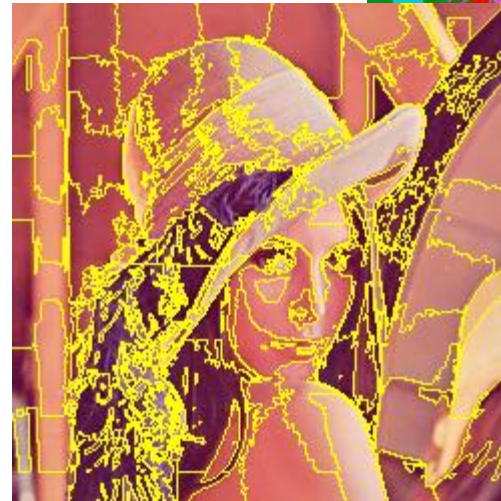
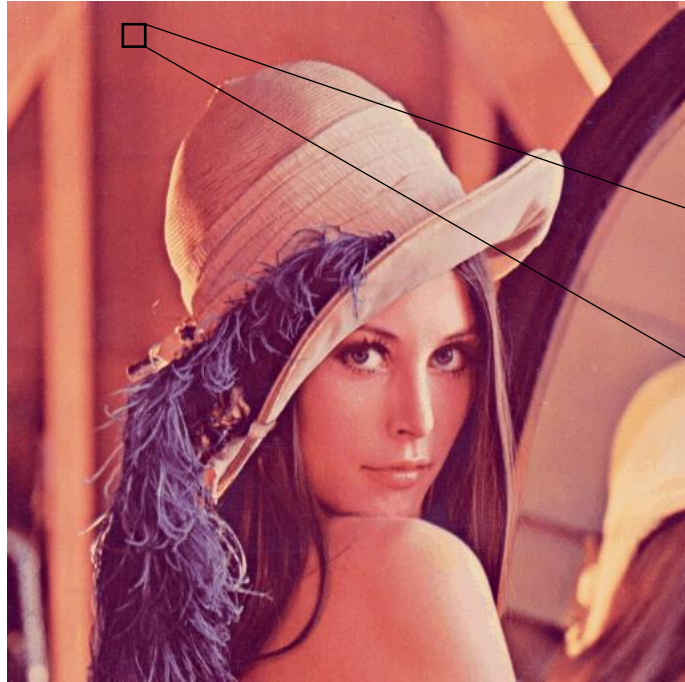


スーパーピクセル解説



スーパーピクセルって何？



512[px]×512[px]

1ピクセル



ピクセルを何らかのまとまりで扱いたい

スーパーピクセル

画像の色や低レベルの特徴の
類似性からグループ化したもの

262144個のピクセルを扱う必要



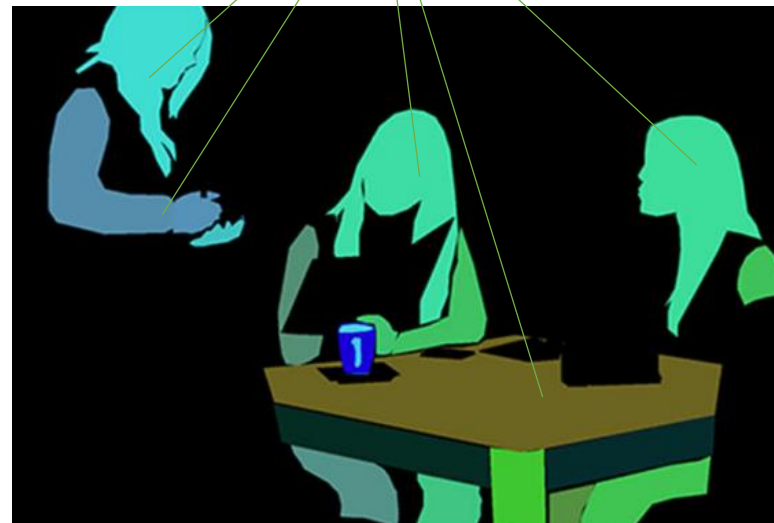
とても大変…

セグメンテーションすることで
スーパーピクセルを生成できる

セグメンテーションとは？

画像をセグメント（小領域）に分割する手法の総称

セグメント



<https://arxiv.org/ftp/arxiv/papers/1809/1809.10198.pdf>

スーパーピクセルはセグメントの1種

スーパーピクセルの歴史

2003年

「スーパーピクセル」誕生

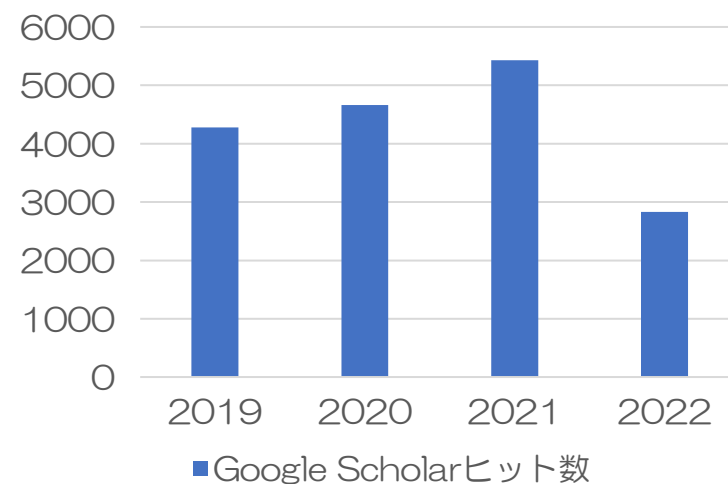
思ったより最近

著者	タイトル	年	会議
X. Ren, J. Malik	<i>Learning a classification model for segmentation</i>	2003	ICCV



現在

現在でも多くの論文が投稿されている



どのアルゴリズムが1番良いの？

著者	タイトル	年	論文
David Stutz, Alexander Hermans, Bastian Leibe	<i>Superpixels: An Evaluation of the State-of-the-Art</i>	2017	ArXiv

<https://github.com/davidstutz/superpixel-benchmark>

スーパーピクセルの28個の手法を比較した論文
GitHub上にコードを公開



代表的なアルゴリズム「SLIC」を解説します～

SLIC

Simple Linear Iterative Clustering

スーパーピクセル生成にk-meansを取り入れた手法

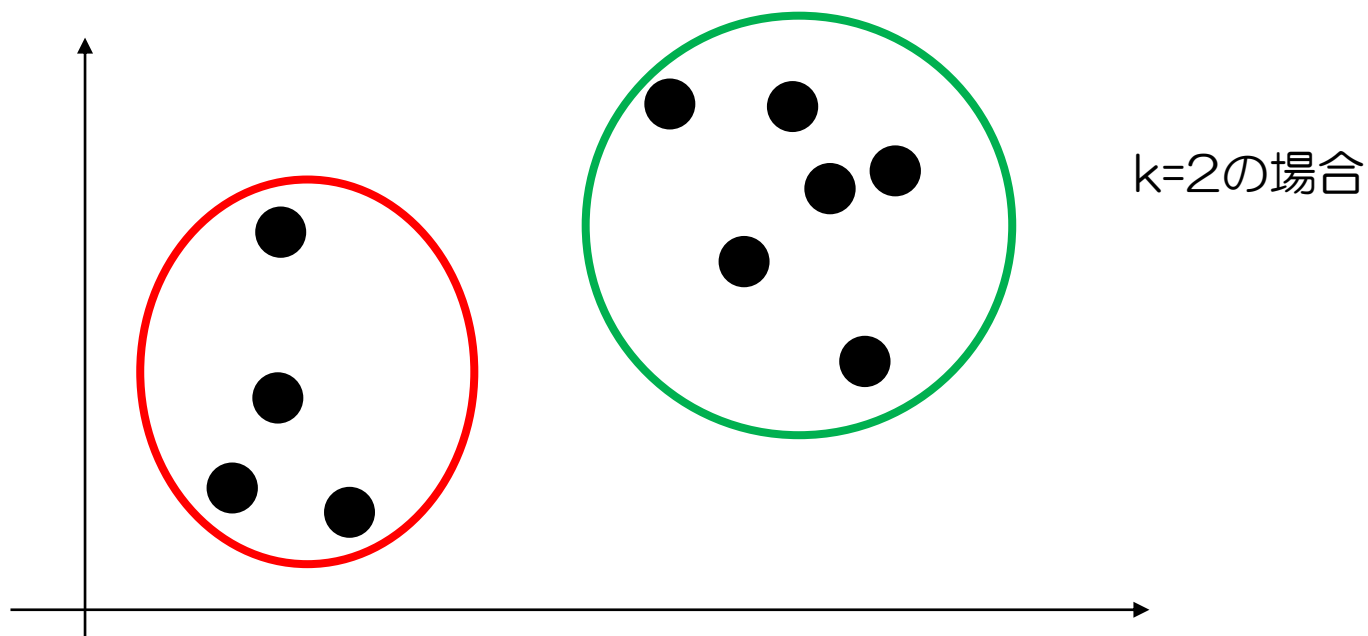
- 色の類似性
 - 画素の近接性
- これらに基づいてクラスタリング

著者	タイトル	年	ジャーナル
R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk	<i>SLIC Superpixels Compared to State-of-the-Art Superpixel Methods</i>	2012	IEEE Transactions on Pattern Analysis and Machine Intelligence

k-means(k平均法)

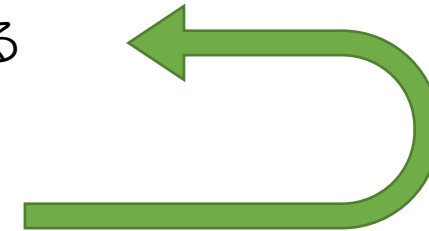
- 教師なし学習の代表的な手法
- クラスタリング → 複数のデータをクラスタ（集団）に分ける

k-means : **k個**のクラスタにわけ（kの値は人間が設定）



k-meansのアルゴリズム

- ① ランダムにk個の代表点を決定
- ↓
② それぞれの代表点に近い点をその代表点のクラスタにする
- ↓
③ 各クラスタの重心を計算、その重心に代表点を移動

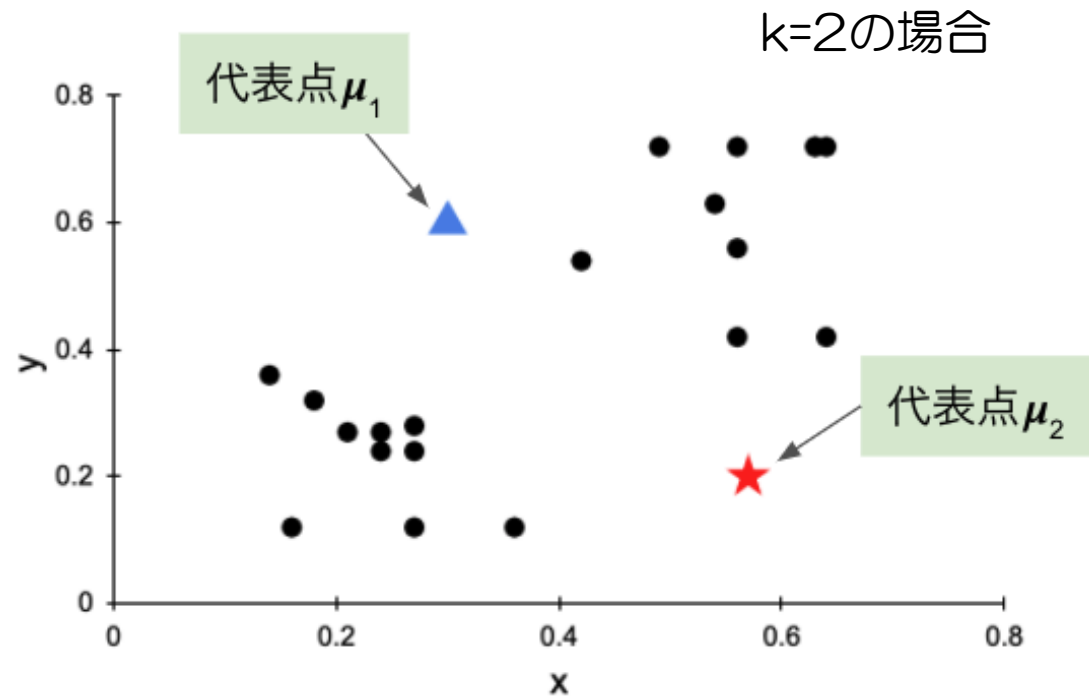


繰り返す

代表点の位置が変化しなくなったら処理終了

k-meansのアルゴリズム

1 ランダムにk個の代表点を決定



k-meansのアルゴリズム

2 それぞれの代表点に近い点をその代表点のクラスタにする

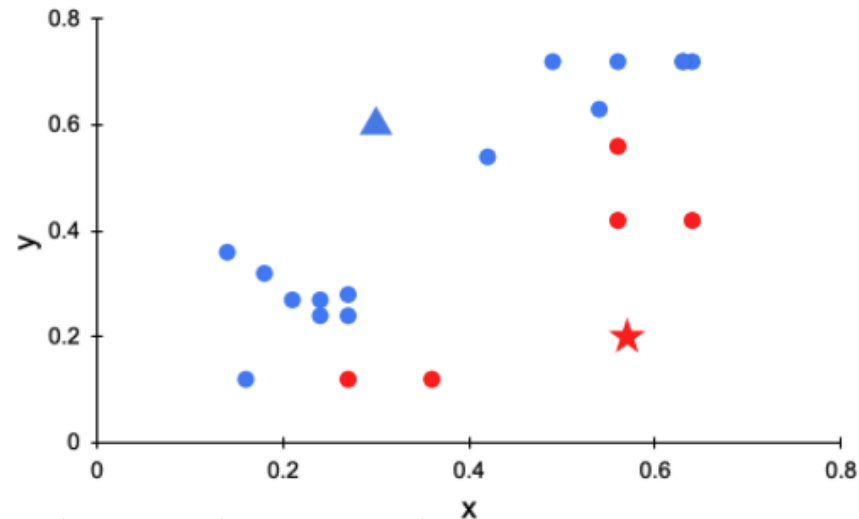
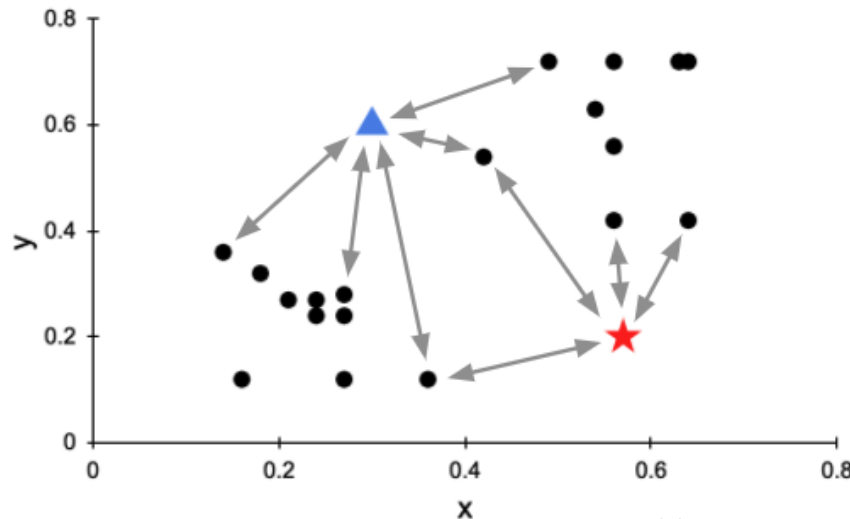
ユークリッド距離

k=2の場合

距離 = $\|X_n - \mu_k\|$ を求める

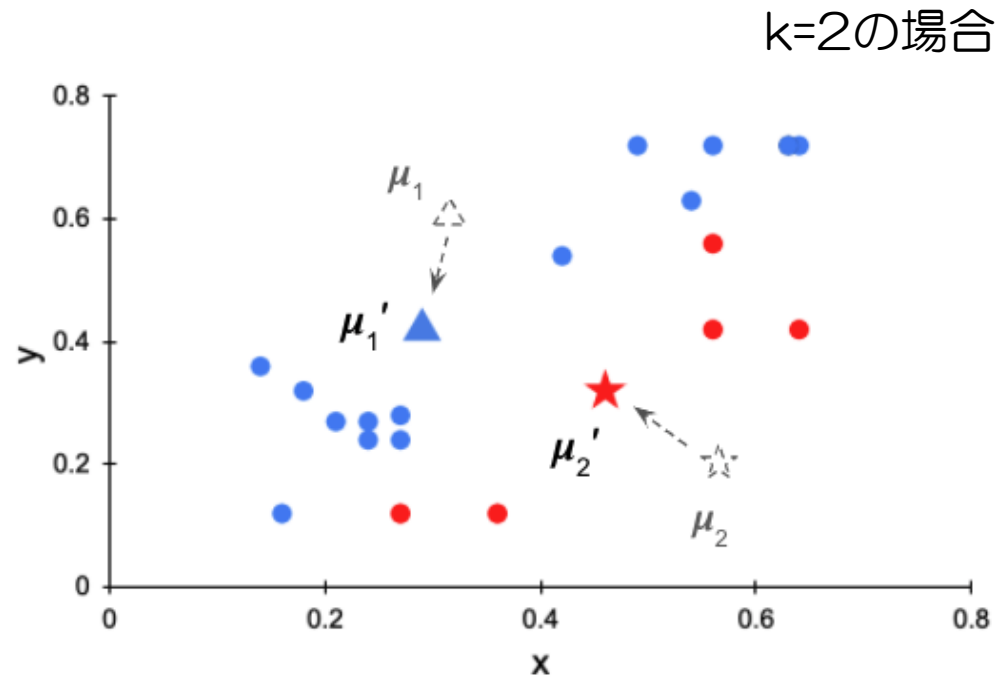


距離が近い方の代表点に所属させる



k-meansのアルゴリズム

3 各クラスターの重心を計算、その重心に代表点を移動



重心の計算は平均をとる
だからk-means

$$\mu_k = \frac{\sum X_n}{N_n}$$

<https://cloud-ace.jp/column/detail291/>

<https://cloud-ace.jp/column/detail291/>

k-means応用

減色処理

k=3と設定すると、3色だけになる



本題に戻ります

SLIC

Simple Linear Iterative Clustering

スーパーピクセル生成にk-meansを取り入れた手法

- 色の類似性
 - 画素の近接性
- これらに基づいてクラスタリング

著者	タイトル	年	ジャーナル
R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk	<i>SLIC Superpixels Compared to State-of-the-Art Superpixel Methods</i>	2012	IEEE Transactions on Pattern Analysis and Machine Intelligence

SLIC アルゴリズム

① RGB画像をLab画像に変換



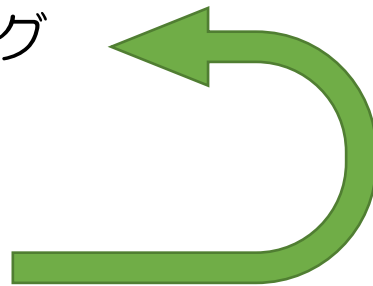
② 初期化



③ 局所クラスタリング



④ 重心を移動



繰り返す

Algorithm 1 Efficient superpixel segmentation

```
1: Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$  by sampling pixels at regular grid steps  $S$ .
2: Perturb cluster centers in an  $n \times n$  neighborhood, to the lowest gradient position.
3: repeat
4:   for each cluster center  $C_k$  do
5:     Assign the best matching pixels from a  $2S \times 2S$  square neighborhood around the cluster center according to the distance measure (Eq. 1).
6:   end for
7:   Compute new cluster centers and residual error  $E$  {L1 distance between previous centers and recomputed centers}
8: until  $E \leq \text{threshold}$ 
9: Enforce connectivity.
```

SLIC アルゴリズム

1 RGB画像をLab画像に変換

Lab画像は知覚的均等性をもつ

色が変化しただけ人間も色が変わったと感じる

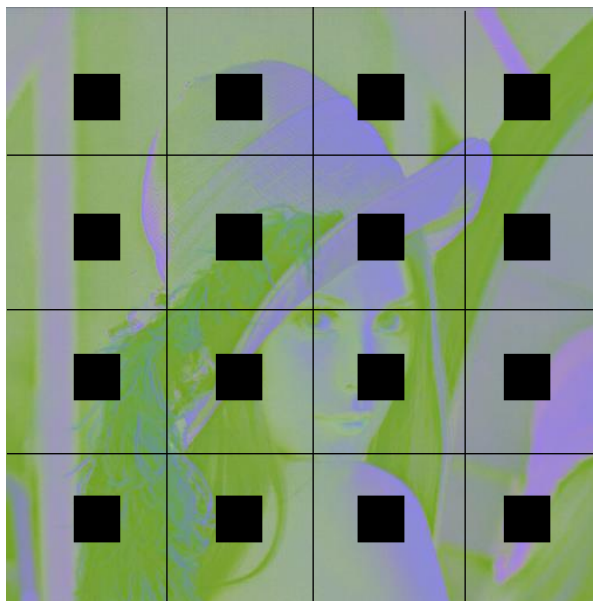
5次元空間 $[l, a, b, x, y]$ とする l, a, b は色情報、 x と y は画素の位置



SLIC アルゴリズム

2 初期化

まず、 k 個のクラスタの重心位置を等間隔で配置 クラスタの重心 $C_k = [l_k, a_k, b_k, x_k, y_k]^T$



ほぼ同じ大きさのスーパーピクセルを生成する
そのため、

スーパーピクセルの大きさは $\frac{N}{k}$ ※ N はピクセル数

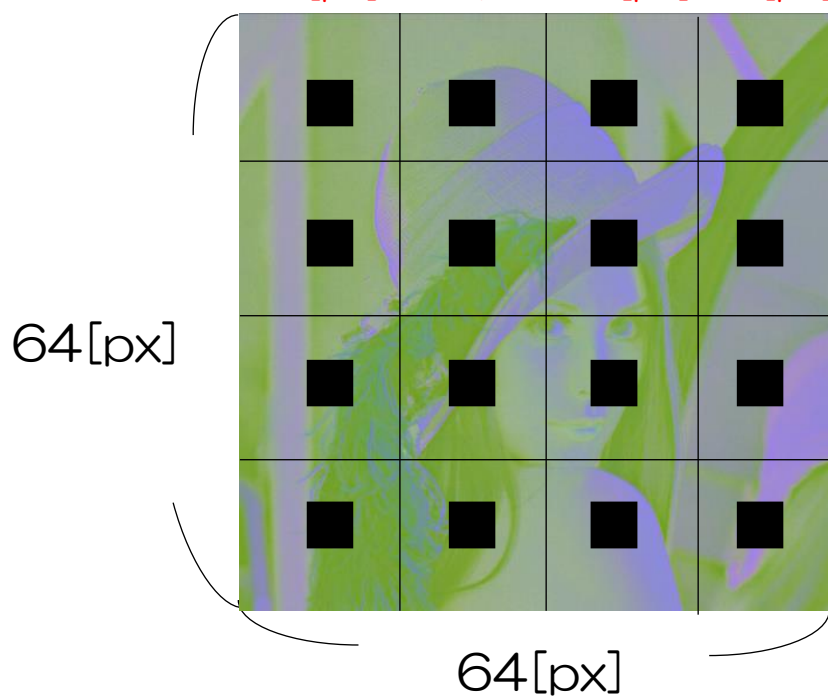
グリッドの間隔 $S = \sqrt{\frac{N}{k}}$

SLIC アルゴリズム

2 初期化

まず、 k 個のクラスタの重心位置を等間隔で配置

16[px] 16[px] 16[px] 16[px]



例) ピクセル数 $N = 64 * 64 = 4096$

クラスタ数 $k = 16$ とする。

1つのスーパーピクセルあたり、 $\frac{N}{k} = \frac{4096}{16} = 256$

256の大きさだと画像全体をスーパーピクセルで埋めれる

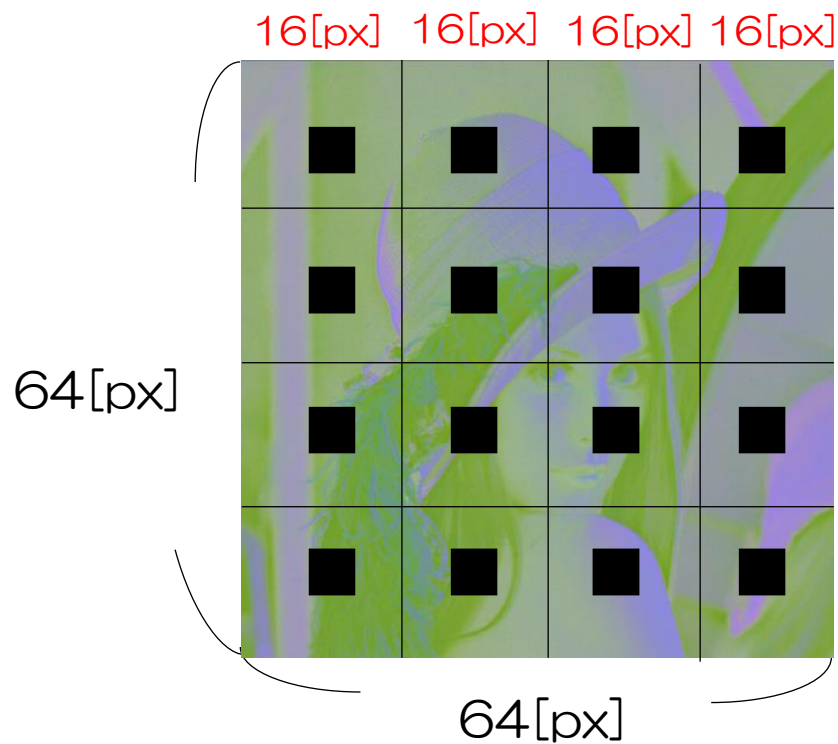
グリッドの間隔 $S = \sqrt{\frac{N}{k}} = 16$

SLIC アルゴリズム

2 初期化

エッジやノイズの多い画素が
重心になることを避けるため

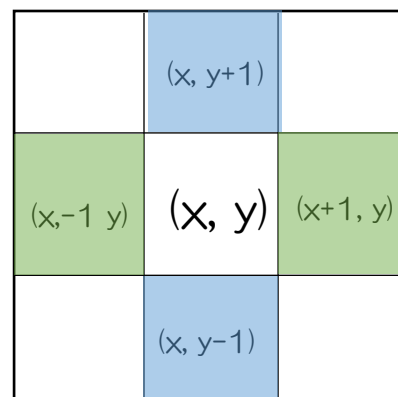
次に、各重心の3×3近傍を参照して勾配が最も小さい位置に重心を移動



3×3、つまり周囲9ピクセルの勾配を計算

座標(x,y)における勾配

$$G(x, y) = \|\mathbf{I}(x + 1, y) - \mathbf{I}(x - 1, y)\|^2 + \|\mathbf{I}(x, y + 1) - \mathbf{I}(x, y - 1)\|^2$$



$\mathbf{I}(x, y)$ は座標(x,y)におけるLabベクトル

L2ノルム（ユークリッド距離）で計算

SLIC アルゴリズム

2 初期化

エッジやノイズの多い画素が
重心になることを避けるため

次に、各重心の3×3近傍を参照して勾配が最も小さい位置に重心を移動

座標(x,y)における勾配

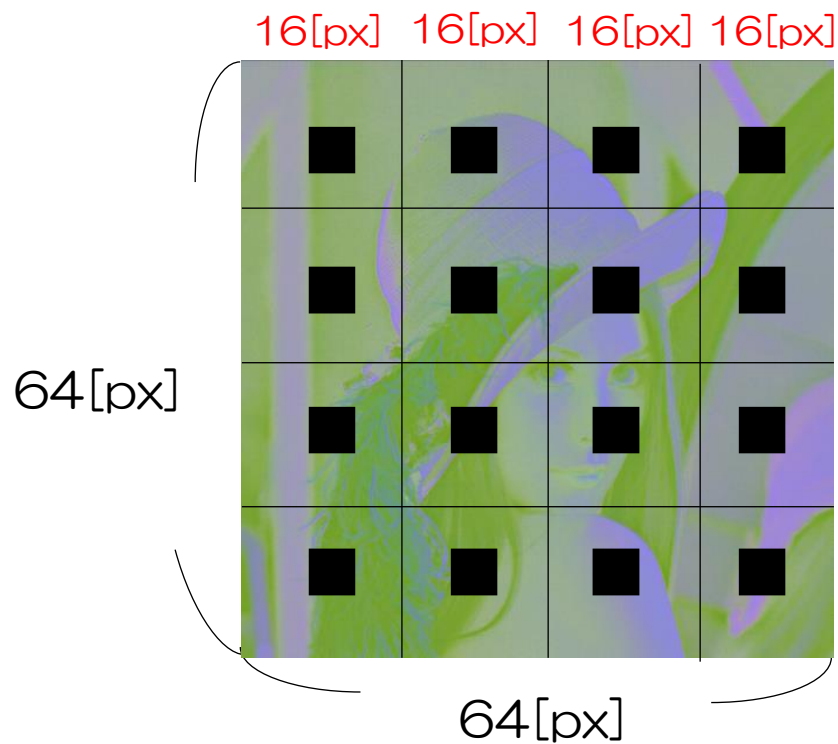
$$G(x, y) = \|\mathbf{I}(x + 1, y) - \mathbf{I}(x - 1, y)\|^2 + \|\mathbf{I}(x, y + 1) - \mathbf{I}(x, y - 1)\|^2$$

例)

1.5	1.9	3.5
0.45	重心 0.85	最小 0.11
0.78	0.88	0.21

移動

1.5	1.9	3.5
0.45	0.85	重心 0.11
0.78	0.88	0.21

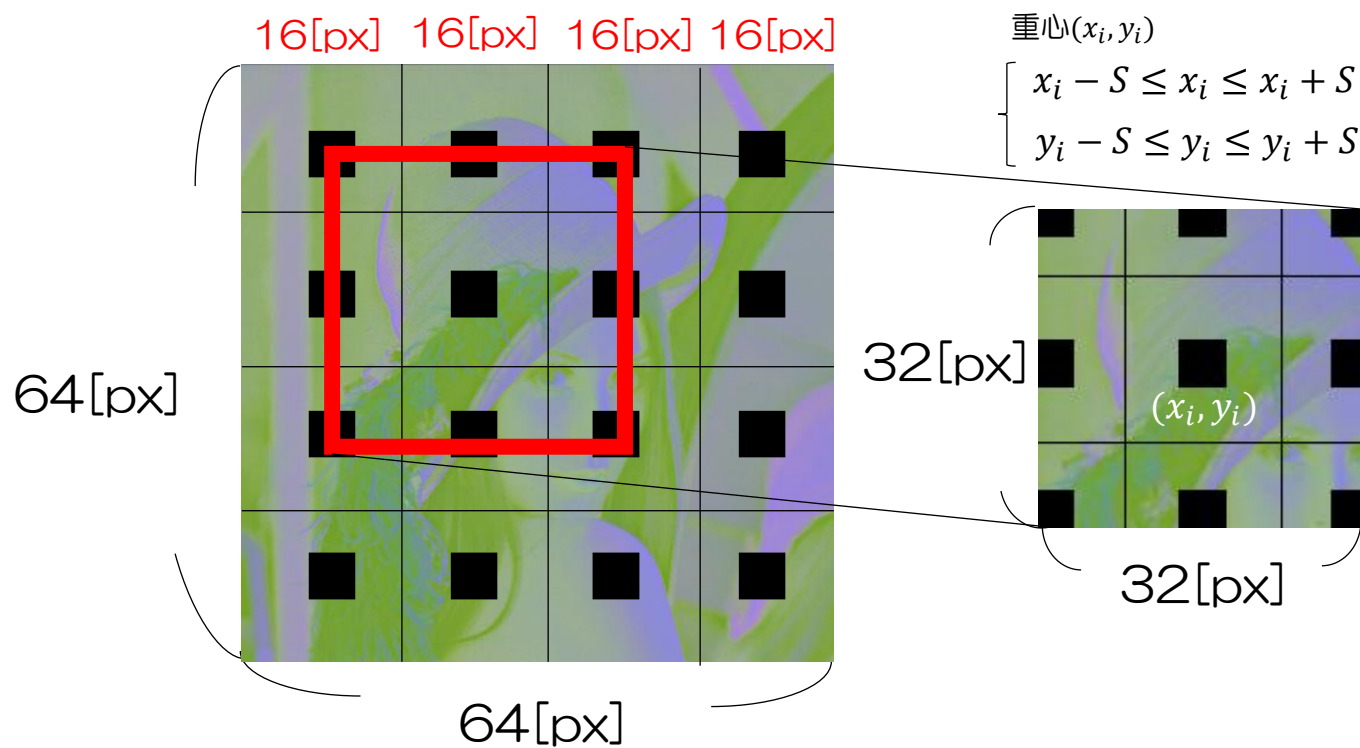


SLIC アルゴリズム

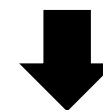
3 局所クラスタリング

クラスタリング計算の対象範囲は $2S \times 2S$ の範囲

グリッドの間隔 $S = \sqrt{\frac{N}{k}} = 16$



この範囲の画素とのみ比較



局所クラスタリング
計算量を削減

k-means	SLIC	N:ピクセル数
$O(Nkl)$	$\rightarrow O(N)$	k:クラスタ数
		l:反復回数

※k-meansは対象範囲を絞らず
画像全体に対して比較を行う

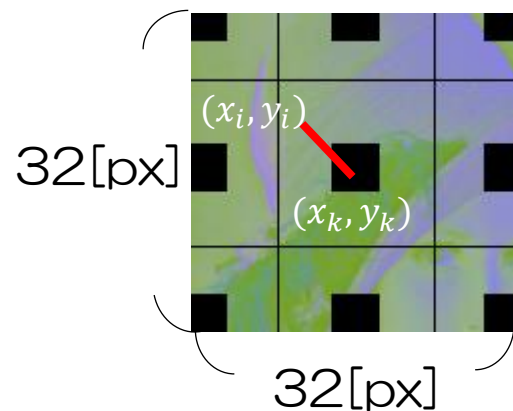
SLIC アルゴリズム

3 局所クラスタリング

距離の計算

重心 (x_i, y_i)

$$\begin{cases} x_i - S \leq x_i \leq x_i + S \\ y_i - S \leq y_i \leq y_i + S \end{cases}$$



$$\text{グリッドの間隔 } S = \sqrt{\frac{N}{k}} = 16$$

5次元空間のため、**単純なユークリッド距離で計算できない**
画素の近接性、色の類似性を計算する必要がある

クラスタの重心 $C_k = [l_k, a_k, b_k, x_k, y_k]^T$

範囲内の点 i $C_i = [l_i, a_i, b_i, x_i, y_i]^T$

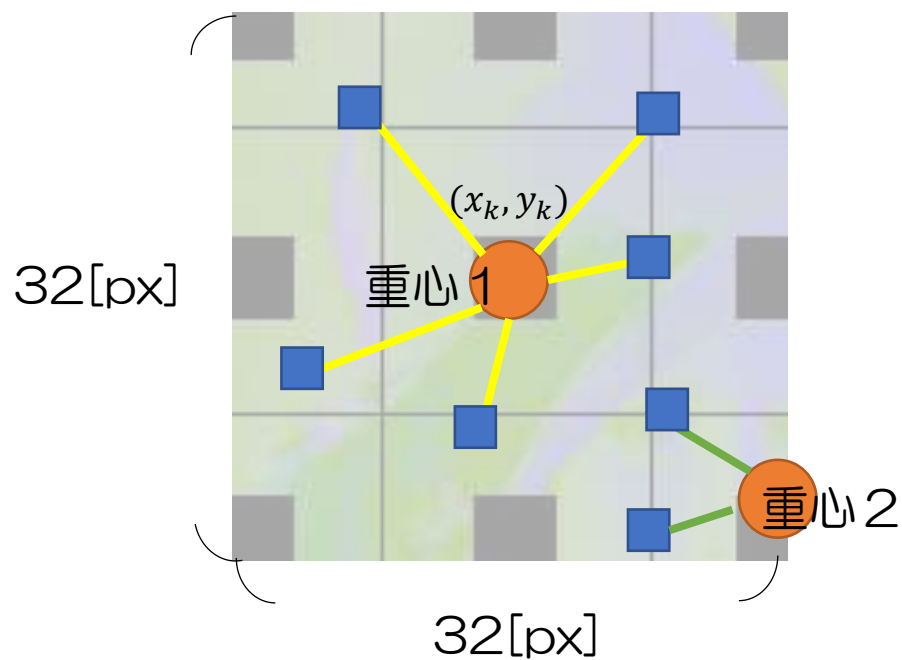
スーパーピクセルサイズを考慮した距離尺度

$$\begin{cases} d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} & \text{色の類似性} \\ d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} & \text{画素の近接性} \\ D_s = d_{lab} + \frac{m}{S} d_{xy} & \begin{array}{l} m: \text{色と位置の情報の比率係数} \\ S: \text{グリッド間隔 (距離xyを正規化するため)} \end{array} \end{cases}$$

m が大きいほど画素の近接性を、 m が小さいほど色の類似性を強調
 m は $[1, 20]$ の範囲。論文では、 **$m=10$** としている。

SLIC アルゴリズム

3 局所クラスタリング

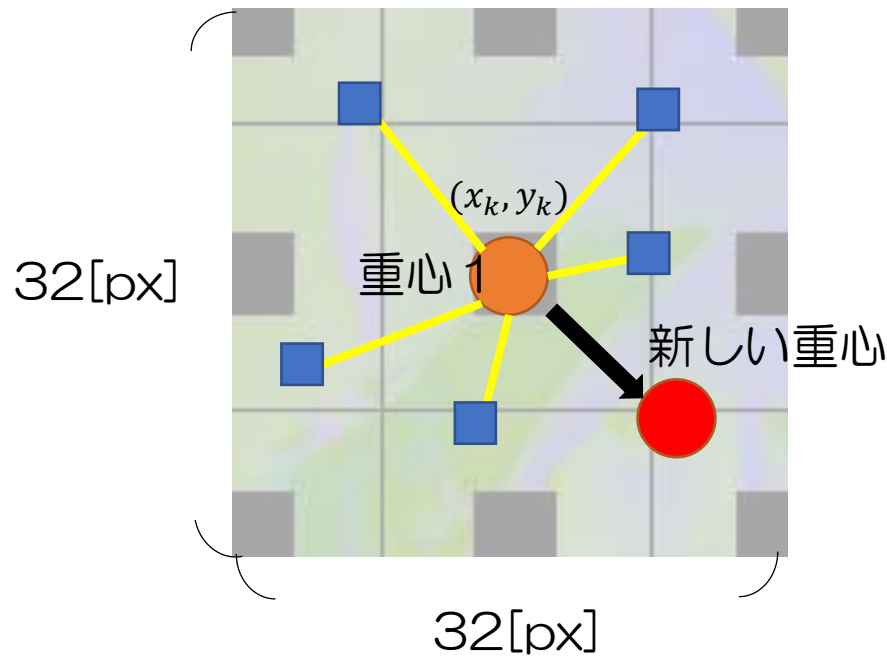


計算した距離 D_s を用いて各画素がどのクラスに属するかを判定

➡ k-meansと同じ

SLIC アルゴリズム

4 重心の移動



クラスタに属している画素のlabxyベクトルの平均（重心）を計算して新たな重心を求める

➡ k-meansと同じ

【処理終了条件】 条件1または条件2を満たす時、処理終了

1. 前の重心と計算して移動した重心のマンハッタン距離の誤差Eが閾値より下回る
2. 10回処理を繰り返す（イテレーション数が10）

イテレーション数10は経験的に求められた値

それでは実装してみましよう

SLIC アルゴリズム

1 RGB画像をLab画像に変換



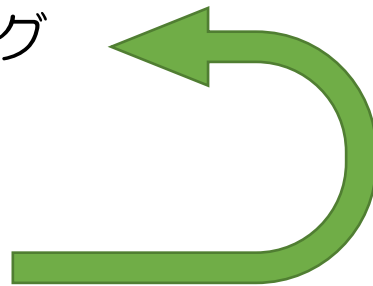
2 初期化



3 局所クラスタリング



4 重心を移動



繰り返す

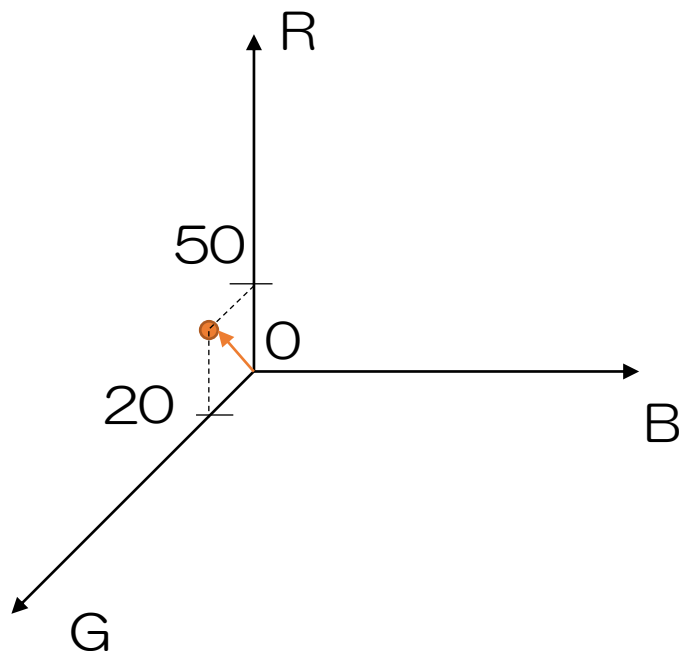
Algorithm 1 Efficient superpixel segmentation

```
1: Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$  by sampling pixels at regular grid steps  $S$ .
2: Perturb cluster centers in an  $n \times n$  neighborhood, to the lowest gradient position.
3: repeat
4:   for each cluster center  $C_k$  do
5:     Assign the best matching pixels from a  $2S \times 2S$  square neighborhood around the cluster center according to the distance measure (Eq. 1).
6:   end for
7:   Compute new cluster centers and residual error  $E$  {L1 distance between previous centers and recomputed centers}
8: until  $E \leq \text{threshold}$ 
9: Enforce connectivity.
```

SLICを扱うにあたって 事前に必要な周辺知識

色は座標として捉えることができる

例) R=50, G=20, B=0



色空間

(赤、緑、青)

RGB

表示用の色空間
sRGB, Adobe RGB など

(シアン、マゼンタ、黄、黒)

CMYK

印刷用の色空間

値を変更すると彩度と明度が同時に変化

(色相、彩度、明度)

HSV

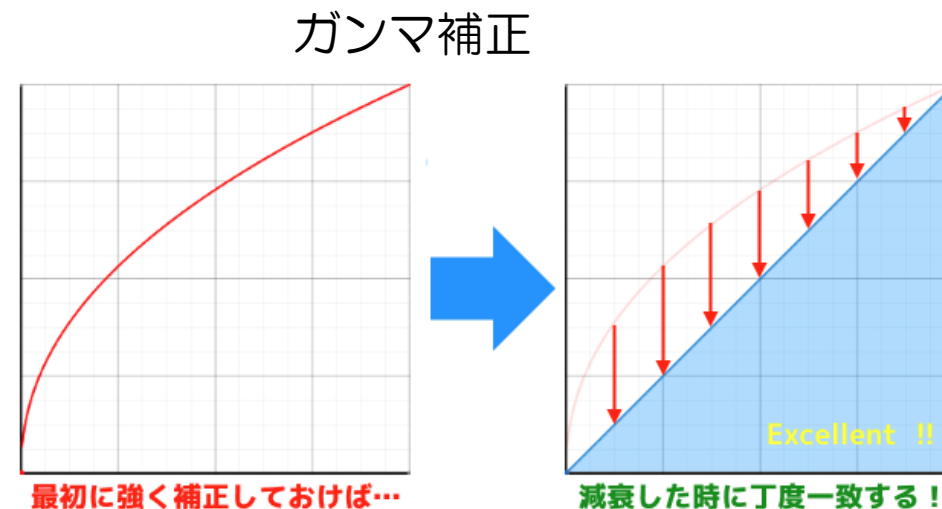
画像処理でよく利用される色空間
彩度を変えずに明度だけ変更可能
人間の知覚に近い

Lab

XYZ色空間を人間の知覚に近づけた色空間
均等色空間
色差、色ズリの検知に利用される色空間

非線形RGBと線形RGB

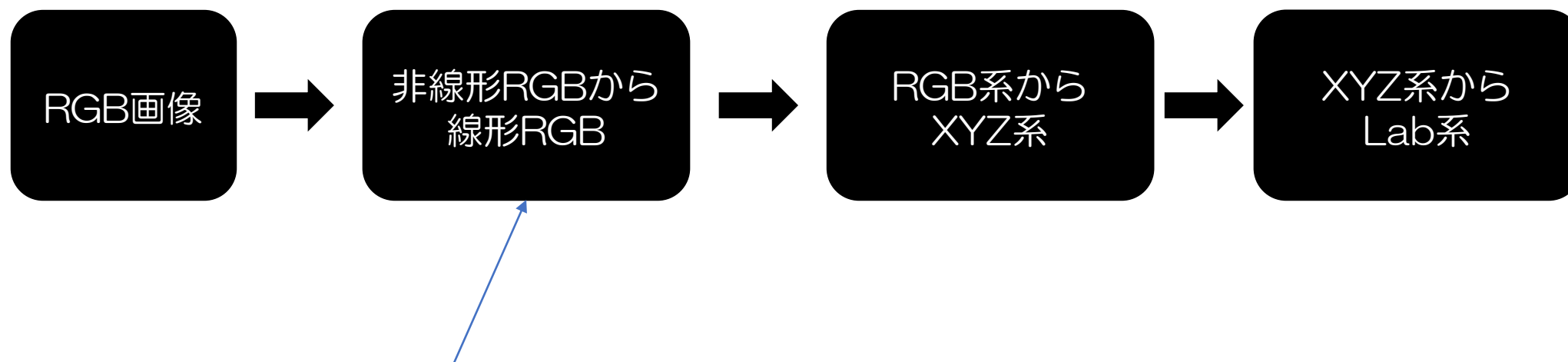
- 非線形RGB (sRGB)
画面出力の際の減衰を考慮してガンマ補正をかけたRGB色空間
- 線形RGB
ガンマ補正をかけていないRGB色空間



<https://tech.cygames.co.jp/archives/2296/>

RGB画像をCIE-XYZ表色系の色空間で処理したい場合には
非線形RGBから線形RGBに変換する処理を行う

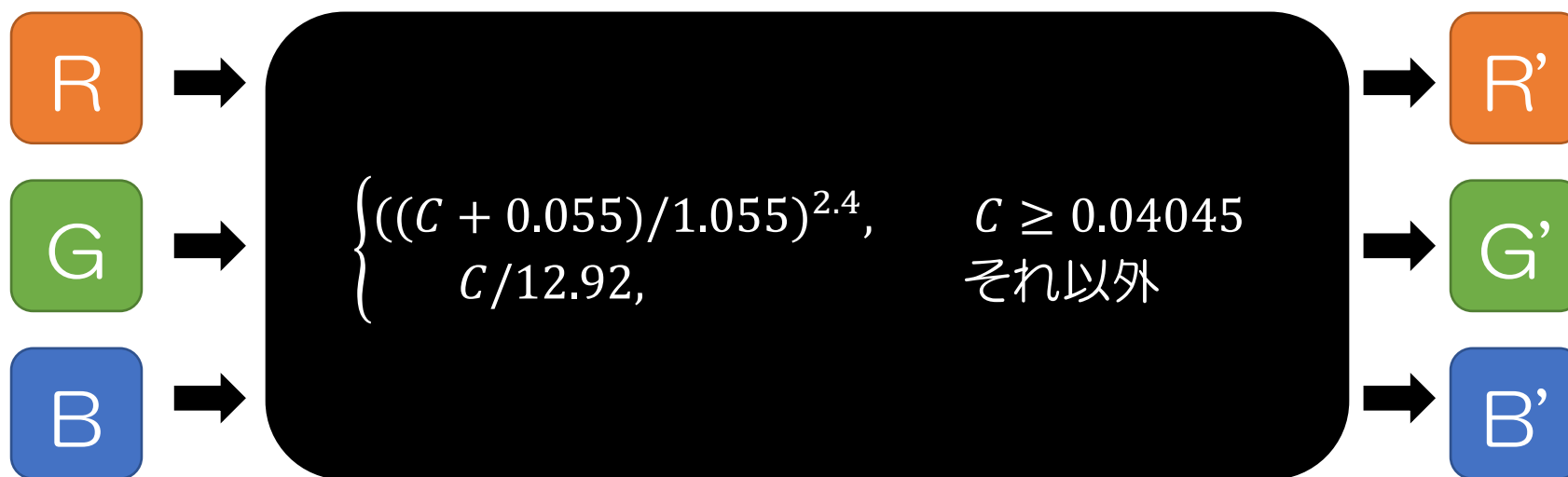
RGB画像からLab画像に変換



RGB画像をCIE-XYZ表色系の色空間で処理したい場合には
非線形RGBから線形RGBに変換する処理を行う

非線形RGBから線形RGBに変換

参考： https://www.jstage.jst.go.jp/article/itej/71/5/71_306/_pdf



【注意】

RGB値は[0,1]の範囲
CはRGBの画素値



正規化が必要

例) $R=0.26$ ならば、 $C = 0.26$ 。 R' は $0.26/12.92$ で計算可能

RGB表色系からXYZ表色系に変換

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9504 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

XYZ表色系からLab表色系に変換

1 完全拡散反射面の三刺激値を加味

$$X' \leftarrow X / X_n \quad X_n = 0.950456 \quad \longleftrightarrow \quad X' \leftarrow \frac{X}{95.0456} * 100$$

$$Y' \leftarrow Y / Y_n \quad Y_n = 1 \quad \longleftrightarrow \quad Y' \leftarrow \frac{Y}{100.0} * 100$$

$$Z' \leftarrow Z / Z_n \quad Z_n = 1.088754 \quad \longleftrightarrow \quad Z' \leftarrow \frac{Z}{108.8754} * 100$$

XYZ表色系からLab表色系に変換

2 変換

$$0 \leq L \leq 100 L \leftarrow \begin{cases} 116 * Y'^{\frac{1}{3}} - 16, & Y' > 0.008856 \\ 903.3 * Y', & \text{それ以外} \end{cases}$$

$$-127 \leq a \leq 127 \quad a \leftarrow 500 * (f(X') - f(Y'))$$

$$-127 \leq b \leq 127 \quad b \leftarrow 200 * (f(Y') - f(Z'))$$

$$f(t) = \begin{cases} t^{\frac{1}{3}}, & t > 0.008856 \\ 7.787t + \frac{16}{116}, & \text{それ以外} \end{cases}$$

変換完了！

XYZ表色系からLab表色系に変換

③ もしOpenCV準拠にするならば…

OpenCVの範囲値に合わせる

$$0 \leq L \leq 100$$



$$L' = L * 255/100$$

$$0 \leq L' \leq 255$$

$$-128 < a \leq 127$$



$$a' = a + 128$$

$$0 \leq a' \leq 255$$


$$-128 < b \leq 127$$



$$b' = b + 128$$

$$0 \leq b' \leq 255$$

もっと詳しく知りたい方は



Technical Survey

技術解説

画像処理と色空間

(正会員) 平井 経太[†]

色情報は、画像処理や画像解析において、重要な特徴の1つである。本稿では、まず、撮像による色獲得の原理、デバイスディペンデントカラーとデバイスインディペンデントカラー、三原色と反対色について述べる。続いて、各種色空間の特徴と数値計算の方法を紹介する。最後に、各色空間のカラー画像処理・解析への応用とその性能を示す。

キーワード：三原色，反対色，RGB色空間，CIE色空間，カラー画像処理，カラー画像解析

https://www.jstage.jst.go.jp/article/itej/71/5/71_306/_pdf

[L, a, b, x, y]の形式に変更

画素に関する情報は色情報[L, a, b]を持っている
これを画素情報を加えた[L, a, b, x, y]に変更

今までの処理をまとめると



ピクセル情報

(H,W,5)

各ピクセルの
l, a, b, x, y値

SLIC アルゴリズム

1 RGB画像をLab画像に変換



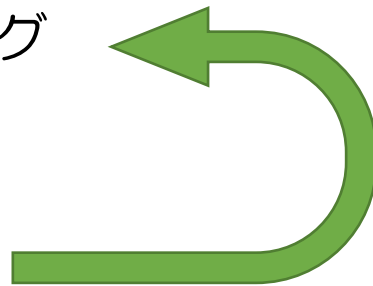
2 初期化



3 局所クラスタリング



4 重心を移動



繰り返す

Algorithm 1 Efficient superpixel segmentation

```
1: Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$  by sampling pixels at regular grid steps  $S$ .
2: Perturb cluster centers in an  $n \times n$  neighborhood, to the lowest gradient position.
3: repeat
4:   for each cluster center  $C_k$  do
5:     Assign the best matching pixels from a  $2S \times 2S$  square neighborhood around the cluster center according to the distance measure (Eq. 1).
6:   end for
7:   Compute new cluster centers and residual error  $E$  {L1 distance between previous centers and recomputed centers}
8: until  $E \leq \text{threshold}$ 
9: Enforce connectivity.
```

初期化

- 1 等間隔で各クラスタの重心を配置
- 2 勾配に基づいて重心の位置を微調整

等間隔で各クラスタの重心を配置

※等間隔で配置するため、人間が指定するクラスタ数とは異なる場合がある

まず、配置できる個数を決定する。

$$\text{グリッドの間隔 } S = \sqrt{\frac{N}{k}} = \sqrt{\frac{HW}{k}}$$

画像幅Wに関して、

幅に対して
どれだけ配置できるか

$$\frac{W}{S} = W * \sqrt{\frac{k}{HW}} = \sqrt{\frac{kW}{H}}$$

画像高さHに関して、

$$\frac{H}{S} = H * \sqrt{\frac{k}{HW}} = \sqrt{\frac{kH}{W}}$$



配置できる個数

$$\left\lceil \sqrt{\frac{kW}{H}} \right\rceil * \left\lceil \sqrt{\frac{kH}{W}} \right\rceil$$

切り上げ（天井関数）

グリッドの列

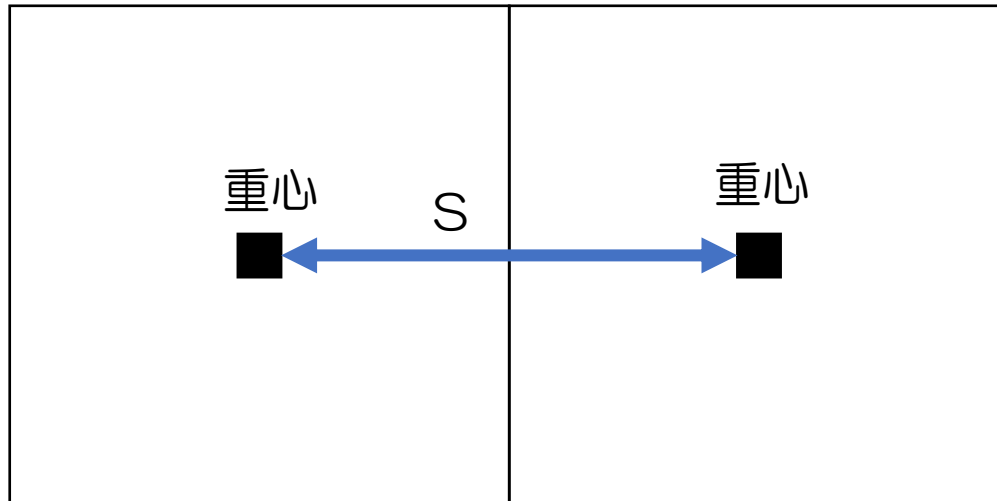
グリッドの行

等間隔で各クラスタの重心を配置

次に、**グリッド間隔を更新**（指定したクラスタ数と異なる可能性が高いため）

$$k = \left\lceil \sqrt{\frac{kW}{H}} \right\rceil * \left\lceil \sqrt{\frac{kH}{W}} \right\rceil$$

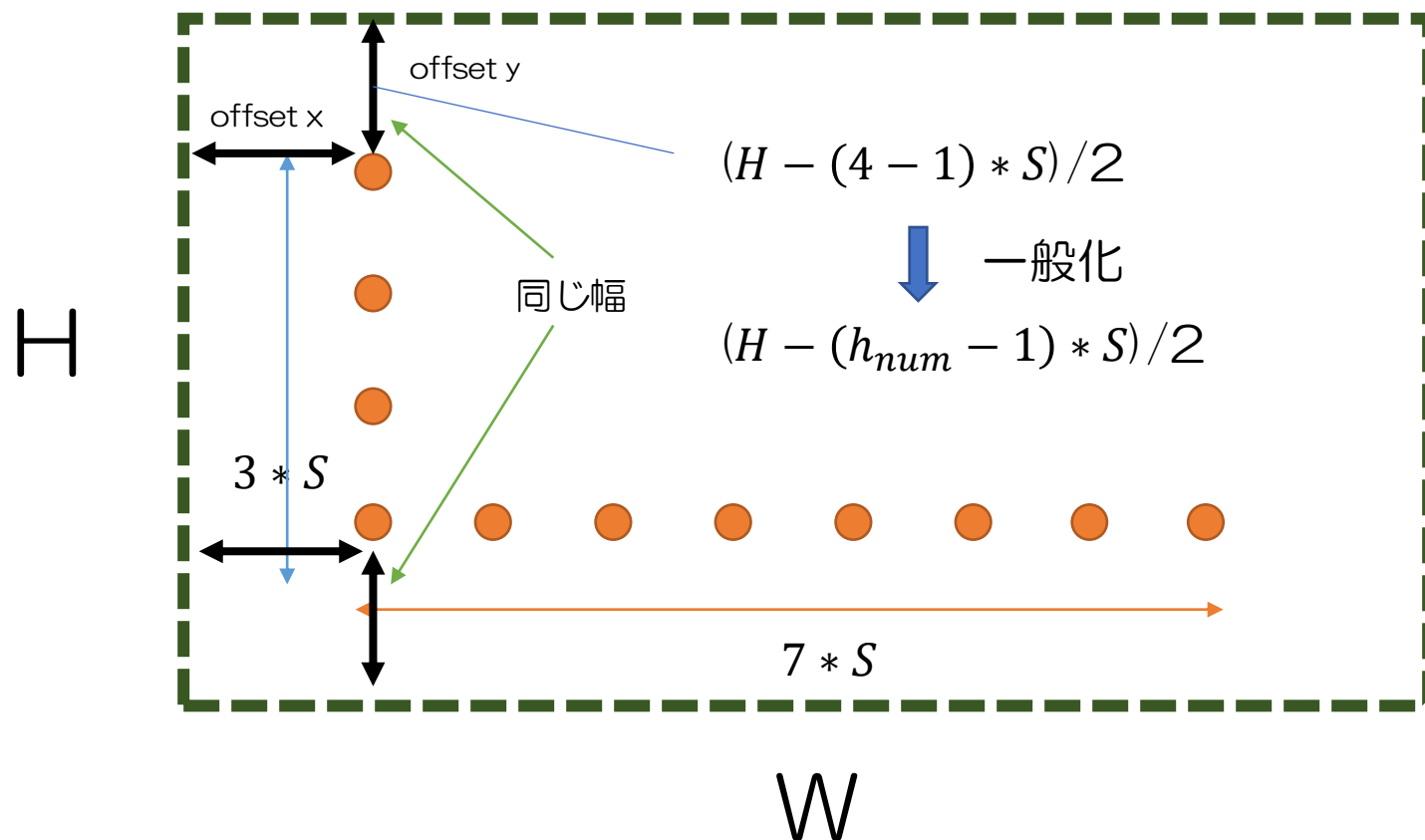
→ グリッドの間隔 $s = \sqrt{\frac{N}{k}} = \sqrt{\frac{HW}{k}}$



等間隔で各クラスタの重心を配置

最後に、等間隔で配置

入力画像



32個のクラスタの場合
4行、8列のグリッド状
 h_{num} 行 w_{num} 列

offset x

$$(W - (W_{num} - 1) * S) / 2$$

offset y

$$(H - (h_{num} - 1) * S) / 2$$

初期化

- 1 等間隔で各クラスタの重心を配置
- 2 勾配に基づいて重心の位置を微調整
特筆することなし

SLIC アルゴリズム

1 RGB画像をLab画像に変換



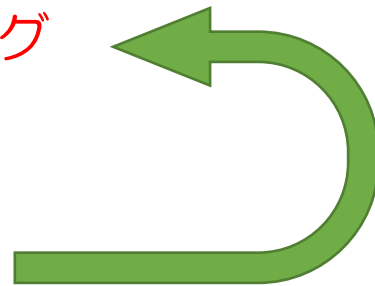
2 初期化



3 局所クラスタリング



4 重心を移動



繰り返す

Algorithm 1 Efficient superpixel segmentation

```
1: Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$  by sampling pixels at regular grid steps  $S$ .
2: Perturb cluster centers in an  $n \times n$  neighborhood, to the lowest gradient position.
3: repeat
4:   for each cluster center  $C_k$  do
5:     Assign the best matching pixels from a  $2S \times 2S$  square neighborhood around the cluster center according to the distance measure (Eq. 1).
6:   end for
7:   Compute new cluster centers and residual error  $E$  {L1 distance between previous centers and recomputed centers}
8: until  $E \leq \text{threshold}$ 
9: Enforce connectivity.
```

局所クラスタリング

クラスタリングのためには以下の3つが必要

ピクセル情報

(H,W,5)

各ピクセルの
l, a, b, x, y値

ラベル情報

(H,W)

各ピクセルが
どのクラスに
属しているか

初期値は
None

距離情報

(H,W)

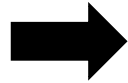
各ピクセルが
属しているクラスの
重心との距離

初期値はinf

局所クラスタリング

重心(x_i, y_i)

$$x_i - S \leq x_i \leq x_i + S$$



画像の幅や高さを超えないようにする必要

$$y_i - S \leq y_i \leq y_i + S$$

実装例

```
x_lower = max(0, center_x - self.S)
x_upper = min(self.w, center_x + self.S)
y_lower = max(0, center_y - self.S)
y_upper = min(self.h, center_y + self.S)

for y in range(y_lower, y_upper):
    for x in range(x_lower, x_upper):
        # 処理
```

これ以外は普通のクラスタリングと変わらないため特筆しません

SLIC アルゴリズム

① RGB画像をLab画像に変換



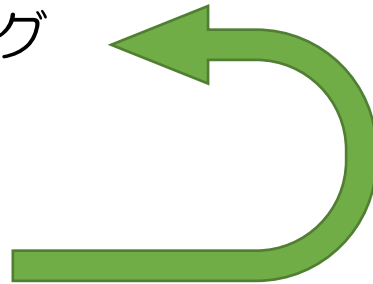
② 初期化



③ 局所クラスタリング



④ 重心を移動



繰り返す

Algorithm 1 Efficient superpixel segmentation

```
1: Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$  by sampling pixels at regular grid steps  $S$ .
2: Perturb cluster centers in an  $n \times n$  neighborhood, to the lowest gradient position.
3: repeat
4:   for each cluster center  $C_k$  do
5:     Assign the best matching pixels from a  $2S \times 2S$  square neighborhood around the cluster center according to the distance measure (Eq. 1).
6:   end for
7:   Compute new cluster centers and residual error  $E$  {L1 distance between previous centers and recomputed centers}
8: until  $E \leq \text{threshold}$ 
9: Enforce connectivity.
```

重心を移動

各クラスタに含まれる全データの[l, a, b, x, y]の平均値を移動先の重心とする

Numpyを駆使して実装してみました

```
for l in range(self.k):
    idxs = np.where(self.pixels_label == l) # 各クラスタに属するピクセル座標[y,x]
    cnt = len(idxs[0]) # 各クラスタに属するピクセルの個数
    avg_y = np.round(np.sum(idxs[0]) / cnt)
    avg_x = np.round(np.sum(idxs[1]) / cnt)
    idxs = np.stack([idxs[0], idxs[1]], 1) # x座標とy座標の情報を結合
    avg_l = np.round(np.sum(self.pixels[tuple((idxs).T)][:,0]) / cnt)
    avg_b = np.round(np.sum(self.pixels[tuple((idxs).T)][:,2]) / cnt)
    avg_a = np.round(np.sum(self.pixels[tuple((idxs).T)][:,1]) / cnt)

    next_center = np.array([avg_l, avg_a, avg_b, avg_x, avg_y], dtype=int)
```

SLIC 結果

最大ステップ数：10
閾値：0.001



入力
256×256



ステップ数：4
クラスタ数：10
閾値以下になり終了



ステップ数：1
クラスタ数：20
閾値以下になり終了

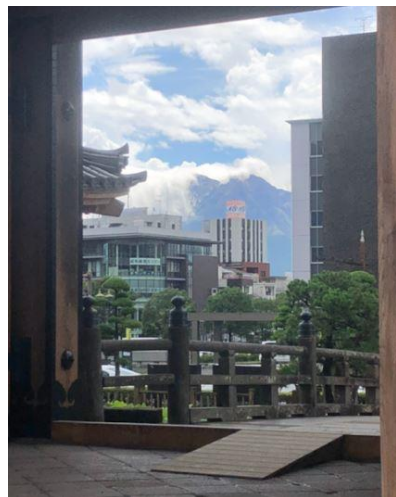


ステップ数：1
クラスタ数：50
閾値以下になり終了

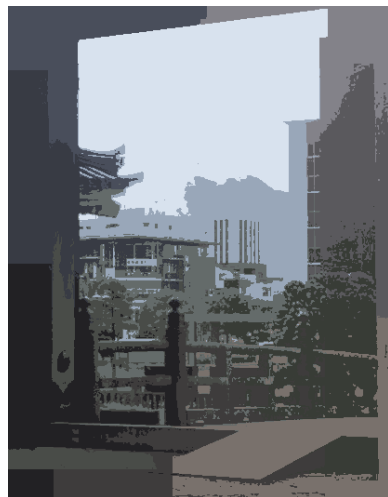
※イテレーション数 = ステップ数の意
クラスタ数は指定数（実際のクラスタ数はこれより数個多い）

SLIC 結果

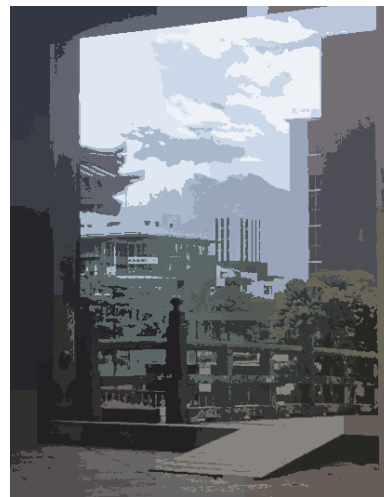
最大ステップ数：10
閾値：0.001



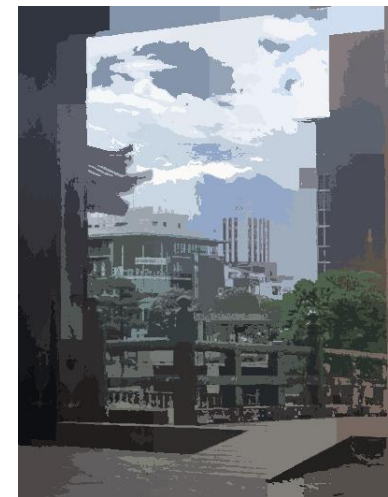
入力
421×553



ステップ数：2
クラスタ数：10
閾値以下になり終了



ステップ数：2
クラスタ数：20
閾値以下になり終了

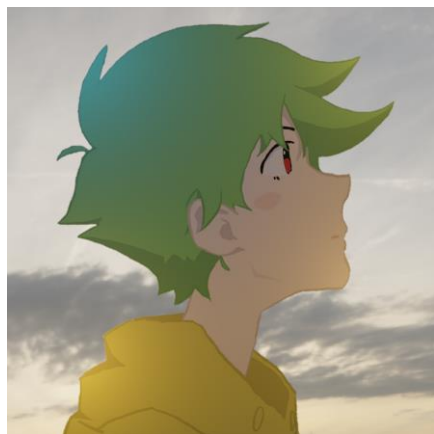


ステップ数：2
クラスタ数：50
閾値以下になり終了

※イテレーション数 = ステップ数の意
クラスタ数は指定数（実際のクラスタ数はこれより数個多い）

SLIC 結果

最大ステップ数：10
閾値：0.001



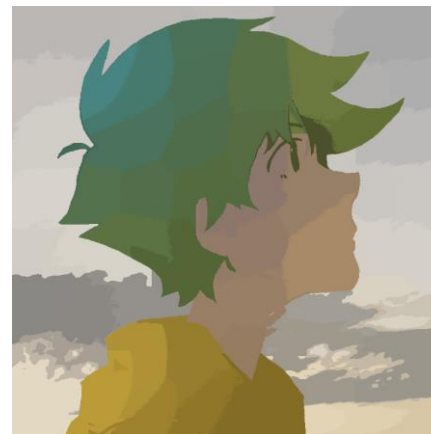
入力
768×768



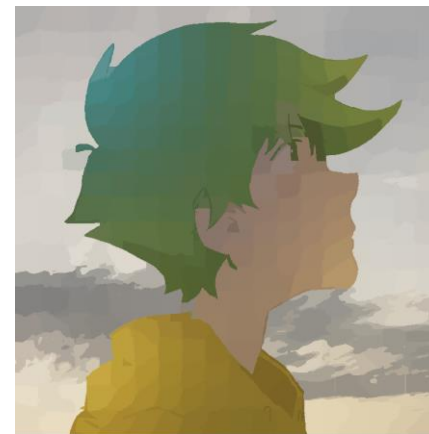
ステップ数：2
クラスタ数：10
閾値以下になり終了



ステップ数：1
クラスタ数：20
閾値以下になり終了



ステップ数：1
クラスタ数：50
閾値以下になり終了



ステップ数：1
クラスタ数：300
閾値以下になり終了

※イテレーション数 = ステップ数の意
クラスタ数は指定数（実際のクラスタ数はこれより数個多い）

ライブラリ

2種類のライブラリでSLICは実装されている

- OpenCV
- skimage

当たり前ですが
ライブラリを使用した方が
速いし簡単です…

OpenCVの結果

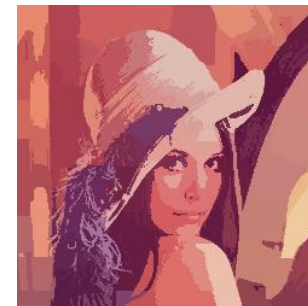
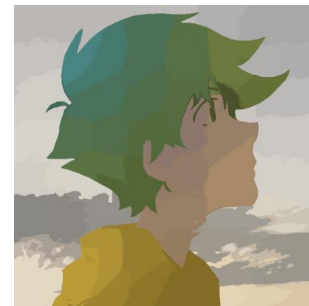


OpenCVの仕様として
クラスタ数ではなくスーパーピクセルのサイズを指定

まとめ

- スーパーピクセルは複数のピクセルを特徴や色を用いてまとめたもの
- セグメンテーション技術の1種
- 代表的なアルゴリズムにSimple Linear Iterative Clustering (SLIC) がある
- SLICはk-meansを取り入れた局所クラスタリングの手法

SLIC結果



RGBとLab

クラスタ数：50

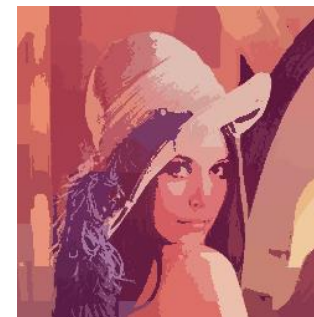


クラスタ数：20



RGB画像で実行

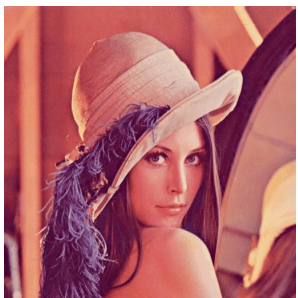
クラスタ数が多いと重心があまり動かないので
似た結果になっている



Lab画像で実行
(論文手法)

参考

RGB



B



G



R

参考

HSVとLab



H



S



V



L



a



b