応用情報工学演習　“Seam Carving”課題

2019/11/14

1. はじめに

Seam Carving とは画像のサイズ変更手法の 1 つであり，2007 年に SIGGRAPH と呼ばれるコンピュータグラフィックスに関する国際会議で発表された非常に有名な手法である．今回は Python を用いて横方向に画像を縮小する Seam Carving を実装してみよう．Seam Carving は，図 1 の赤線ように画像から取り除いても目立たないシーム（画像の上から下まで走る連続した幅 1 画素の経路）を繰り返し取り除くことにより，画像の重要な部分を残しつつサイズを縮小する．

図 1: シームの例

図 1 の赤線ような取り除いても目立たないシームを取り除くことで，画像の横幅を 1 画素分だけ小さくすることができる．この処理を所望の大きさになるまで繰り返す．図 2 は横幅を 256 画素から 150 画素まで縮小した際の結果である．Scaling (画像全体で一定の縮小)では船や看板などが横方向に大きく縮められてしまい，Cropping (切り出し)では画像の両端がなくなってしまっている．それに対して Seam Carving では，船の大きさや「嵐」の文字を保ちつつ縮小できていることがわかる．

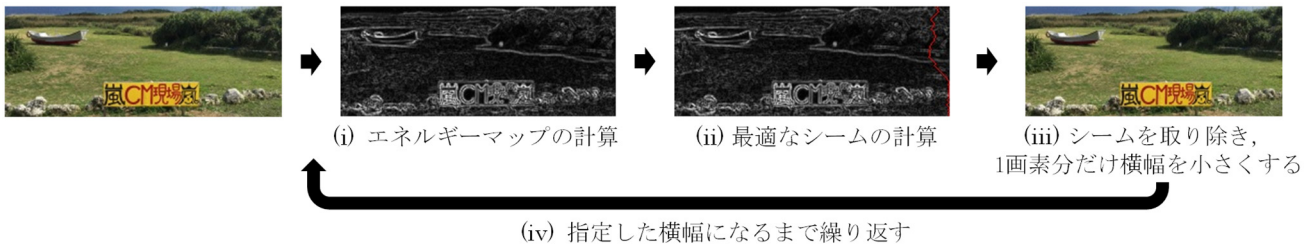
Seam Carving

Scaling (画像全体で一定の縮小)

Cropping (切り出し)

図 2: 縮小方法の比較

Seam Carving の概要は以下のとおり:



(i) 入力画像の各画素で，「取り除くとどれだけ目立つか」を表すエネルギーの値を計算する．

(ii) 経路上のエネルギーの和が最小になるように，画像の上から下への連続した経路（シームと呼ぶ）を見つける．

(iii) 見つけたシームを取り除くことで，画像の横幅を 1 画素縮める．

(iv) (i)~(iv)を所望の横幅になるまで繰り返す．

2. Seam Carving

各ステップの詳細を説明する．

2.1 エネルギーマップの計算

入力画像の各画素で，「取り除くとどれだけ目立つか」を表すエネルギーの値を計算する．ここでは，画素値が空間的にあまり変化しないのっぺりとした領域の画素は取り除いても目立ちにくく，逆に画素値が空間的に大きく変化するエッジ付近の画素を取り除くと目立ちやすいと考え，エネルギーを以下の式で定義する．

$$E(\mathrm{y},\mathrm{x}) = \left|\frac{\partial}{\partial x} I(y,x)\right| + \left|\frac{\partial}{\partial y} I(y,x)\right| + \left|\frac{\partial^2}{\partial \mathrm{x}^2} I(y,x) + \frac{\partial^2}{\partial y^2} I(y,x)\right|, \tag{1}$$

ここで，(y,x)は y 行目 x 列目の画素であり，$\mathrm{E}(\mathrm{y},\mathrm{x})$はその画素のエネルギーの値，$\mathrm{I}(\mathrm{y},\mathrm{x})$は入力画像の画素値である．入力画像の大きさを$\mathrm{N}\times\mathrm{M}$画素とし，$\mathrm{y}=\{0,\dots,\mathrm{N}-1\}$, $\mathrm{x}=\{0,\dots,\mathrm{M}-1\}$とする．図 3 は全画素でエネルギーを計算したエネルギーマップである．明るい画素ほど高いエネルギーを持つ．

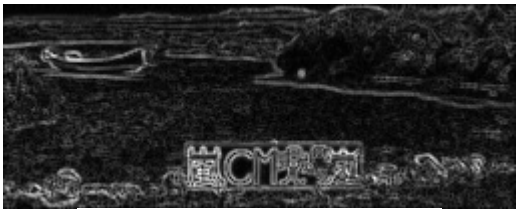


図 3: エネルギーマップ

2.2 動的計画法による最適なシームの計算

経路上のエネルギーの和が最小になるように，画像の上から下への連続した経路（シームと呼ぶ）を見つける．y 行目におけるシームの x 座標を$\mathrm{x}=\mathrm{s}(\mathrm{y})$と定義すると，シームはベクトル表記を用いて$\mathbf{s}=\big(\mathrm{s}(0),\dots,\mathrm{s}(\mathrm{N}-1)\big)^{\mathrm{T}}$と表現できる．エネルギーの和が最小になるような（取り除いても最も目立たない）シームを見つける問題は以下のように定式化される．

$$\mathbf{s} = \underset{s}{\operatorname{argmin}} \sum_{y=0}^{N-1} E(y,s(y)), \qquad \text{s.t. } |s(y)-s(y+1)| \le 1 \tag{2}$$

ここで，制約条件 $|s(y)-s(y+1)| \le 1$ は，シームが連続する（x 座標が 2 以上離れない）ための制約である．不連続を許してしまうと，シームを取り除いた際に画像に不自然な切れ目が発生する恐れがある．図 4 は図 3 のエネルギーマップにおける最適なシームである．

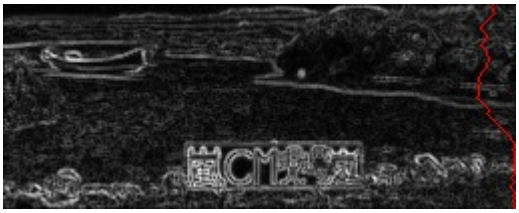


図 4: 最適なシーム

式(2)の最適化問題は，動的計画法を用いることで O(MN)の計算量で大域的最適解を求めることができる．図 5 の左は簡単なエネルギーマップの例である．図 5 の右の赤の経路がエネルギーの和が最小の連続する経路であり，その和は$0+1+2+0=3$となっている．

6	4	2	0	5	1	7
3	5	1	9	8	0	7
2	7	5	2	8	7	8
3	3	5	4	0	6	7

6	4	2	**0**	5	1	7
3	5	**1**	9	8	0	7
2	7	5	**2**	8	7	8
3	3	5	4	**0**	6	7

図 5: 簡単なエネルギーマップの例とその最適なシーム

図 5（右）で得られたシームの座標(行，列)は(0, 3), (1, 2), (2, 3), (3, 4)なので，s(0)=3, s(1)=2, s(2)=3, s(3)=4 となり$\mathbf{s}=(3,2,3,4)^{\mathrm{T}}$と表現される．以下で最適なシームを見つける手順を解説する．

2.2.1 エネルギーの累積計算

まず，エネルギーを上から下に累積計算する．

下図の緑の画素に注目すると，そこへの到達可能な経路は真上か右上の 2 通りが考えられる．そのうちエネルギーの低いほう（ここでは 4）を選んで累積する（3+4=7）．この際に，選んだ経路を保存しておく．

6	4	2	0	5	1	7
3	5	1	9	8	0	7
2	7	5	2	8	7	8
3	3	5	4	0	6	7

次に，注目画素を一つ隣に移動する．ここでは，左上，真上，右上の 3 通りの経路が考えられ，そのうち最もエネルギーの低いほう（ここでは 2）を選んで累積する（5+2=7）．選んだ経路は保存しておく．

6	4	2	0	5	1	7
7	5	1	9	8	0	7
2	7	5	2	8	7	8
3	3	5	4	0	6	7

上記の累積計算を一番下の行まで繰り返す.

6	4	2	0	5	1	7
7	7	1	9	8	1	8
9	8	6	3	9	8	9
11	9	8	7	3	14	7

A(y, x)を累積エネルギーマップとすると，この計算は次の式を使って表せる.

$$A(y,x) = E(y,x) + \min\{A(y-1,x-1), A(y-1,x), A(y-1,x+1)\} \quad (3)$$

2.2.2 最小値を見つける

次に，累積計算したエネルギーマップA(y, x)の一番下の行において最小値を見つける.

$$x^* = \underset{x}{\operatorname{argmin}}\, A(N-1,x)$$

6	4	2	0	5	1	7
7	7	1	9	8	1	8
9	8	6	3	9	8	9
11	9	8	7	3	14	15

この最小値の値$A(N-1,x^*)$ が，考えられる全経路のうち最小の累積エネルギーとなっている（ここでは 3）.

2.2.3 バックトラック

最後に，上記で見つけた最小の座標$(N-1,x^*)$ から，あとは保存した経路を戻ることで最適なシームを求めることができる．この保存した経路を戻る処理をバックトラックと呼ぶ.

6	4	2	0	5	1	7
7	7	1	9	8	1	8
9	8	6	3	9	8	9
11	9	8	7	3	14	15

2.3 シームの除去

2	21	90	69	53	24	96
25	70	16	31	66	10	51
90	28	16	33	31	56	90
36	86	86	13	43	91	83

➡

2	21	90	53	24	96
25	70	31	66	10	51
90	28	16	31	56	90
36	86	86	13	91	83

見つけたシームの位置にある画素を入力画像から取り除くことで，1 画素分だけ横幅の小さな画像に

することができる.

2.4 所望の横幅になるまで手順 2.1~2.3 を繰り返す

3. 課題

以下の手順で Seam Carving のソースコードを完成させなさい.

(STEP 1) エネルギーマップを計算する `computeEnergy()`メソッドにコードを追加して完成させなさい. 入力画像 Iから式（1）に則りエネルギーマップ Eを計算するように実装しなさい. 1 次微分には OpenCV に実装されている cv2.Sobel()フィルタ, 2 次微分には cv2.Laplacian フィルタを使うとよい.

（STEP 1 のヒント）入力画像は 3 チャンネルなので，フィルタリング後の結果も 3 チャンネルである. 3 チャンネルの結果を足し合わせて 1 チャンネルのエネルギーマップにする必要がある. 入力画像のサイズは(102, 256, 3)の 3 次元配列なので，出力はサイズ(102, 256)の 2 次元配列にする.

(STEP 2) 最適な（エネルギーの和が最小になる）シームを見つける `findSeam()`メソッドにコードを追加して完成させなさい. 上記の 2.2 で説明された動的計画法の処理を実装し，return でシーム $\mathbf{s}$ を返す.

※シームは zeros で初期化されているので（つまり$\mathbf{s} = (0, \ldots, 0)^{\mathrm{T}}$），何もコードを追加していない状態だと，入力画像の左端が繰り返し削れていく.

（STEP 2 のヒント）両端の処理は，if 文を使った例外処理をするか，もしくは両端の外側を∞のエネルギーに設定する padding 処理の 2 通りが考えられる. 自分の実装しやすいほうでよい.

※シームを除去するメソッド removeSeam()や繰り返し処理は実装済みである.

(STEP 3) コマンドラインで"python TestStep2.py"を実行し，パスすることを確認しなさい. TestStep2.py は STEP 2 の findSeam()メソッドが正しく動作するかどうかを確認するテストコードである.

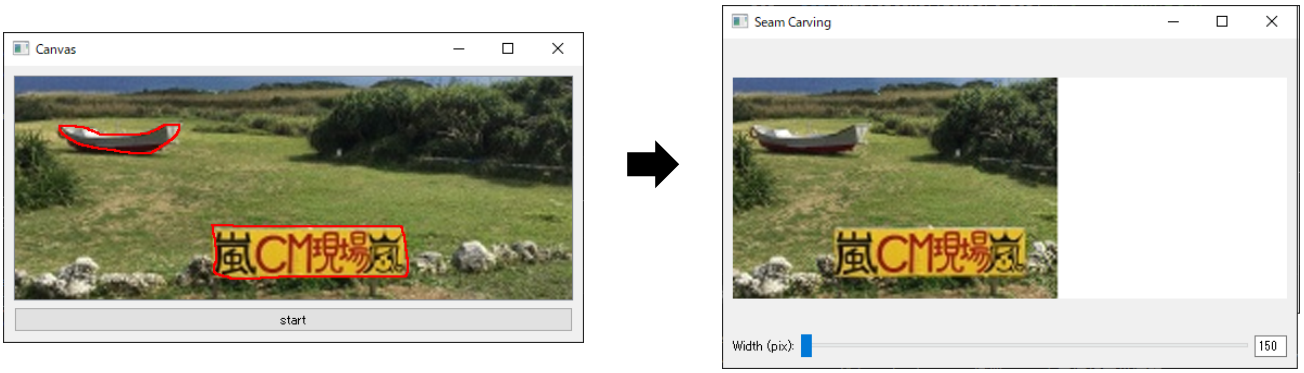
(STEP 4) コマンドラインで"python SeamCarving.py"を実行し，結果の見た目が図 1 の左の画像（ground_truth.jpg）とおおよそ一致することを確認しなさい.

※エネルギー計算の細かな違いなどによって若干の結果の違いが生まれるため，完全に一致しなくてもよい.

（オプション課題）絶対に取り除かれたくない領域をマウスで指定できる機能を PyQt の GUI で実装しなさい．エネルギーマップの値を変更することで，その部分が絶対に取り除かれないようにしなさい．

※実装済みの部分を好きに書き換えてよい．

※オプション課題に取り組む場合は SeamCarvng.py とは別ファイルで提出すること．



4. 課題提出方法

- LETUS の課題提出 BOX に 11/28(木) 16:00 までに提出すること．
- 提出物：
  - ソースコード（処理概要が分かるようにコメントを入れること）

以上

参考文献：

S. Avidan and A. Shamir, “Seam Carving for Content-Aware Image Resizing,” Proc. SIGGRAPH, 2007.
https://perso.crans.org/frenoy/matlab2012/seamcarving.pdf