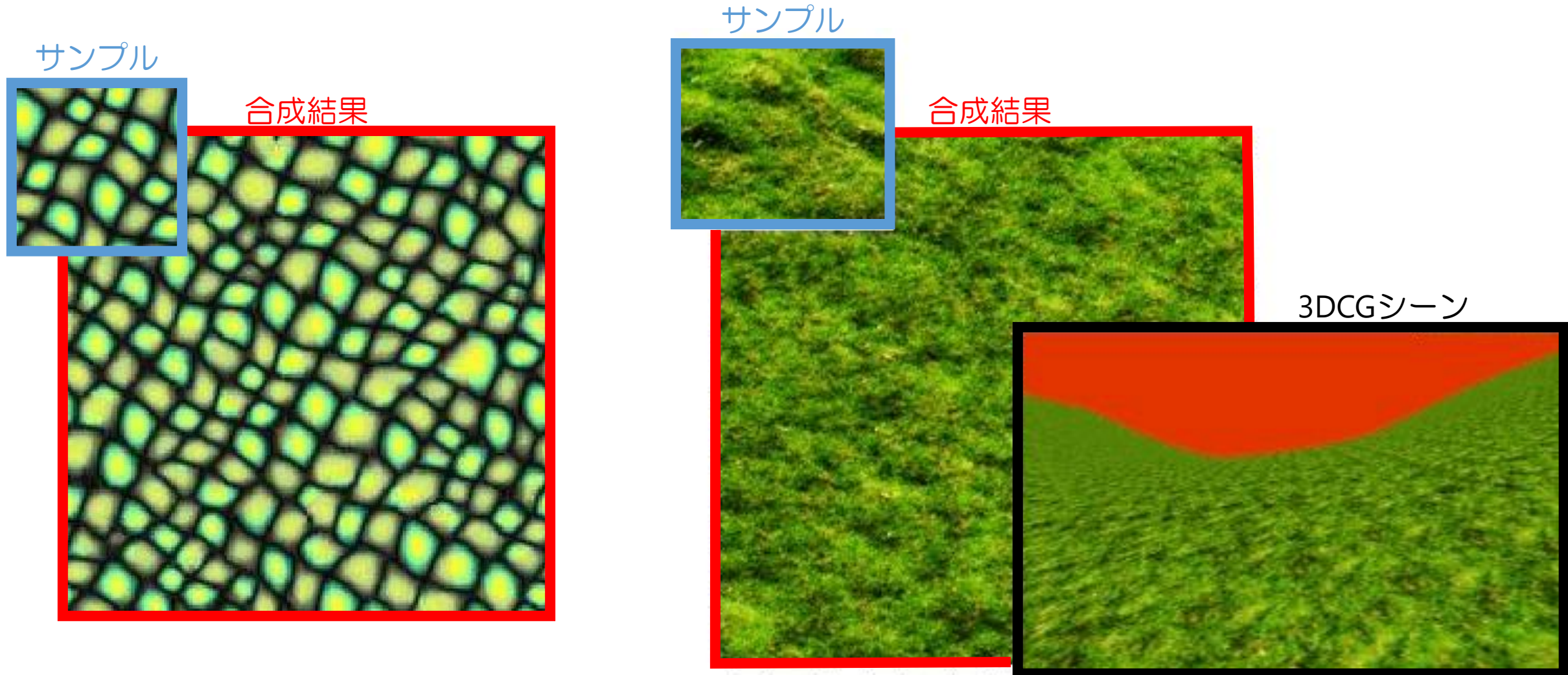


合成結果



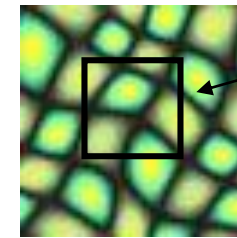
- Image Cloning とは少し違う問題

シナリオ 2：大きなテクスチャ画像の合成



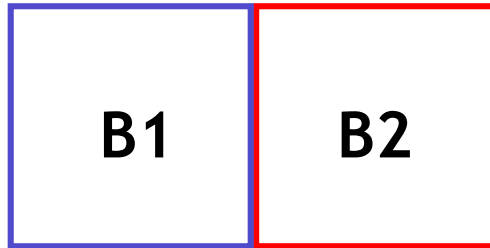
パッチの切り貼りによるテクスチャ合成

Image Quilting [Efros01]

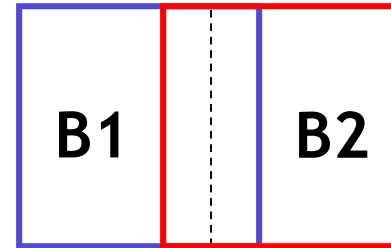


block

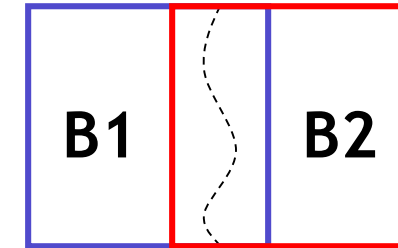
Input texture



Random placement
of blocks



Neighboring blocks
constrained by overlap



Minimal error
boundary cut

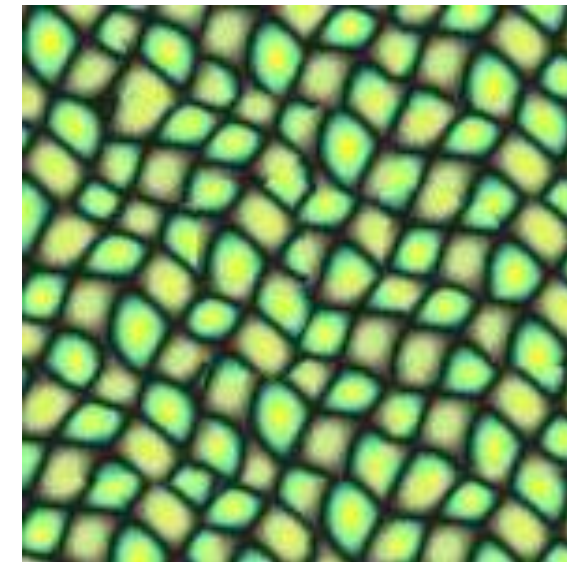
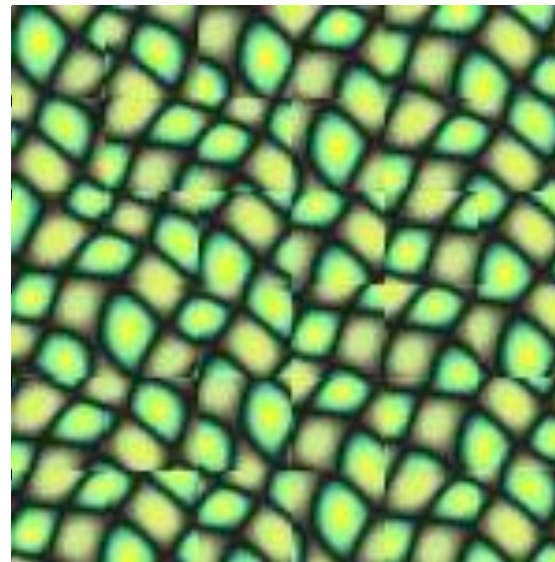
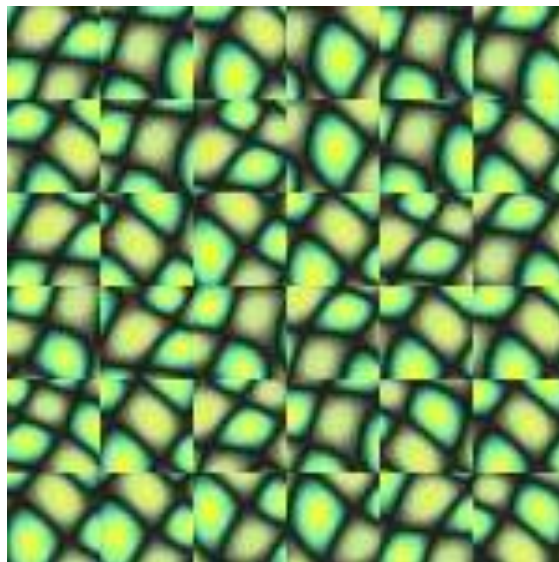
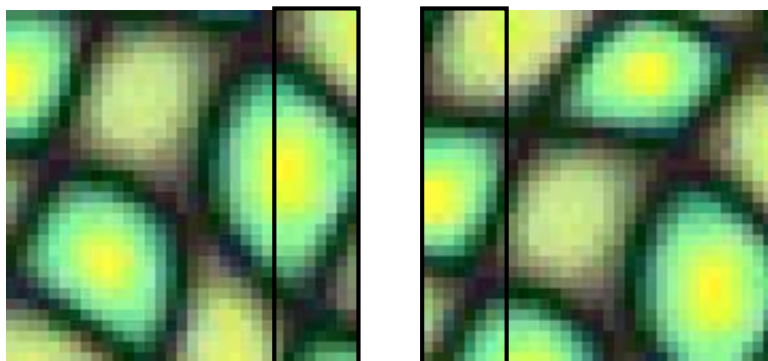
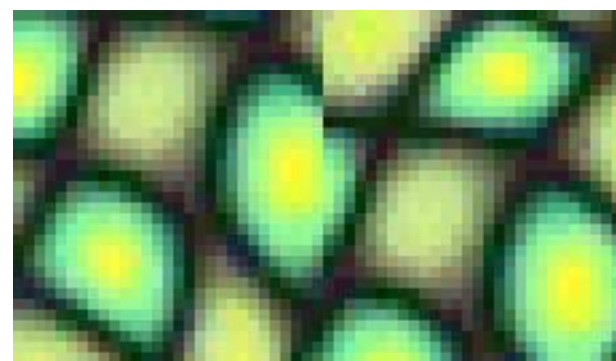


Image Quilting [Efros01]

overlapping blocks

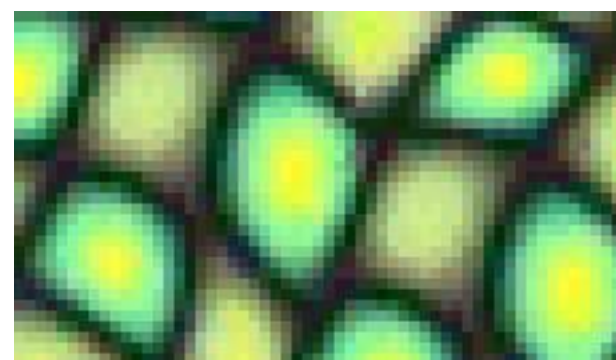


vertical boundary



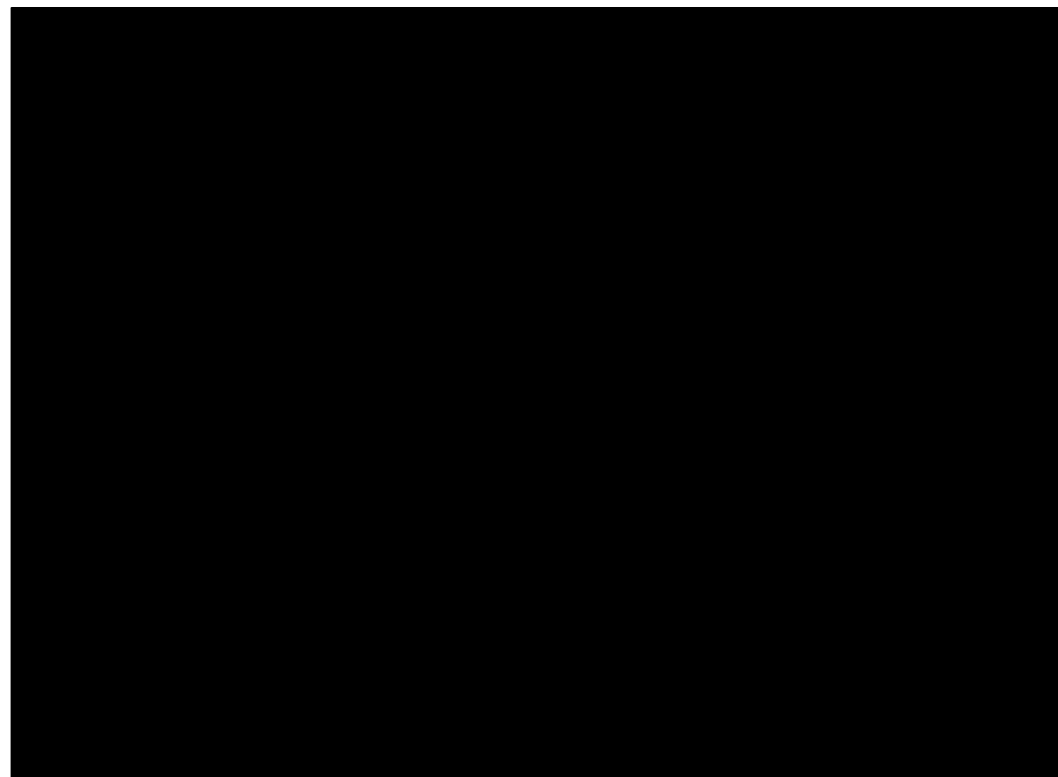
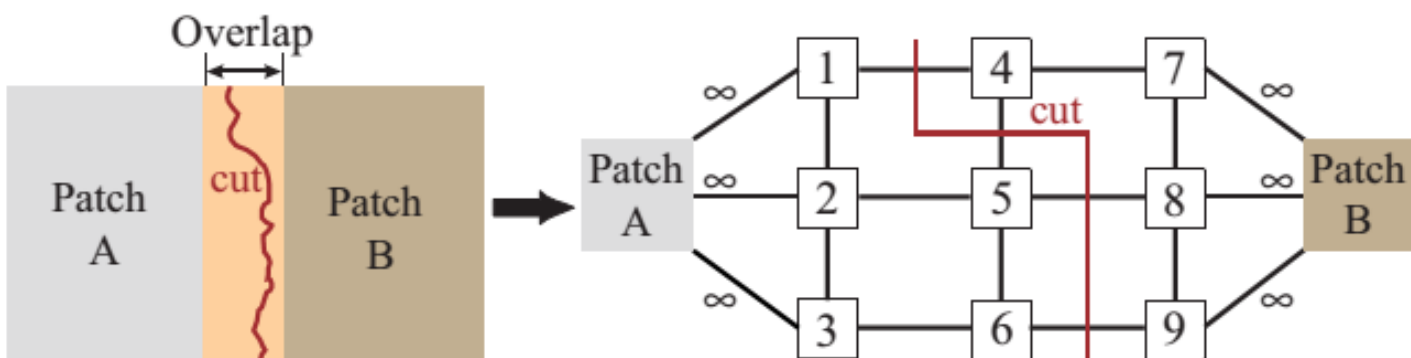
A diagram illustrating the calculation of overlap error. It shows two vertical strips of the image pattern, one to the left of the other, with a minus sign between them. The entire expression is enclosed in large square brackets, followed by a superscript '2'. To the right of this is an equals sign, followed by a vertical strip showing the resulting error, which is a jagged red line on a grayscale background.

overlap error



min. error boundary

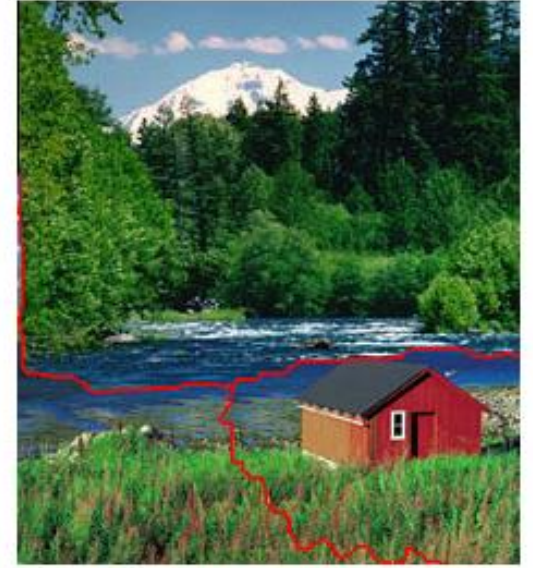
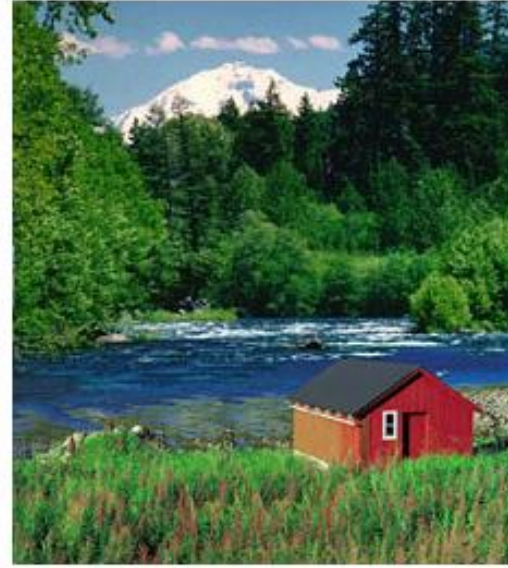
Graphcut Textures [Kwatra03]



<https://www.youtube.com/watch?v=Ya6BshBH6G4>

- パッチ間の最適な切れ目 (seam) の計算を、グラフの最小カット問題として定式化

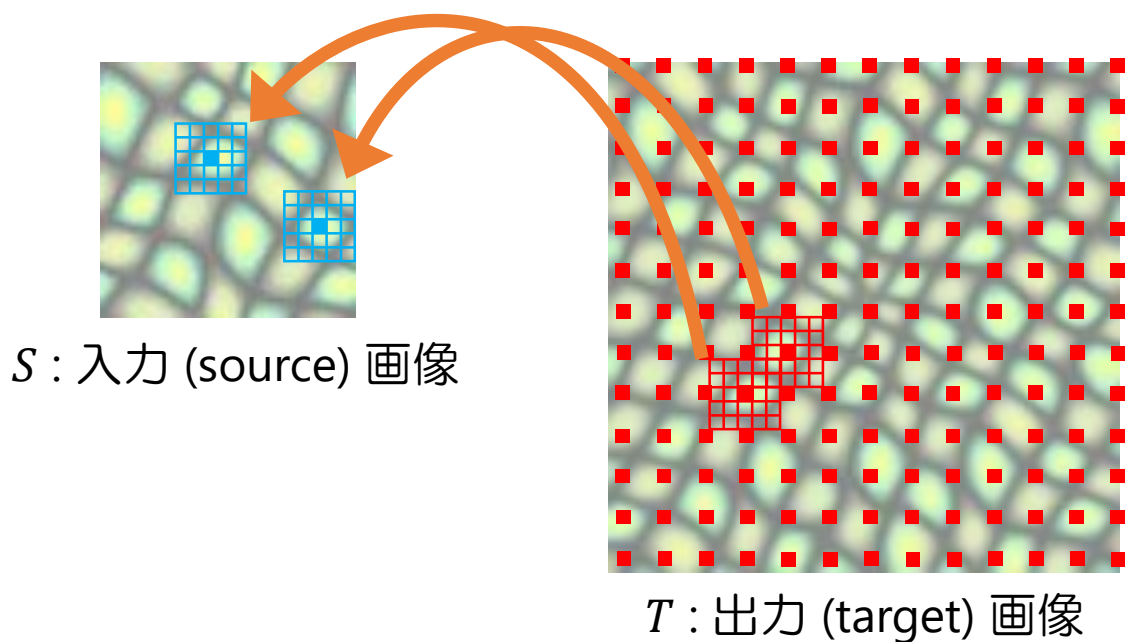
Graphcut Textures [Kwatra03]



最適化によるテクスチャ合成

入力画像と出力画像の類似度 [Kwatra05]

最も似ているパッチを探す



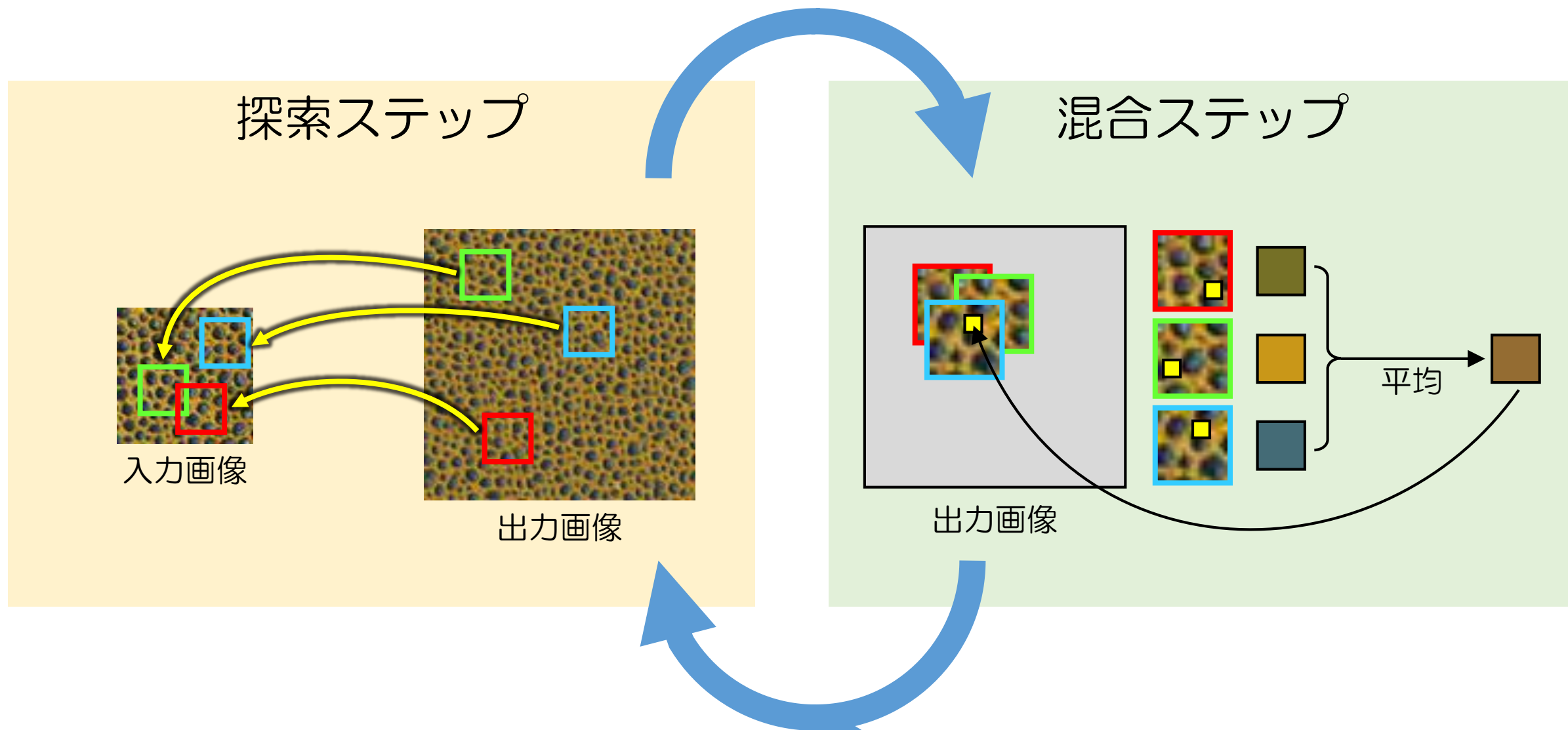
ピクセルごとの
差の二乗の和

$$D(S, T) = \sum_{t \in T} \min_{s \in S} \|s - t\|^2$$

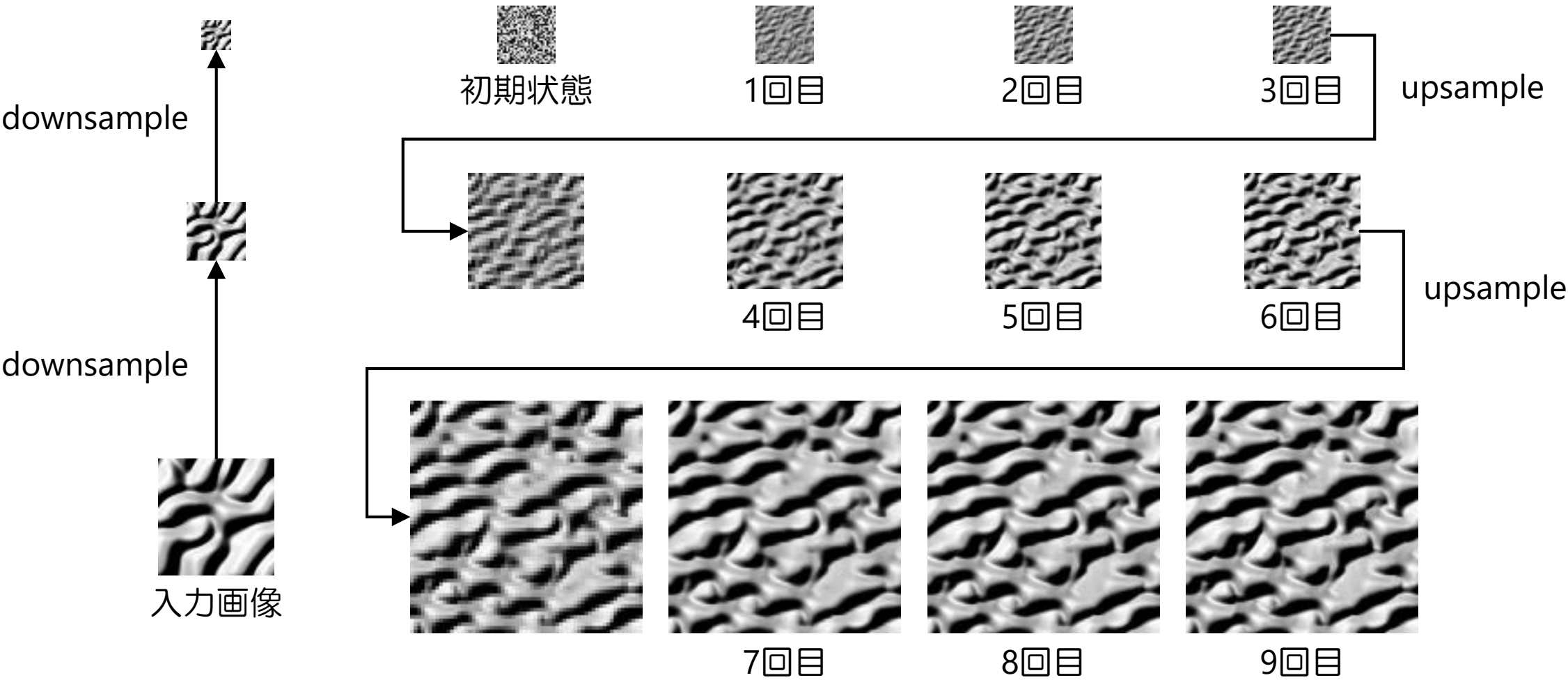
パッチ

- これを最小化する T を求めたい
- 直接には無理 → 繰り返し計算

繰り返し計算による最適化 [Kwatra05]



多重解像度合成

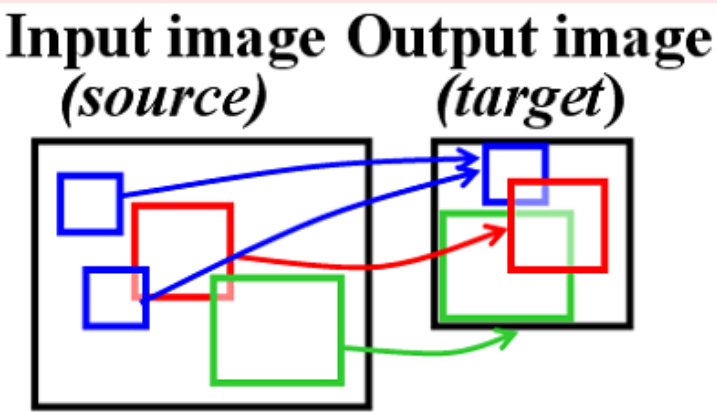


双向性類似度 [Simakov08; Wei08]

$$D(S, T) = \sum_{s \in S} \min_{t \in T} \|s - t\|^2 + \lambda \sum_{t \in T} \min_{s \in S} \|s - t\|^2$$

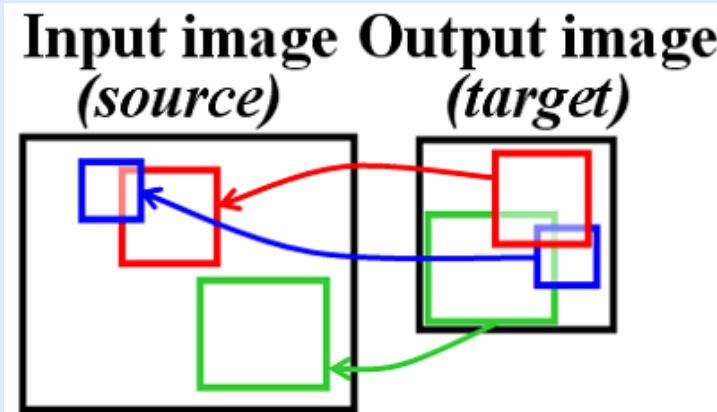
Completeness 項

Input image (source) Output image (target)



Coherence 項

Input image (source) Output image (target)



Completeness / Coherence 項の意義

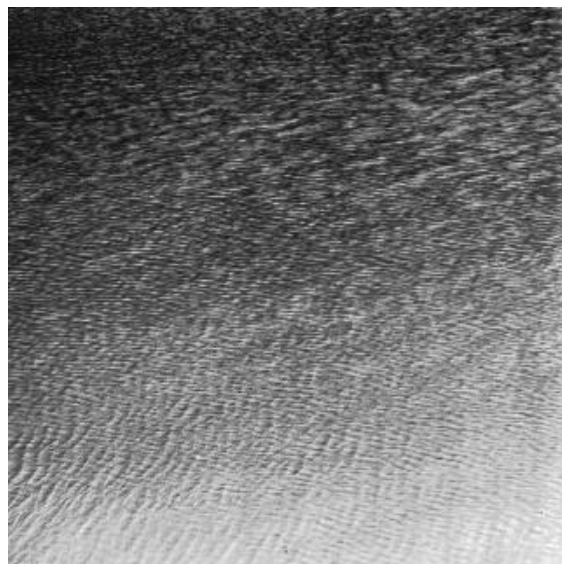
Completeness項：「入力画像の全てのパッチについて、それとよく似たパッチが出力画像中に含まれる」

$$\sum_{s \in S} \min_{t \in T} \|s - t\|^2$$

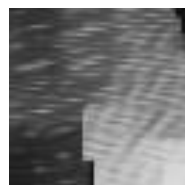
Coherence項：「出力画像の全てのパッチについて、それとよく似たパッチが入力画像中に含まれる」

$$\sum_{t \in T} \min_{s \in S} \|s - t\|^2$$

入力画像



出力画像



Completeness のみ



双方向



Coherence のみ

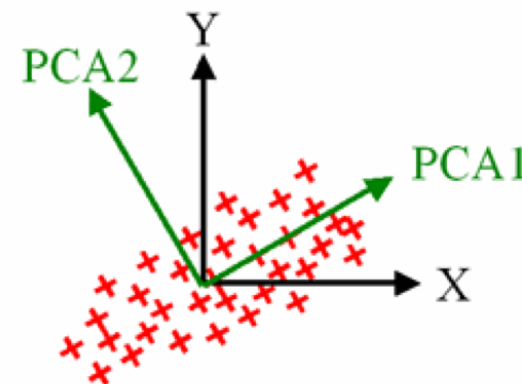
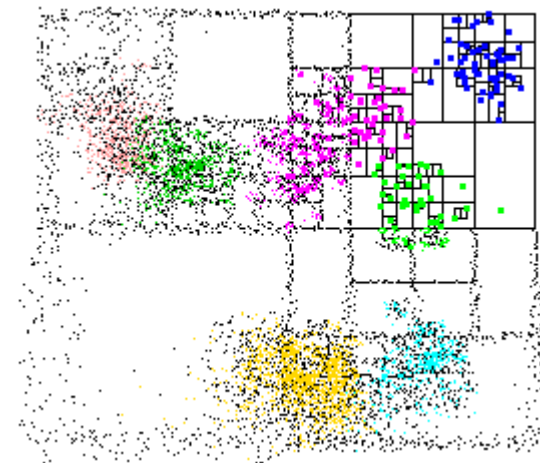


双方向

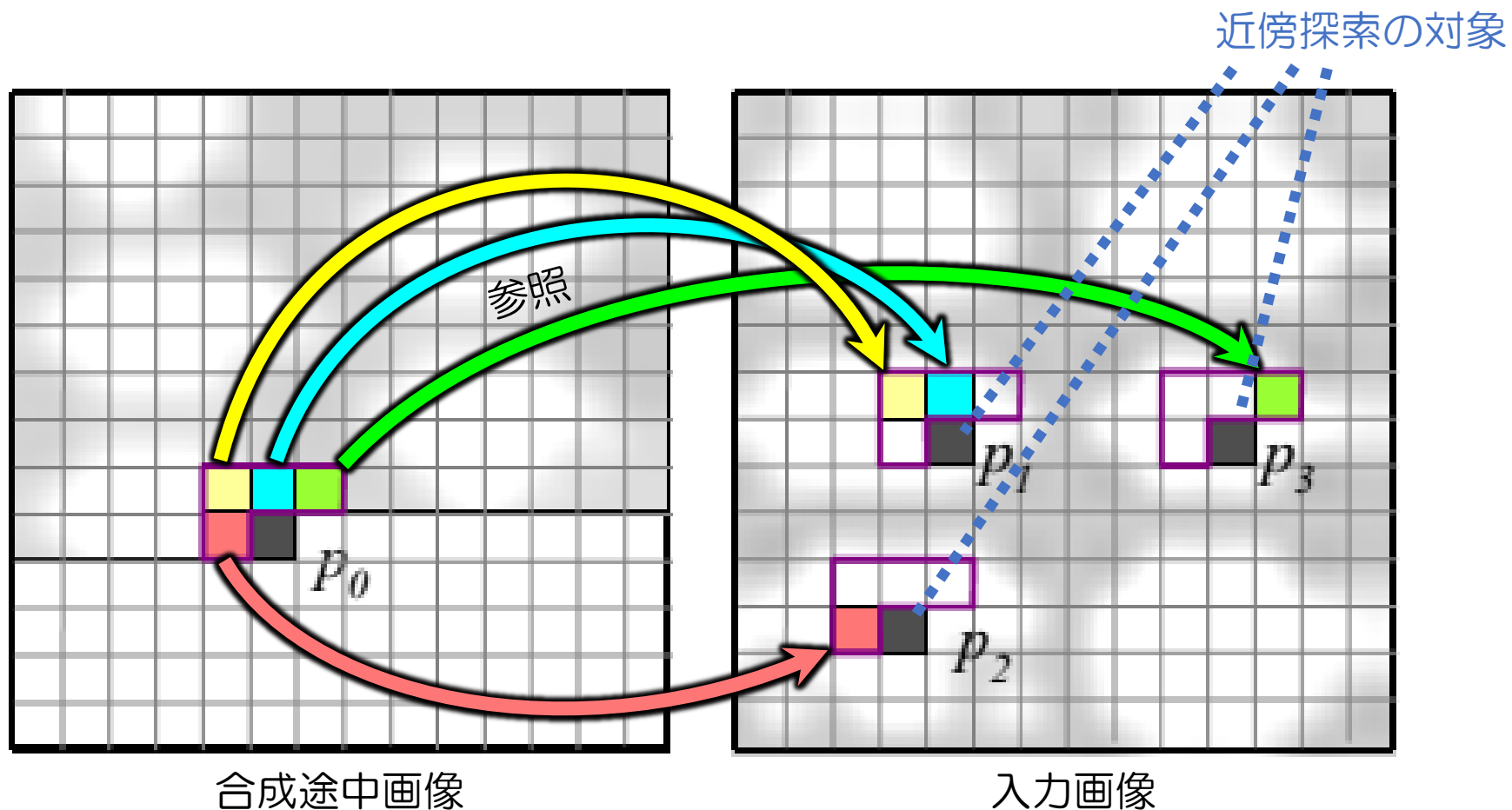
近傍探索を高速化する手法

その 1：空間的データ構造 + 次元削減

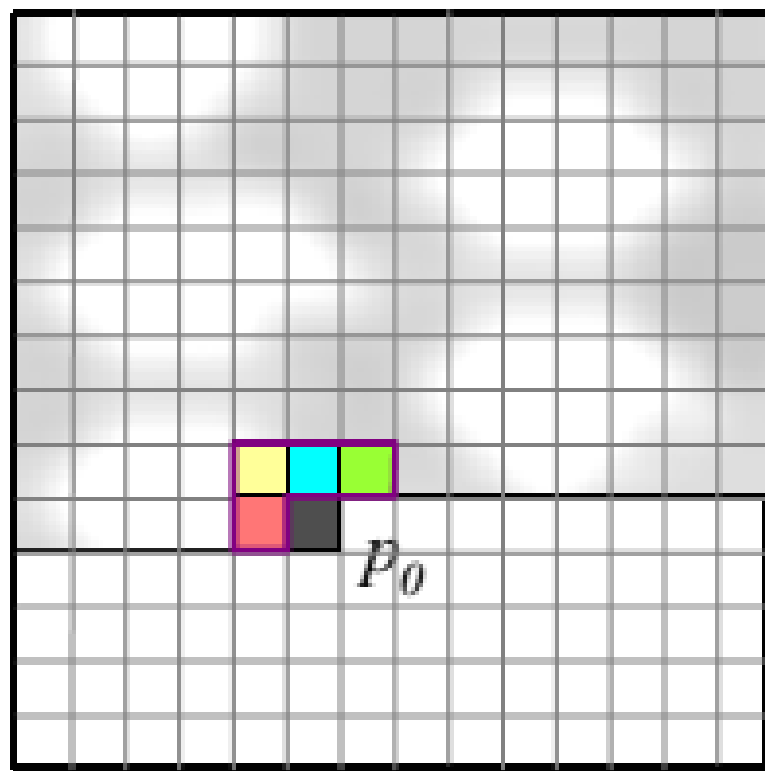
- 5x5 のパッチの各ピクセルが RGB 値を持つ 
→ 75 次元ベクトル
- 高次元空間における nearest neighbor の探索
→ kd-tree による高速化
- kd-tree は次元数が大きすぎると性能が出ない
→ Principal Component Analysis による次元削減



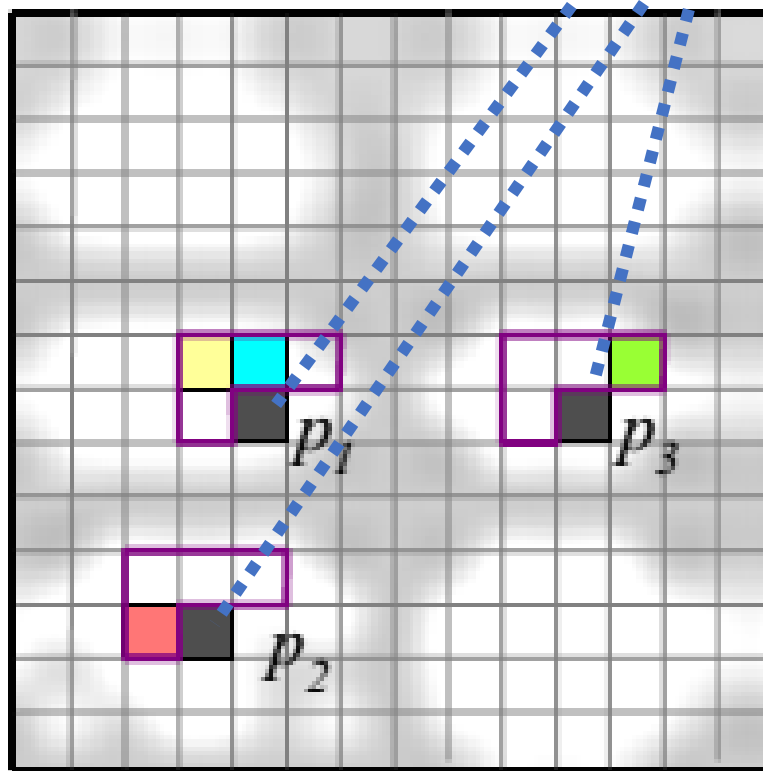
その 2 : k-coherence [Tong02]



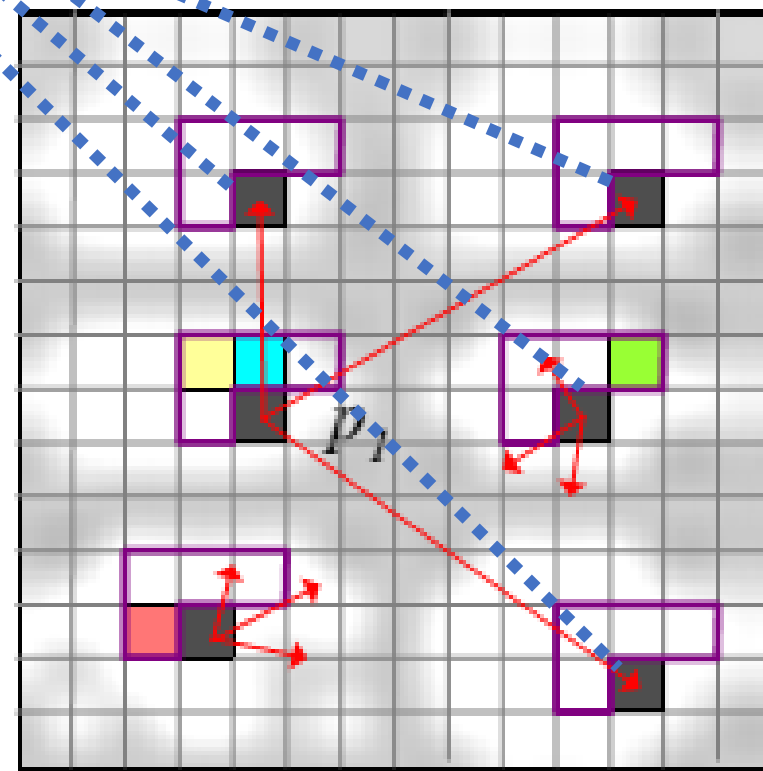
その 2 : k-coherence [Tong02]



合成途中画像



入力画像



入力画像内の類似近傍を前計算

本命：PatchMatch [Barnes09]

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

**Connelly Barnes¹, Eli Shechtman^{2,3},
Adam Finkelstein¹, and Dan B Goldman²**

¹Princeton University

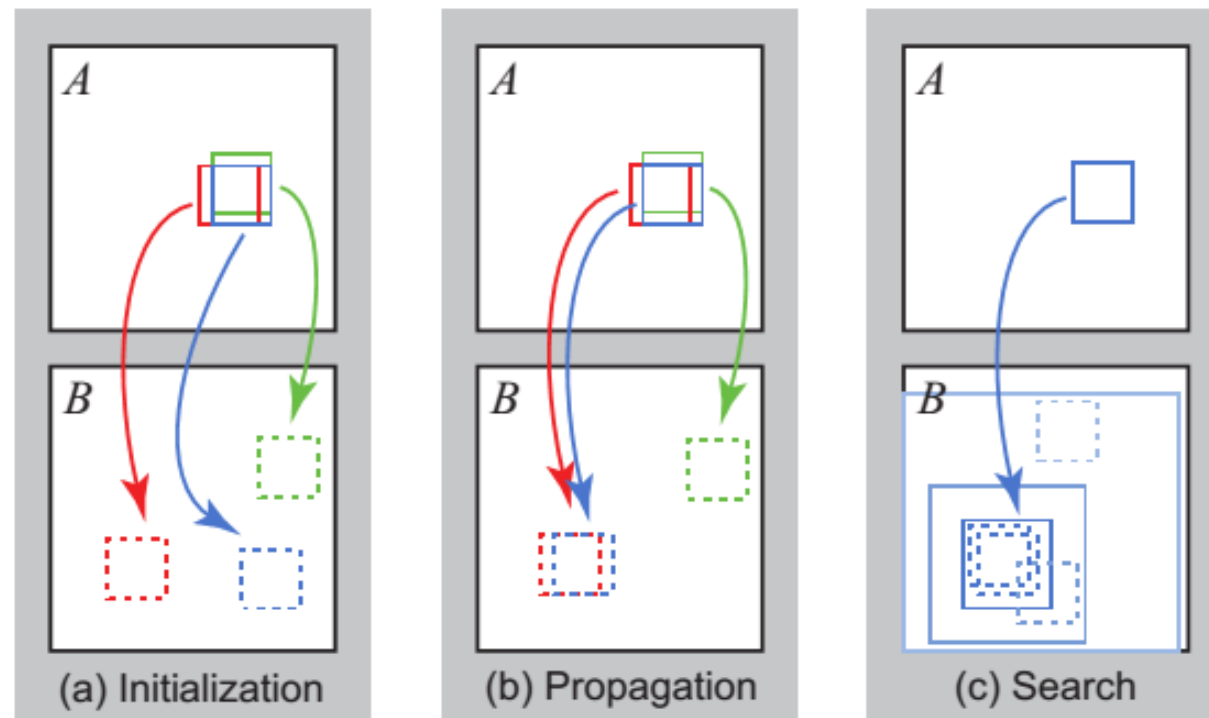
²Adobe Systems

³University of Washington

<https://www.youtube.com/watch?v=dgKjs8ZjQNg>

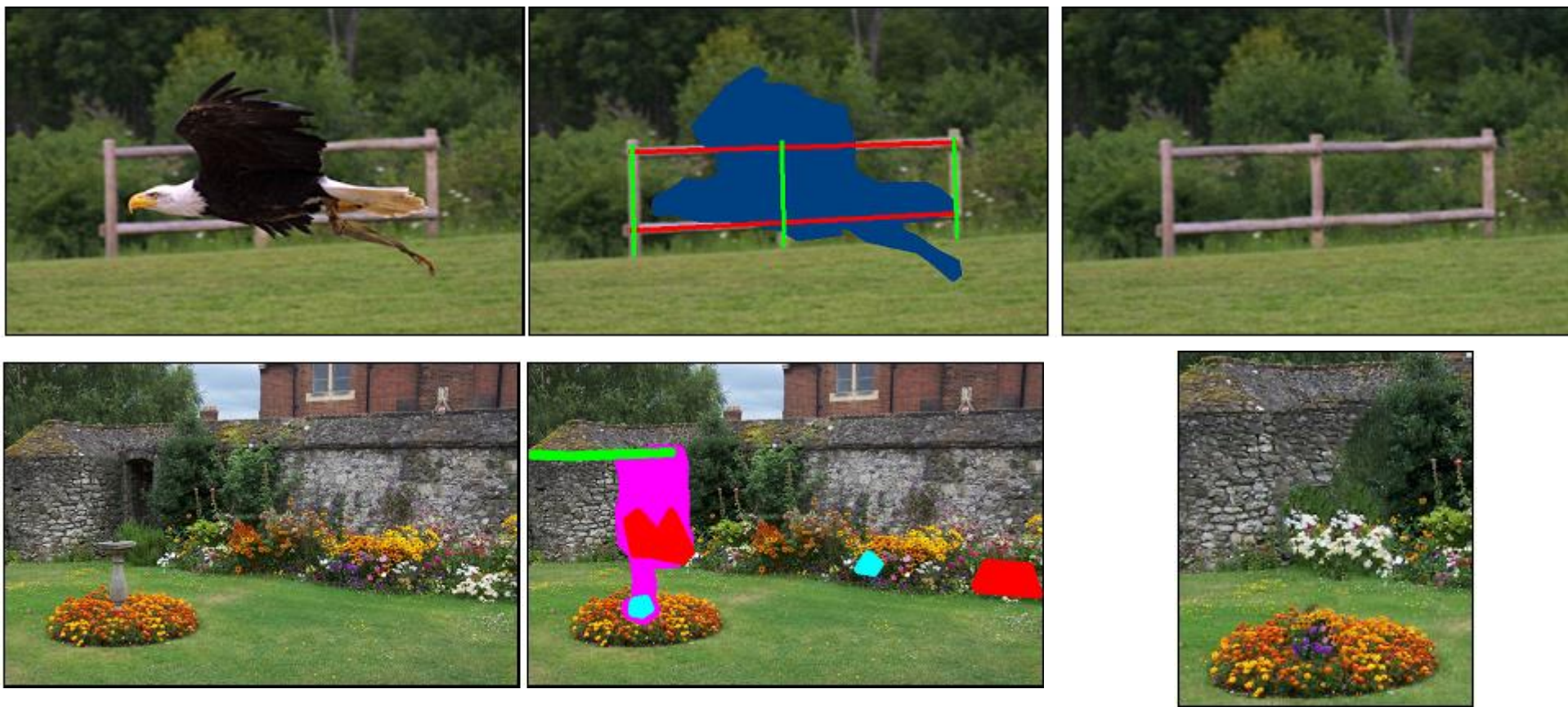
本命：PatchMatch [Barnes09]

- ランダムなマッチで初期化
- スキャンライン順にマッチを更新
 - **Propagation:**
左隣と上隣のマッチが現在のマッチよりも良ければ、それを採用
 - **Random Search:**
何個かランダムなマッチを試し、良いものが見つければ採用
- デモ



拡張 & 応用例

探索空間の限定によるコントロール



- マーカー制約が与えられた出力ピクセルは、同じマーカーを持つ入力ピクセルとしかマッチしない

Image Analogies [Hertzmann01]



- 任意の画像フィルターをテクスチャ合成によってシミュレート
- 多彩なアプリケーション例

Image Analogies – Texture by Numbers

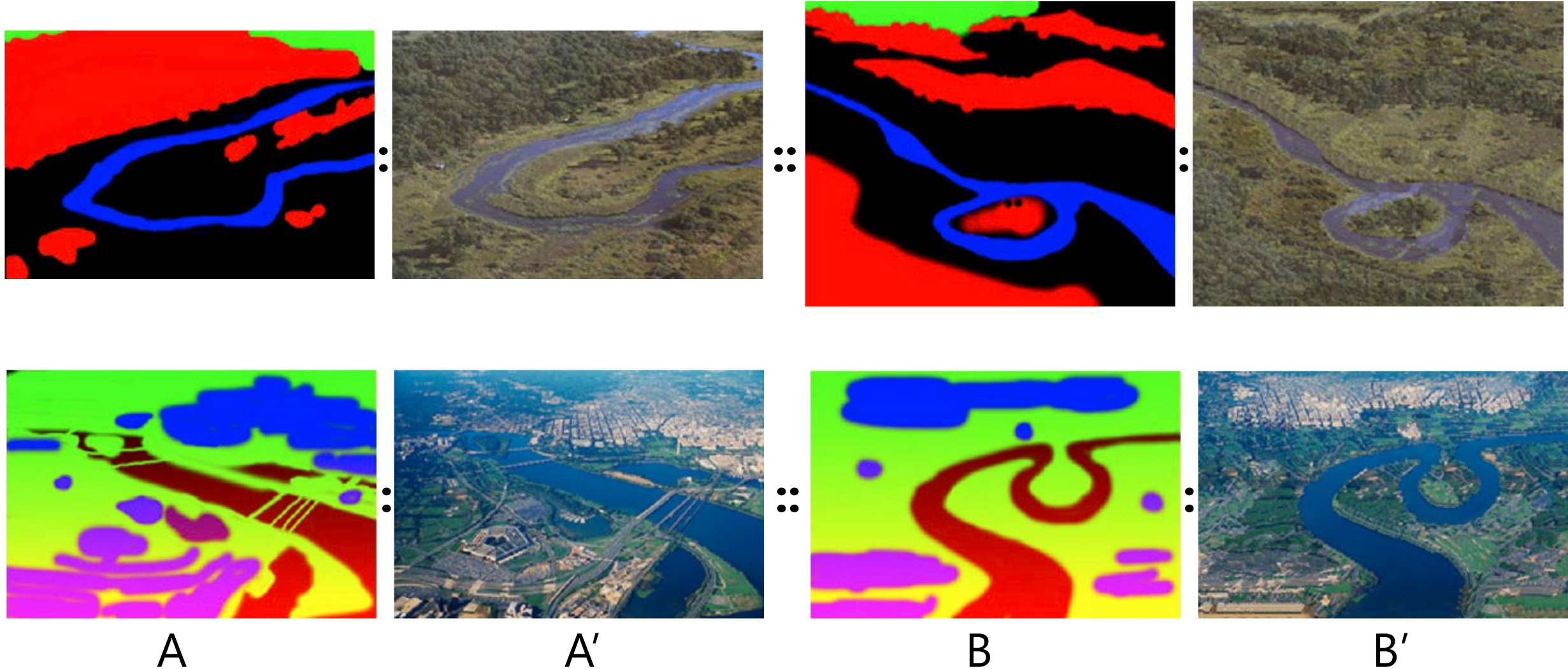
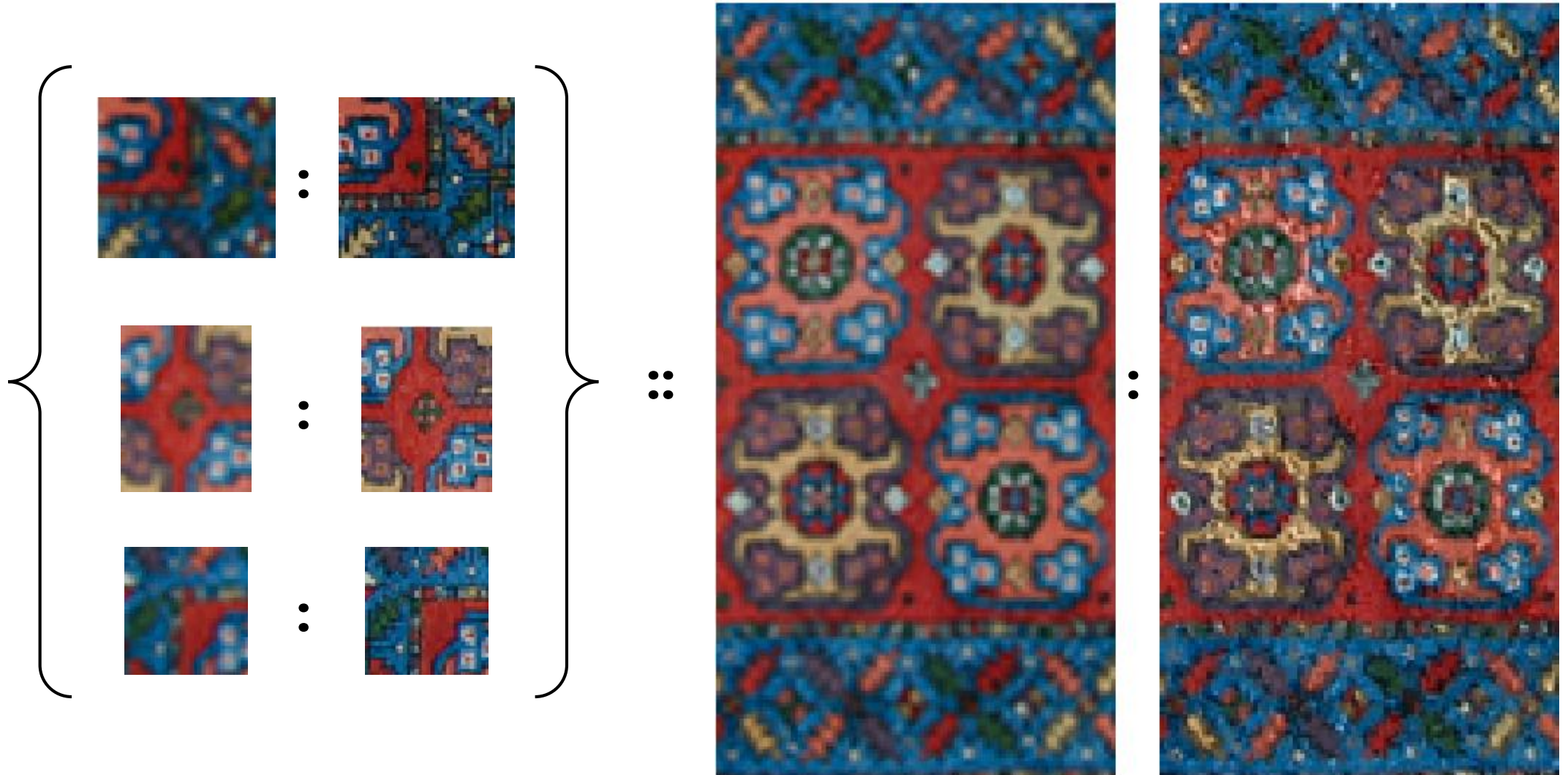
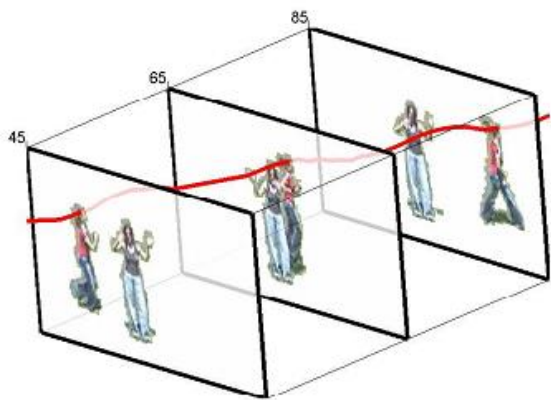


Image Analogies – Super Resolution



動画内物体の消去



Frame 8



Frame 22



Frame 29



Frame 36



Frame 43



Frame 57



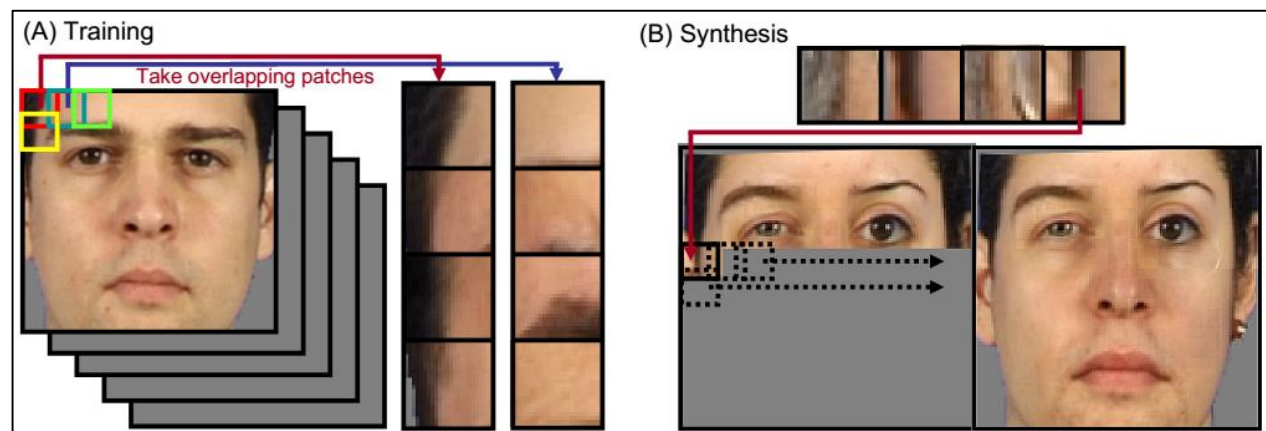
(a) Input sequence

(b) Erasing the occluding person

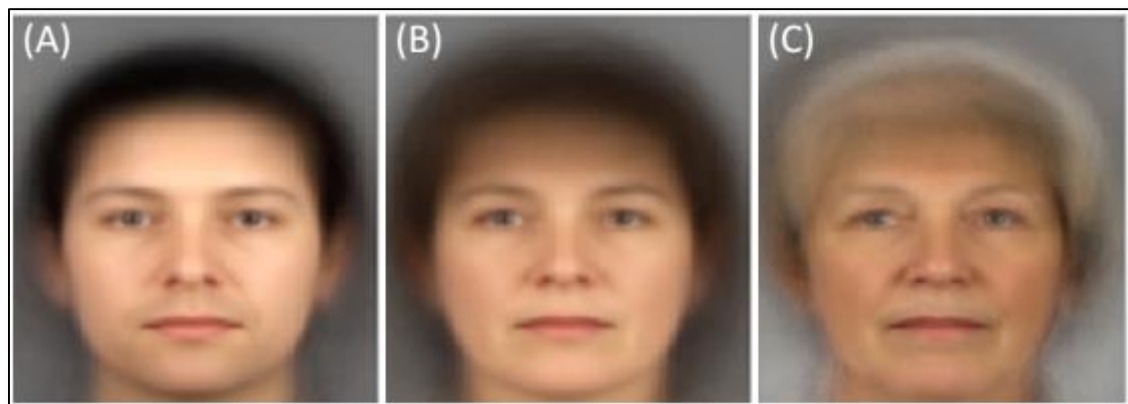
(c) Spatio-temporal completion

顔画像のランダム合成 [Mohammed09]

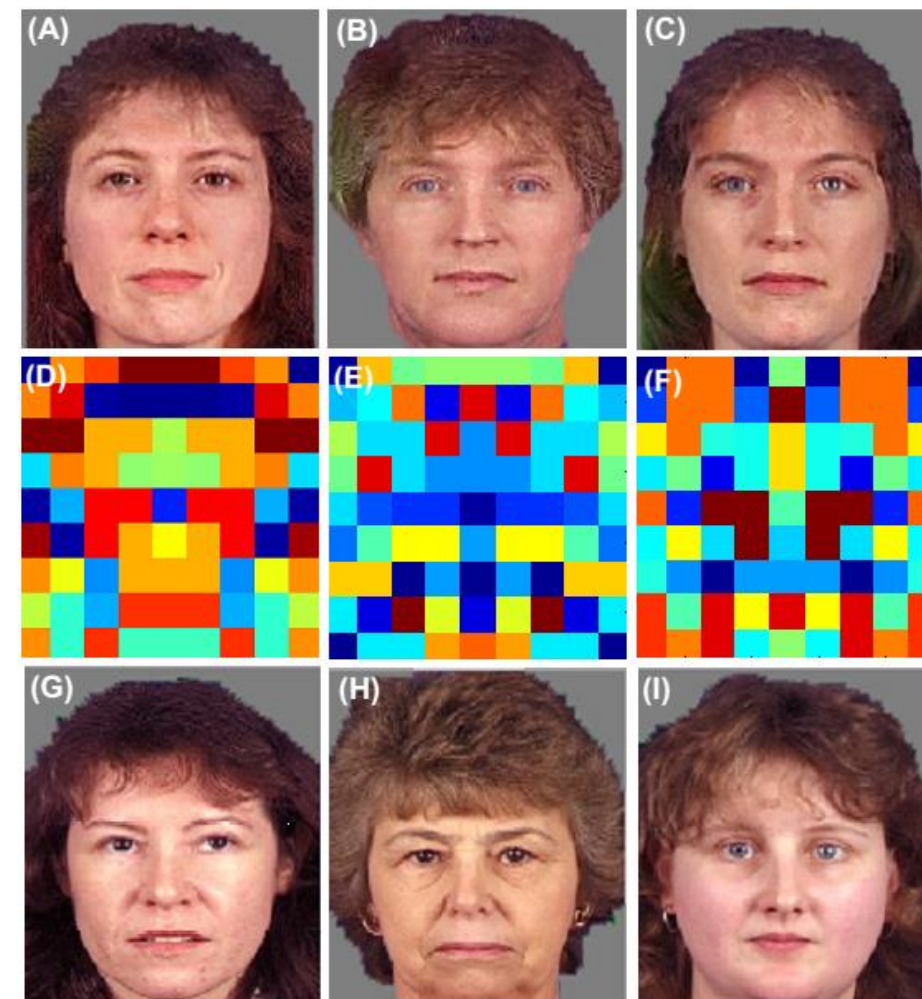
合成結果



位置合わせ済みのパッチごとに画像合成

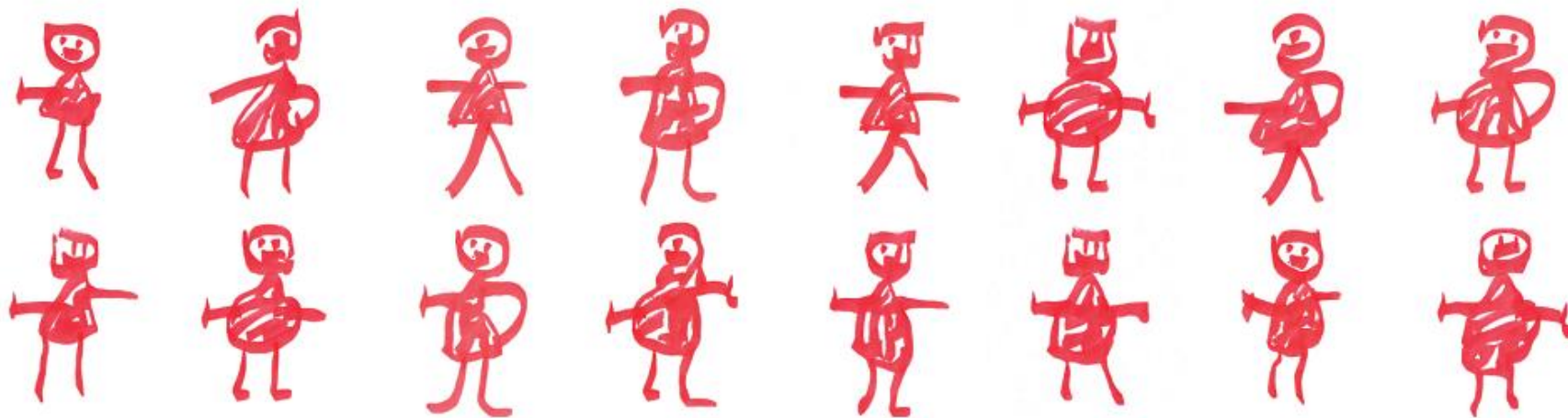
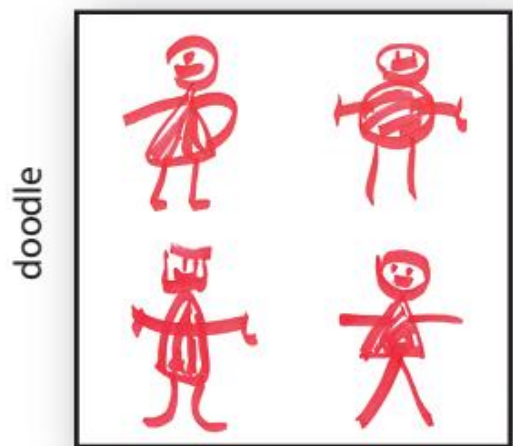


パラメトリックな「平均顔」モデル



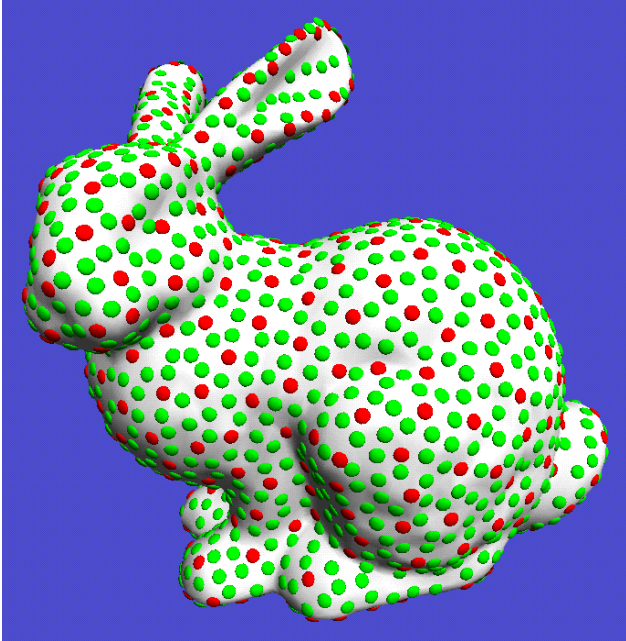
合成結果に最も近いトレーニング画像

構造を持った画像のランダム合成 [Risser10]

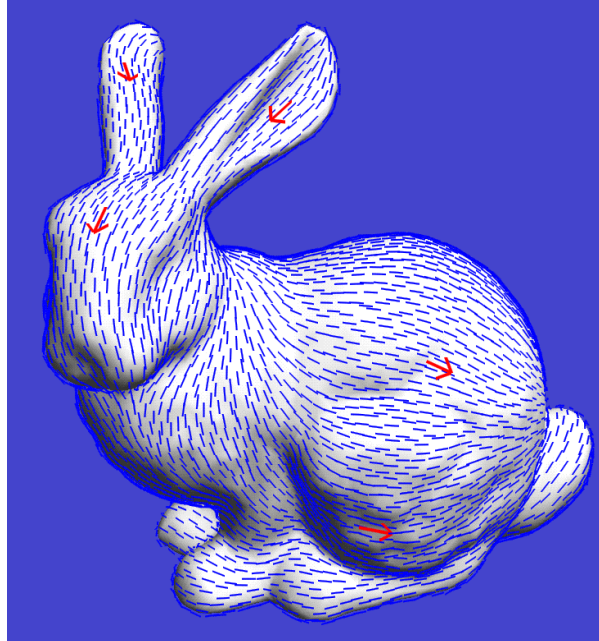


3DCGのためのテクスチャ合成

サーフェス上のテクスチャ合成 [Wei01; Turk01]

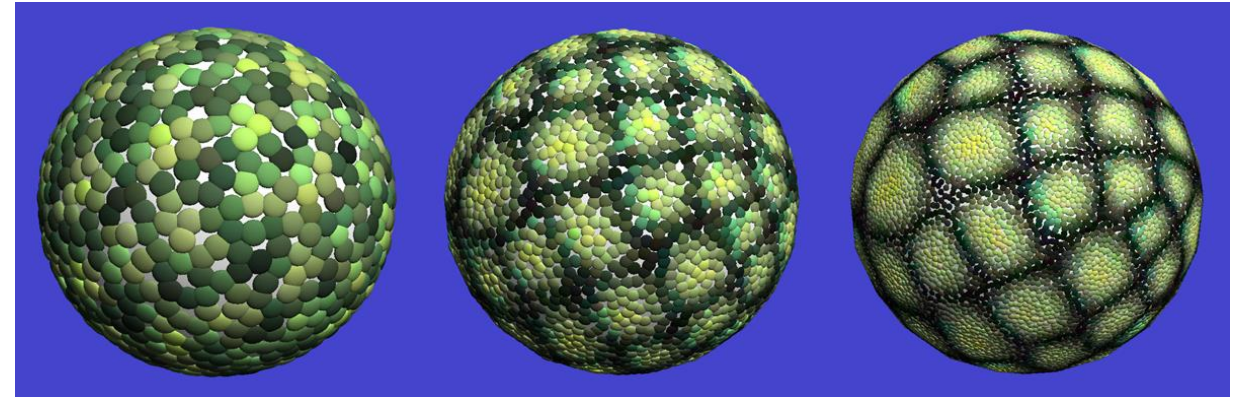


一様なサンプル点

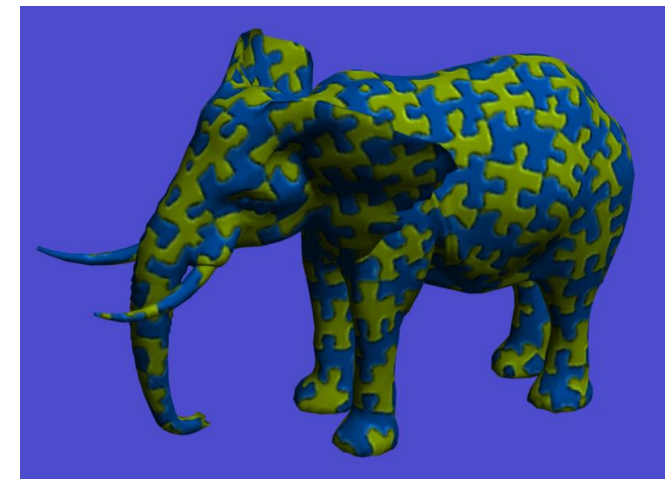
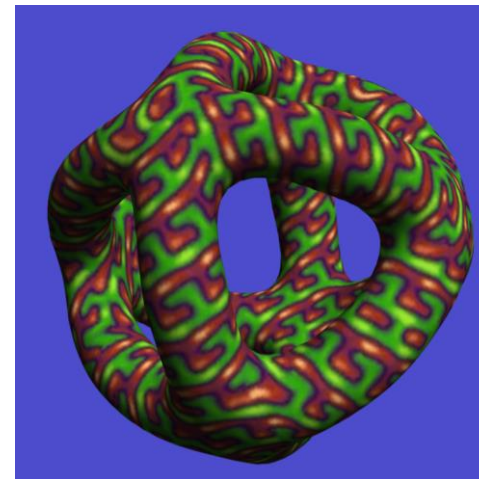


ベクトル場

- 本質的には UV parameterization の上で合成しているのと同様

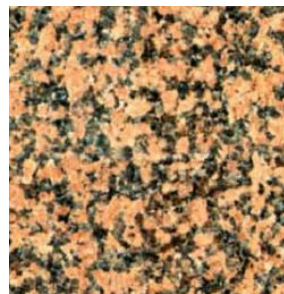


多重解像度による合成



ソリッドテクスチャ

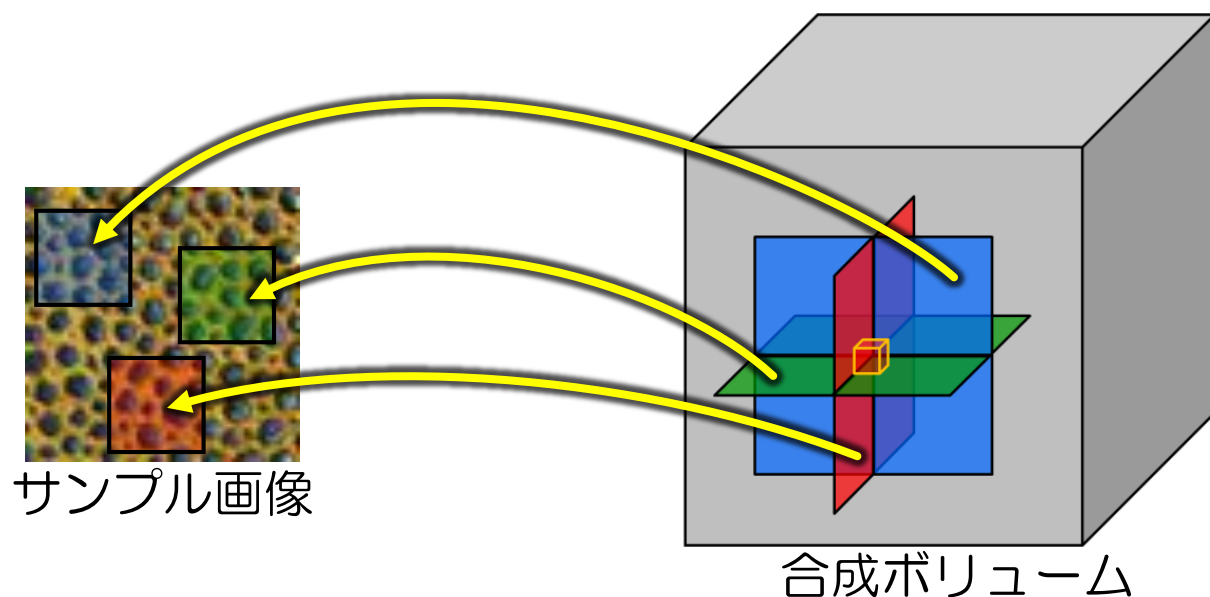
- 3DボリュームとしてRGB情報を保持
→ XYZ 座標から直接色が求まる = 簡単！
- 初期の技術
 - ノイズ関数等を組み合わせ、係数等を調整
 - 統計的なアプローチによる自動合成
(ノイズに近いタイプの画像に限定)



```
marble(x, y, z)  
= colormap(sin(x + noise(x, y, z)))
```

最適化によるソリッドテクスチャ合成

- 2D版 [Kwatra05] をほぼ素直に3Dに拡張

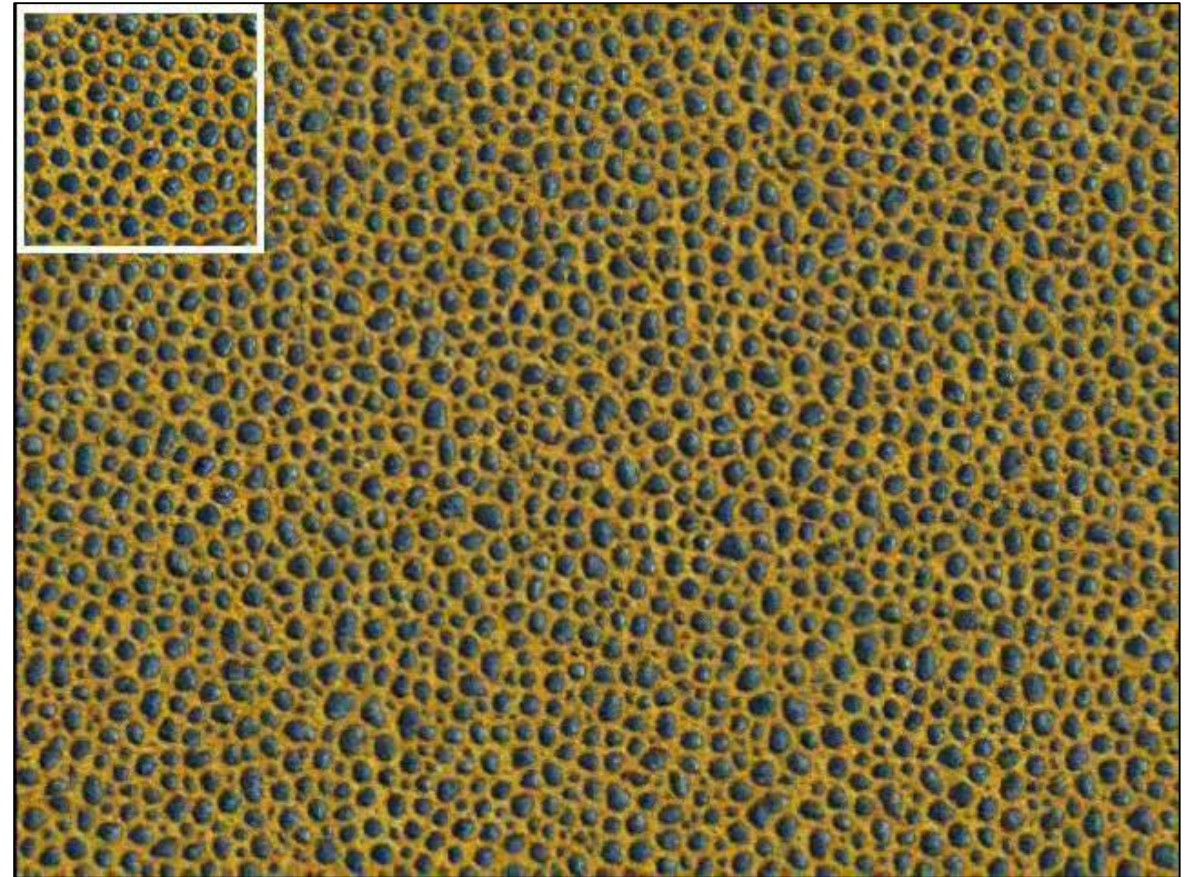


Solid
Optimization

<https://www.youtube.com/watch?v=KG8kykTNBVk>

GPU並列実行による高速オンデマンド合成 [Lefebvre05]

- 基本的な考え方は [Kwatra05] と同様
 - 前計算と計算過程の独立性がキモ
- 描画時に合成 = メモリ容量節約
→ ゲーム向き



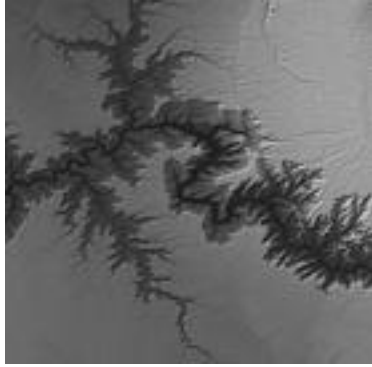
建物に特化したオンデマンド合成 [Lefebvre10]

- 縦横の seam を前計算しておき、GPU実行時に繋ぎ合わせる



テクスチャ合成の考え方を 画像以外の対象に適用した例

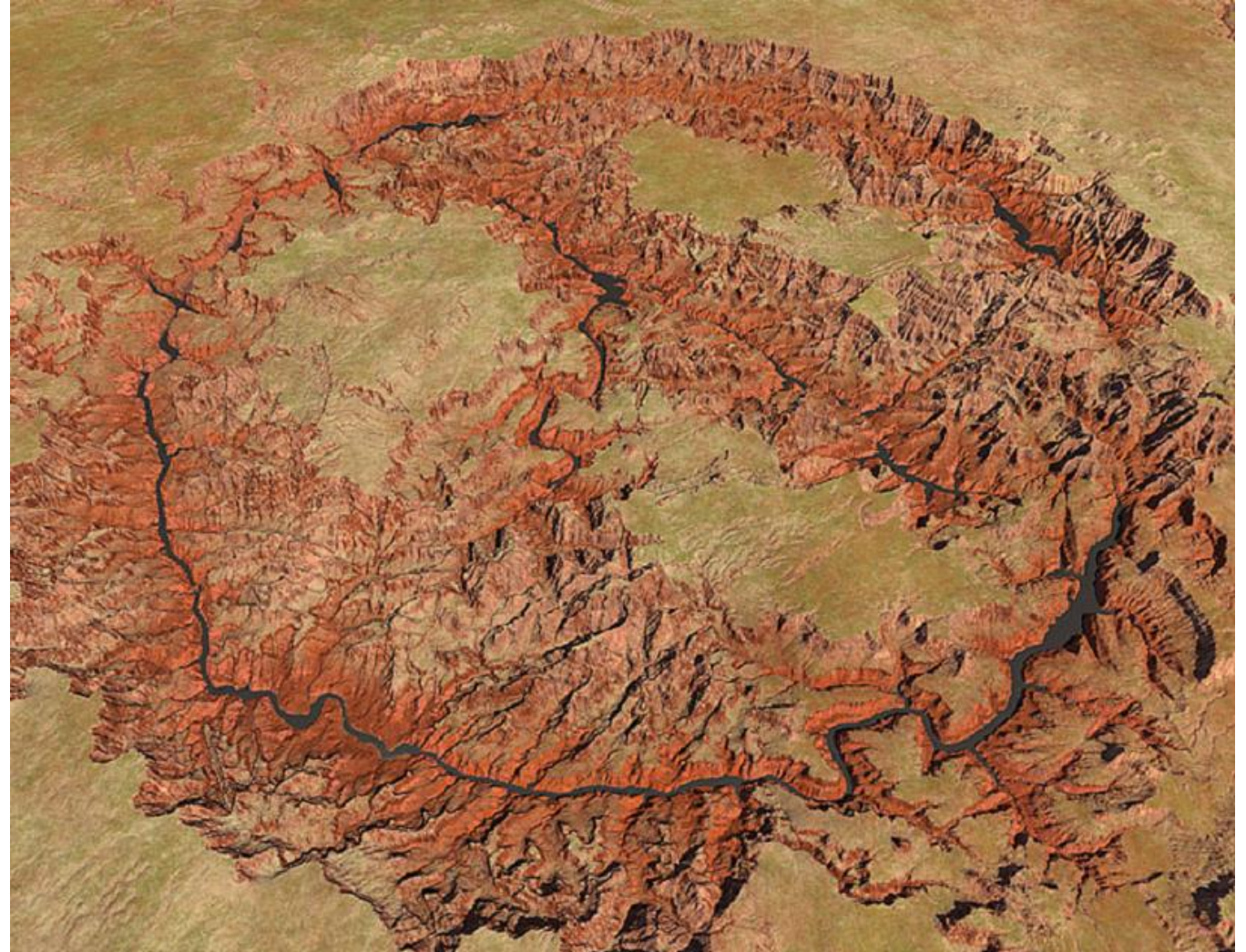
地形 (height field) の合成 [Zhou07]



実際の地形データ

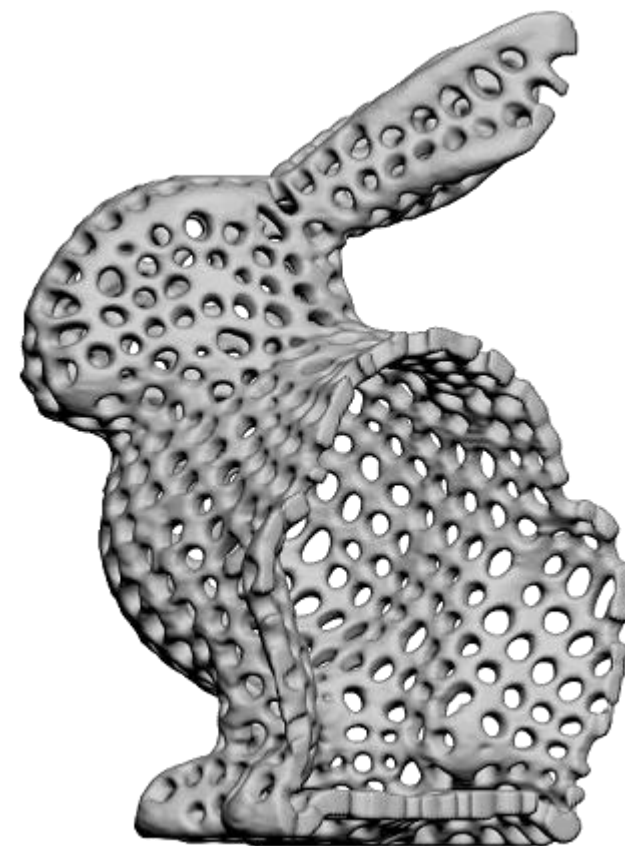
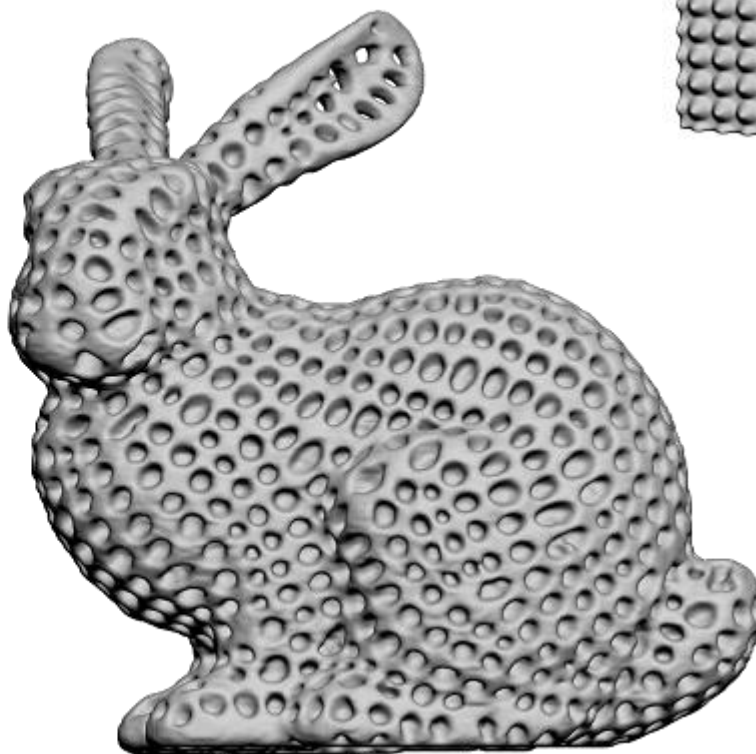


ユーザのスケッチ



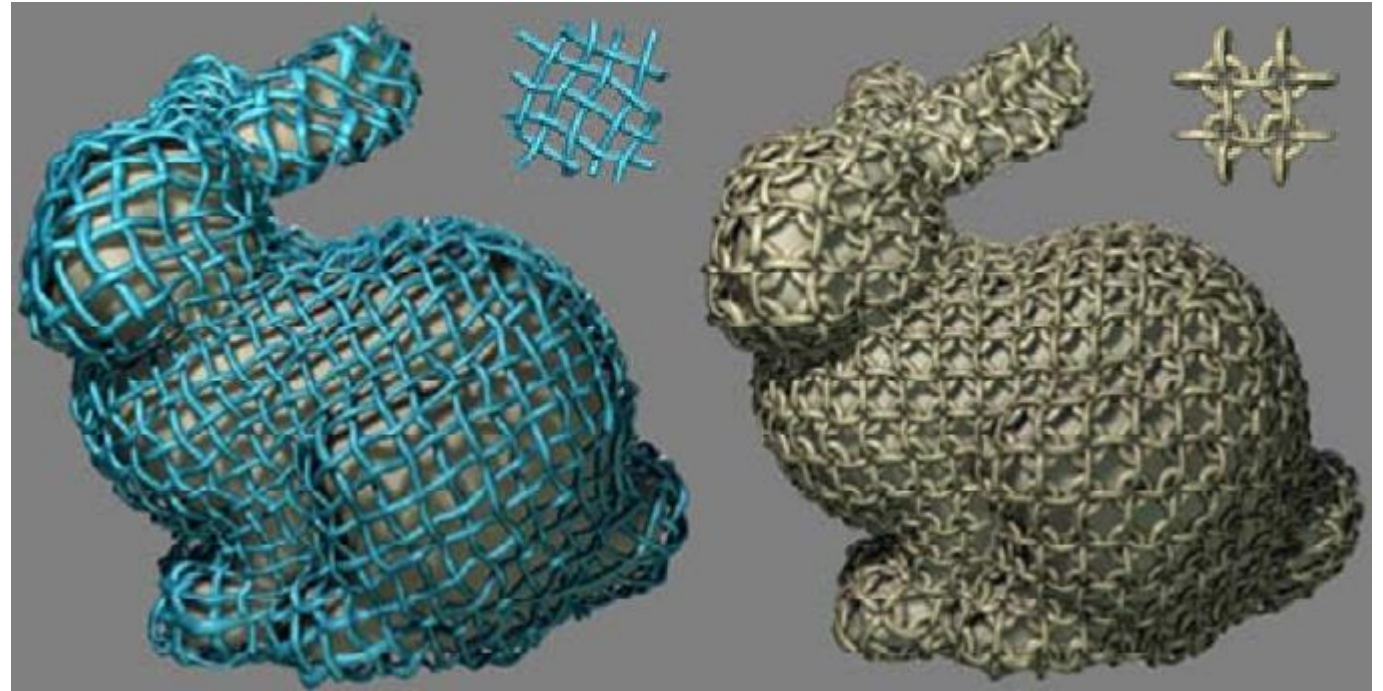
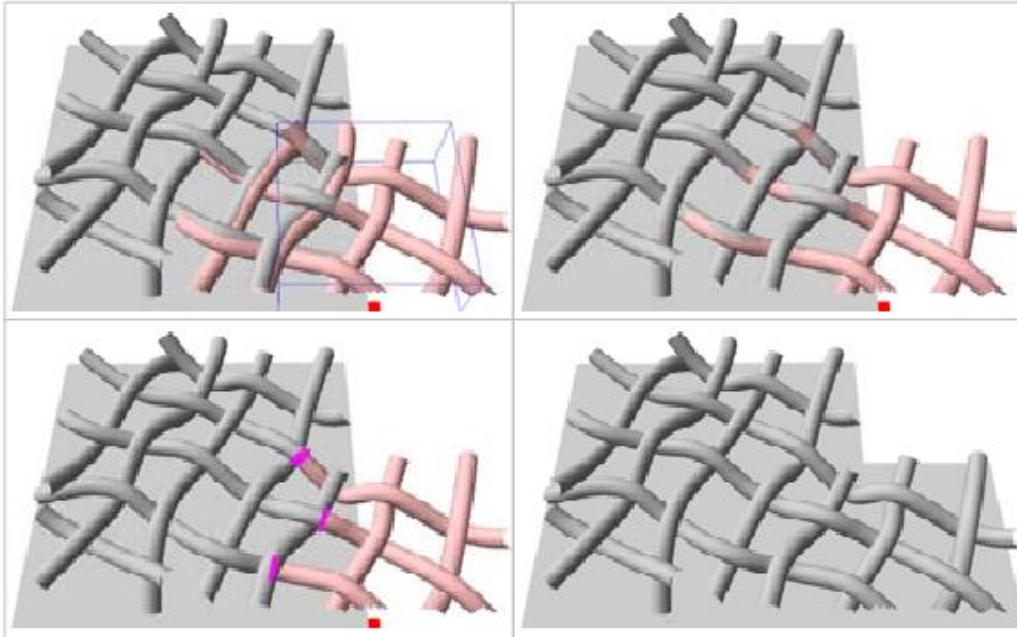
合成結果

Surface Details の合成 [Bhat04]



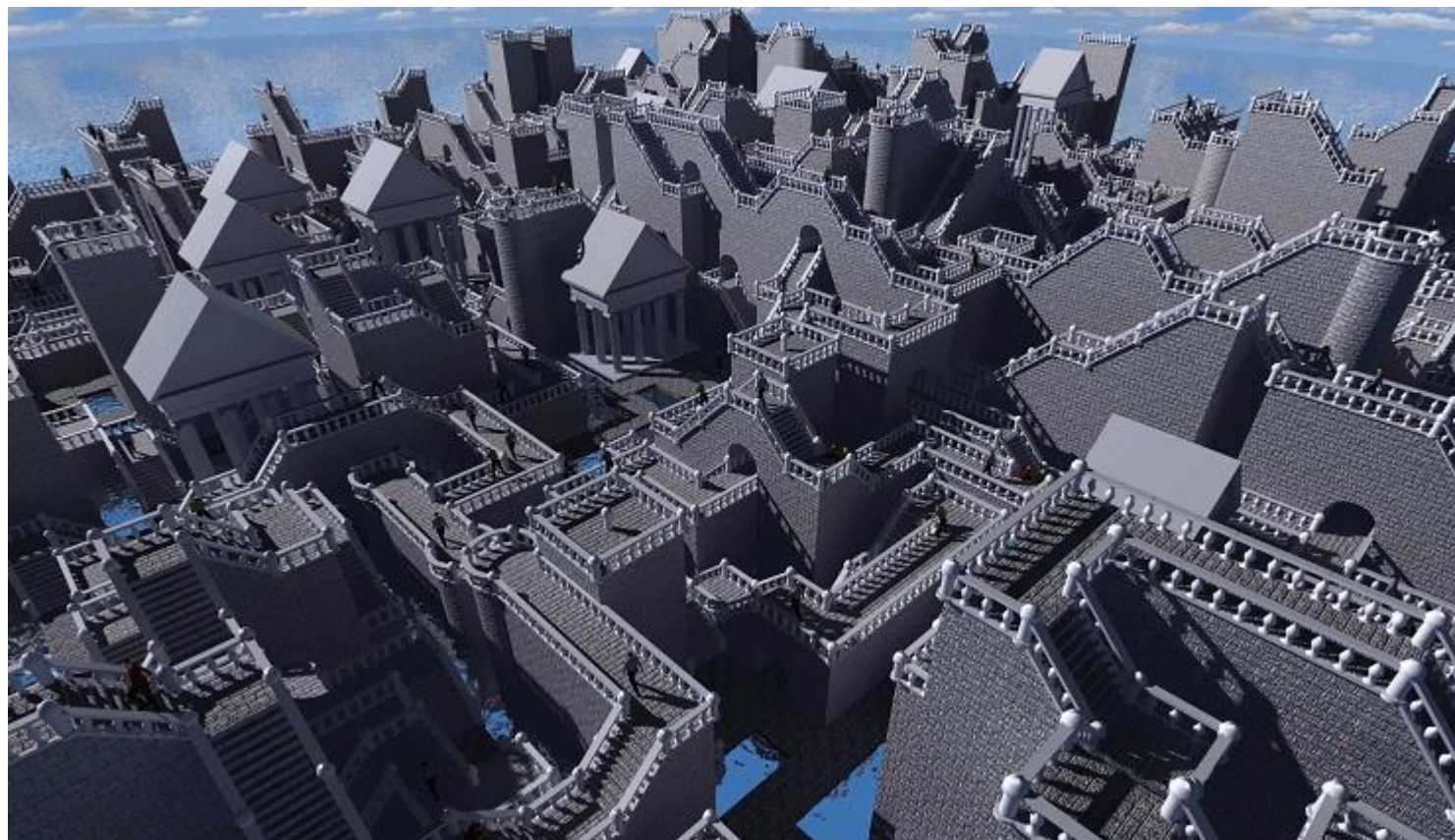
- ボリュームに対して3Dテクスチャ合成
→ height field でない形状も扱える

Mesh Quilting [Zhou06]

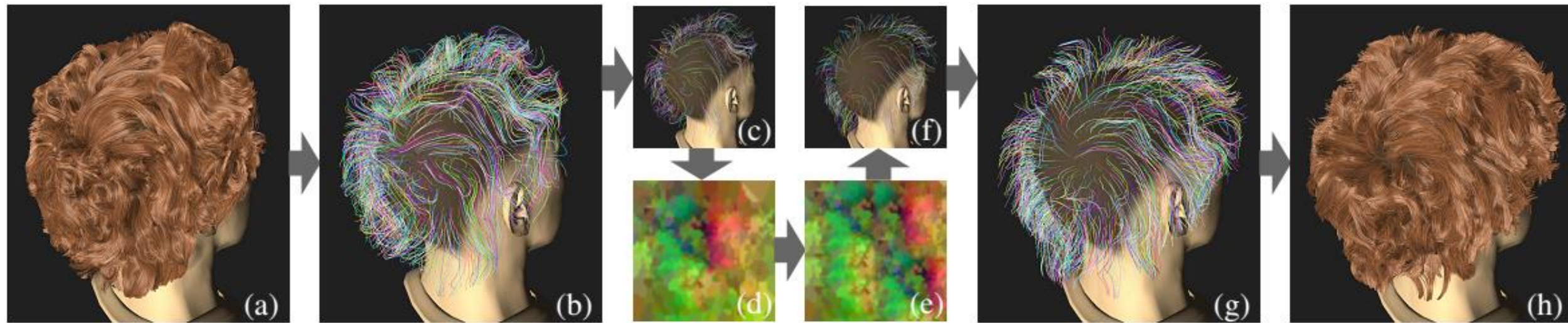


- 三角形メッシュ頂点を頑張って繋ぎ合わせる

人工物モデルの合成 [Merrell07]



頭髪形状の合成 [Wang09]



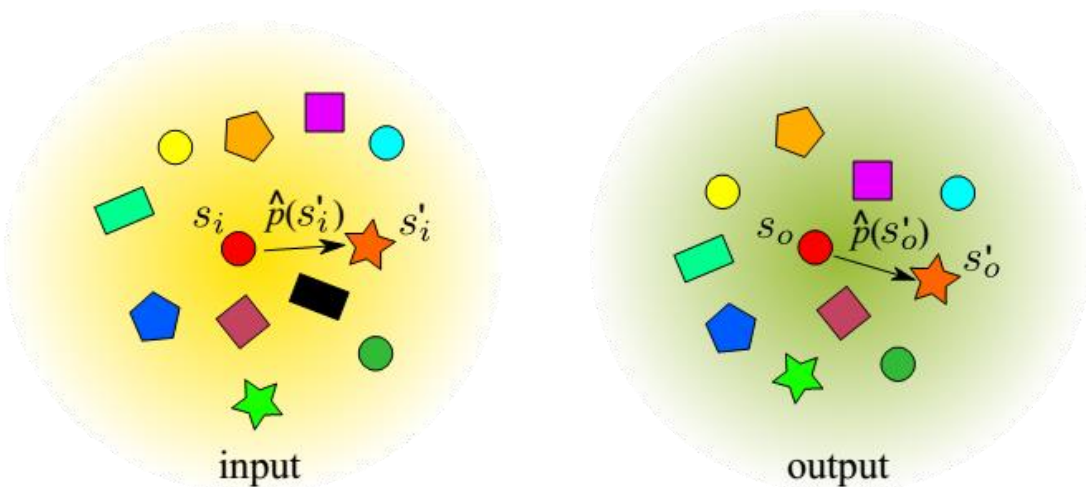
- 一本の髪は N 個の頂点から成る折れ線 = $3N$ 次元ベクトル
➔ これを色だと思なしてテクスチャ合成

流体の渦の合成 [Ma09]

- 低解像度の速度場に沿って、細かい渦の速度場をテクスチャ合成
 - 速度ベクトルを色だと見なす

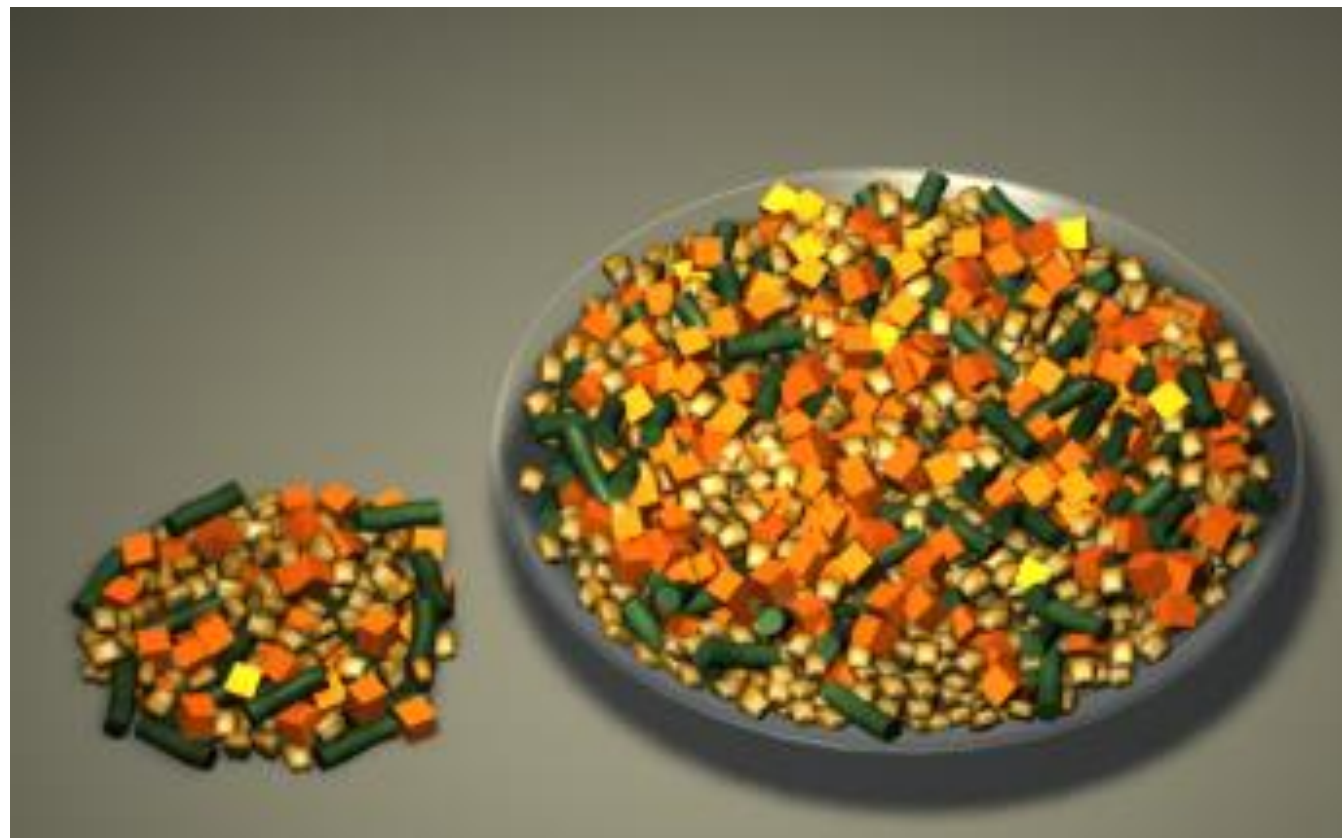


要素配置の合成 [Ma11]



- サンプル点の分布に対して近傍類似度を定義

- [Kwatra05] と似た最適化アルゴリズム



参考情報

- 実装

- <http://www2.mta.ac.il/~tal/ImageCompletion/ImageCompletion1.01.zip>
- http://www.cs.princeton.edu/gfx/pubs/Barnes_2009_PAR/patchmatch-2.1.zip
- https://haxelion.eu/project/Patchmatch_Open_Source_Implementation/
- <http://research.nii.ac.jp/~takayama/cggems12/cggems12.zip>

- サーベイ

- State of the art in example-based texture synthesis [Wei EG09STAR]
- Solid-Texture Synthesis; A Survey [Pietroni CGA10]

- 書籍

- Computer Graphics Gems JP 2012
 - Chapter 6: 画像からのソリッドテクスチャ合成