Flickr は海岸線を描けるか？

大森　雅己† 廣田　雅春†† 石川　博††† 横山　昌平††††

† 静岡大学大学院情報学研究科　〒 432–8011 静岡県浜松市中区城北 3–5–1
†† 静岡大学創造科学技術大学院/日本学術振興会特別研究員 DC　〒 432–8011 静岡県浜松市中区城北 3–5–1
††† 首都大学東京システムデザイン学部情報通信システムコース　〒 191–0065 東京都日野市旭が丘 6–6
†††† 静岡大学大学院情報学研究科　〒 432–8011 静岡県浜松市中区城北 3–5–1
E-mail: †gs13008@s.inf.shizuoka.ac.jp, ††dgs11538@s.inf.shizuoka.ac.jp , †††ishikawa-hiroshi@sd.tmu.ac.jp,
††††yokoyama@inf.shizuoka.ac.jp

あらまし　Flickr などで共有されている写真には，写真の撮影位置を示すジオタグや，写真が何であるかを表すタグが付与されているものが数多くある．我々は，任意のタグ X を持つ写真群の，世界地図空間における撮影位置の偏在が，X の地理的な特徴を示しているのではないかという仮説を立てた．この仮説を検証するために,「beach」というタグから「海岸線」を描画する事に取り組んだ．我々の調査で,「beach」とタグづけられた写真の約 80%が実際の海岸線から 500m 以内で撮影されている事が分かった．そこで，その膨大な撮影位置の点群から，海岸線を描く手法を提案し評価する．提案手法は特定の地物のみに対応するものではないが，明確な正解を有する海岸線に着目して定量的な評価をする事でソーシャルタギング全体の信憑性の推定ができると考えている．

キーワード　ジオタグ，ソーシャルタギング，GIS

1. はじめに

近年，GPS が搭載されたカメラやスマートフォンの普及により，ジオタグが付与された写真が増加している．本研究において，ジオタグとは，写真の撮影地点を表す緯度・経度である．それらの写真は，Flickr [1] や Panoramio [2] などの写真共有サイトで共有され，ソーシャルタギングによってタグ付けされている．本研究において，タグとは，この Flickr 上のソーシャルタギングによって付与されたタグである．タグの例として,「beach」や「japan」などの場所を表すタグ,「2013」や「winter」などの時間を表すタグ,「iphone」や「canon」などのその写真の撮影に用いられた機材やそのメーカーを表すタグ,「bird」や「tree」などの撮影対象を表すタグなどがあげられる．

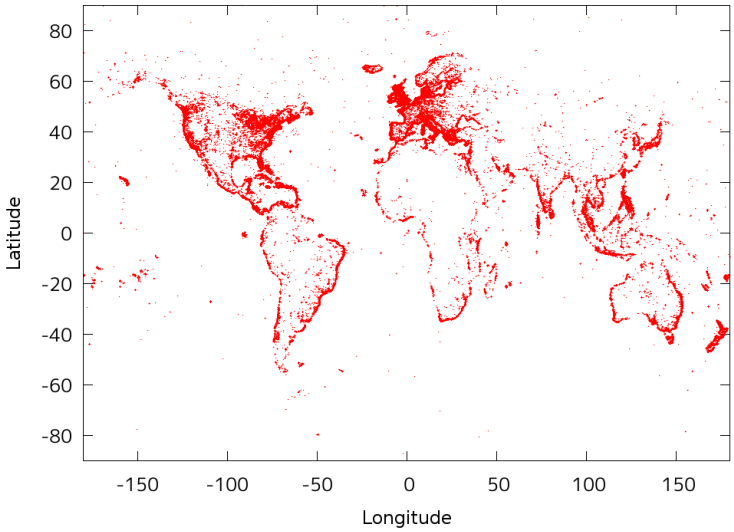
我々は，写真の中身を見ずにジオタグとタグだけから海岸の形などの地理的な形状を再現可能ではないかと考えた．そのため，ある 1 つの撮影対象を表すタグが付与された写真を地図上に配置した時，写真は，地図上のそのタグに関連のある場所に多く存在すると考えられる．例えば，タグ「coast」が付与された写真の多くは，海岸付近で撮影されていると考えられる．そこで，我々は，写真の撮影位置の集合から，タグが表す地理的な形状や広がりを把握することができるのではないかという仮説を立てた．そこで，本論文では，この仮説を検証するために，写真に付与されたジオタグとタグから海岸線を抽出することを目指す．また，抽出した海岸線と実際の海岸線の距離を算出し，その誤差の評価を行う．海岸線を選んだ理由は，地図から実際の海岸線の形状を容易に把握できることや，OpenStreetMap [7] や NOAA [8] から実際の海岸線データを取得可能なことから，提案手法の性能を評価することが可能であることがあげられる．本論文では，海岸線に着目して線を引くが，本手法は，海岸線に限らずに与えられたタグが表している地理的な領域を求めるための手法である．

写真の撮影位置から海岸線を再現するためには，まず，海岸線付近で撮影された写真に付与されると考えられるタグが，実際に海岸線付近で撮影された写真に付与されているのかを検証する必要がある．そこで，2 章では，Flickr 上の海岸線に関するタグが付与された写真を用いた地理的なソーシャルタギングの信憑性の評価について述べる．3 章では，写真数が多く，海岸線付近で撮影されている割合の高い，タグ「beach」を用いて，写真の撮影位置から海岸線を描く手法の概要について述べる．4 章では，提案手法を実装し，ハワイとイギリス付近で撮影された写真を用いて実行した結果を示し，それに対する考察を述べる．5 章では，4 章での結果について実際の海岸線との距離を求め，評価を行う．6 章では，本研究とそれに対する関連研究について述べる．7 章では，本研究で得られた成果のまとめと今後の展望について述べる．

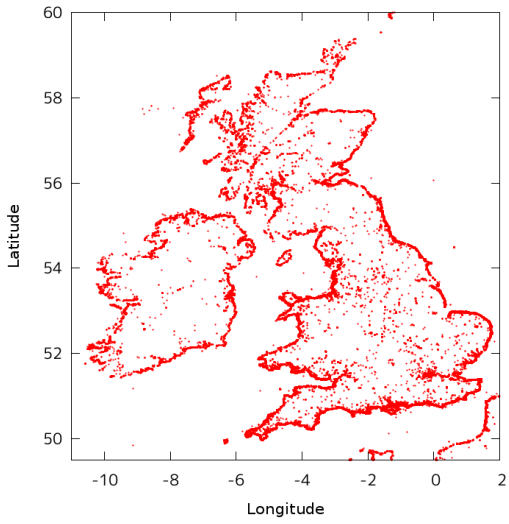
2. 地理的なソーシャルタギングの信憑性

最初に，我々は，海岸線付近で撮影された写真に付与されると考えられるタグが実際に海岸線付近で撮影された写真に付与されているかを調査し，地理的なソーシャルタギングの信憑性の定量的な評価を行う．本研究において，地理的なソーシャルタギングの信憑性は，あるタグが付与された写真の撮影位置と，そのタグが表している場所との距離と定義し，その距離が近いほど信憑性が高いとする．例えば,「coast」というタグについては，そのタグが付与された写真の撮影地点と海岸線の距離が近い写真が多いほど信憑性が高いと評価する．

我々は，地理的なソーシャルタギングの信憑性を調査するため，Flickr から海岸線を表すと考えられるタグが付与された写



(a) 世界全体



(b) イギリス付近

図 1　タグ「beach」が付与された写真の撮影位置

表 1　海岸や海を表すタグと，そのタグが付与された写真数

Tag	Number of phogoraphs
beach	2,488,923
sea	1,689,924
coastline	60,245
shoreline	47,114

表 2　実際の海岸線からそれぞれのタグが付与された写真の撮影位置までの距離の平均誤差

Tag	Average error [m]
beach	7,293.93
sea	8,307.95
shoreline	23,510.43
coastline	3,921.20

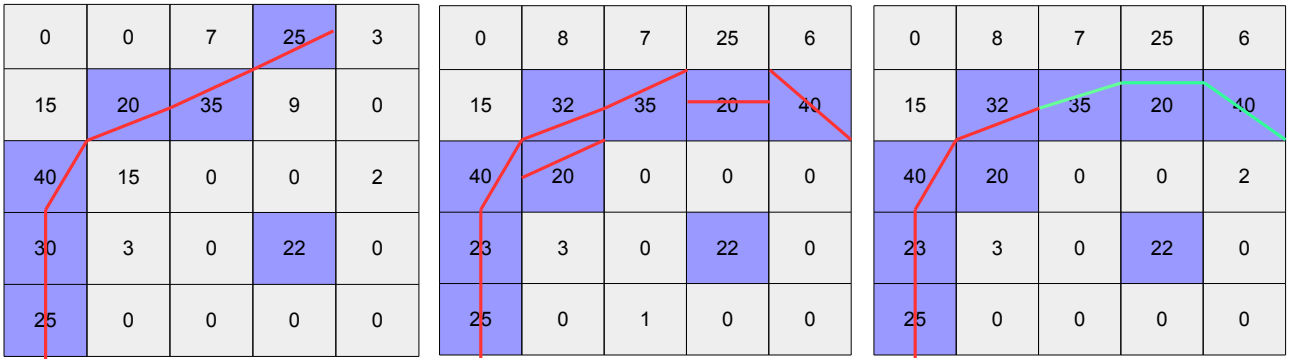
表 3　それぞれのタグが付与された写真の撮影位置の分布

Tag	$\leqq 100m$[%]	$\leqq 500m$[%]
beach	51.34	80.44
sea	48.20	76.77
coastline	56.25	82.54
shoreline	51.92	70.93

真を収集した．表 1 は，写真の収集に用いたタグとそのタグが付与された写真の数を表している．これらの写真の撮影位置と正解データとの距離を求めた．正解データとは，実際の海岸線のデータであり，今回は，OpenStreetMap の海岸線データを利用した．

写真の撮影位置と実際の海岸線との距離を調査した結果を表 2，表 3 に示す．表 2 は，それぞれのタグが付与された写真の撮影位置と正解の海岸線との距離の平均誤差を示している．表 3 は，それぞれのタグの付与された写真ごとに，海岸線から 100m 以内で撮影された写真の割合，500m 以内で撮影された写真の割合を示している．表 2 から，写真の撮影位置と実際の海岸線の位置との平均誤差は，どのタグも数千 m 以上であったが，表 3 から，多くの写真が海岸線から 500m 以内で撮影されていることがわかる．平均誤差が大きい原因として，ノイズとなる写真の存在が考えられる．ノイズとなる写真には，誤ったジオタグやタグが付与された写真や，閲覧数を上げるために，故意に写真の内容と関係がないタグが付与された写真などがある．これらの写真の撮影位置は，海岸線とは関係なく様々な場所に存在する．そのため，海岸線から遠い場所ににも写真が多く存在し，写真の撮影位置と実際の海岸線の位置との平均誤差が大きくなる．また，表 2 より，タグ「sea」の平均誤差は，タグ「beach」の平均誤差よりも大きい．その理由として，タグ「sea」は，道路や山の中腹，海上など海岸から離れた場所であっても，撮影された写真に海が写っていた場合に付与されると考えられる．また，海岸線付近で撮影された写真においても，500m 以内に写真が多く，海岸線から多少離れた場所で撮影された写真が多い．その理由としては，写真の撮影位置と撮影対象との距離，潮の満ち引き，GPS の誤差などが考えられる．その中でも撮影対象との距離が主な理由として考えられる．海岸の写真を撮影する時，多くの場合は，浜辺など陸から海岸を撮影し，海岸線上で撮影することは少ないと考えられる．そのため，海岸線の位置と撮影位置に数百 m 程度の差があると考えられる．

表 1, 表 2, および表 3 より，タグ「beach」の写真が写真数が最も多く，海岸線付近で撮影されている割合が高いタグであることが分かる．表 1 のタグ「beach」の写真の撮影位置をプロットしたものを図 1 に示す．図 1(a) は，すべての写真をプロットした結果である．図 1(b) は，撮影位置がイギリス付近の写真のみをプロットした結果である．図 1(a) から，それぞれの大陸のおおよその形が把握できる．また，写真の枚数が多いヨーロッパ大陸やイギリス付近では，図 1(a) から海岸線の形状を把握することは困難であるが，拡大すると，図 1(b) のように，海岸線の形状が把握できる．そこで，データ数が十分にあり，海岸線付近で撮影された写真の割合が高いタグ「beach」



(a) 線がつながっている例　(b) 線がつながっていない例　(c) 線をつないだ状態

図 2　線 の 描 画

が付与された写真を用いて海岸線を再現することを試みる．

3. 提 案 手 法

本論文では，タグ「beach」を持つ写真の撮影位置から海岸線を描く手法を提案する．提案手法は，海岸線を再現する地域をグリッドに分割し，それぞれのセル内で線を引くフェーズ，周囲のセル間で，つながっていない線をつなぎ修正するフェーズの 2 つのフェーズからなる．

3.1 線 の 描 画

本節では，写真の撮影位置に基づいて線を引く．線を引く流れは，グリッドに分割，線を引く候補となるセルの決定，候補となったセル内に線を描画である．線を引く候補となるセルの決定と候補となったセル内に線を描画する手順を擬似コードで表したものを Algorithm 1 に示す．ここで，引数 θ は候補セルを決めるための閾値，num($cell$) は，$cell$ に含まれている写真の枚数を返す関数，around4($cell$) は，$cell$ に隣接する 4 セルを返す関数，around8($cell$) は，$cell$ の周囲の 8 セルを返す関数，max($cells$) は，$cells$ の中で，セル内の写真の枚数が最も多いセルを返す関数，opposite($cellA$,$cellB$) は，$cellA$ に対して $cellB$ と反対の方向にあるセルを返す関数，center($cellA$,$cellB$) は，$cellA$ の中心と $cellB$ の中心との中間点を返す関数である．また，本手法を用いて線を引く例を図 2(a) に示す．セル内の数値がそれぞれのセル内で撮影された写真の枚数，青色のセルが線を引く候補となるセル，赤色の線が引いた線である．

まず，海岸線を再現する地域を選択し，その地域を 1 つのセルの 1 辺が約 Nm となるグリッドに分割する．そして，それぞれのセルに対して，作成されたセルの範囲内で撮影された写真の枚数を算出する．それぞれのセルに対して，隣接する上下左右のセルと写真の枚数を比較し，少なくとも 1 つのセルとの枚数の差が閾値 θ 以上の場合，枚数が多い方のセルを線を引く候補とする．図 2(a) の候補となるセルは，$\theta = 20$ とした場合の例である．

次に，候補となったそれぞれのセルに対して，周囲の 8 つのセルの中に 1 つでも候補となるセルが存在した場合，8 つのセルのうち，最も枚数の多いセルから，その 8 つのセルに含まれる最も枚数の多いセルから隣接しないセルの中で，最も枚数の多いセルへ向かって線を引く．その際，8 つのセルのうち，最も枚数の多いセルに隣接しないすべてのセルに写真が存在しない場合は，中心のセルに対し，最も枚数の多いセルから点対称となる位置のセルへ線を引く．また，周囲の 8 セルすべてのセルが候補となるセルでない場合は，線を引かない．

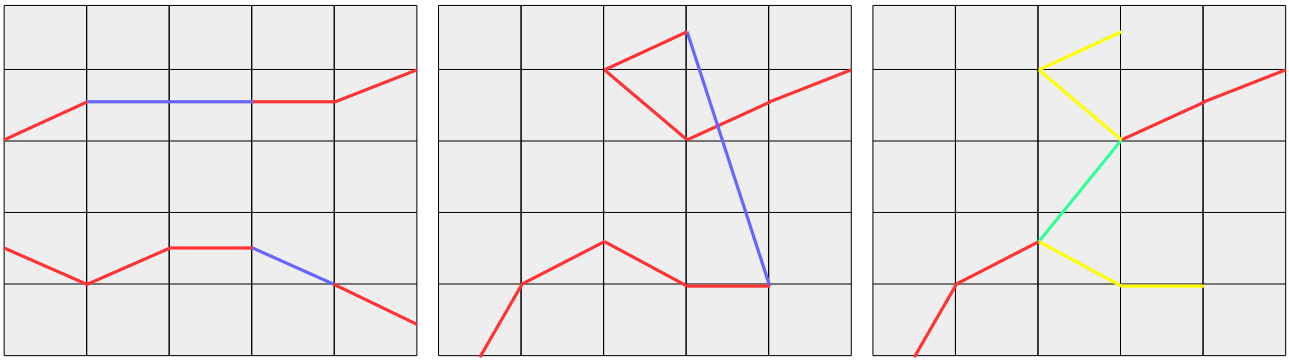
Algorithm 1 線の描画

```
1: arguments: θ
2: candidate_cells ⇐ φ
3: foreach cellα ∈ すべてのセル
4:    foreach cellβ ∈ around4(cellα)
5:       if num(cellα) − num(cellβ) > θ then
6:          candidate_cells ⇐ candidate_cells ∪ cellα
7:       end if
8:    end foreach
9: end foreach
10:
11: foreach cellC ∈ candidate_cells
12:    count ⇐ 0
13:    foreach cellA ∈ around8(cellC)
14:       if is_candidate_cell(cellA) then
15:          count ⇐ count + 1
16:       end if
17:    end foreach
18:    if count > 0 then
19:       cellα ⇐ max(around8(cellC))
20:       cellβ ⇐ max(around8(cellC)−cellα−around4(cellα))
      /* cellC の周囲 8 セルから cellA と隣接するセルと，cellA を除外した 5 セルの内，最も写真枚数が多いセル */
21:       if num(cellβ) = 0 then
22:          cellB ⇐ opposite(cellC,cellα)
23:       end if
24:       drawline(center(cellC,cellα), center(cellC,cellβ))
25:    end if
26: end foreach
```

3.2 隣接するセル間の線の接続

本節では，隣接するセル間の線をつなげる手法について述べる．図 2(b) に示すように，前節で述べた方法では，セル内の線と隣のセルの線がつながっていない場合がある．そのため，図 2(c) のように線をつなげることで修正を行う．

セ ル 間 の 線 を つ な げ る ア ル ゴ リ ズ ム を 擬 似 コ ー



(a) 単純につなぐことが可能な場合　(b) 離れたセルをつないだ悪い例　(c) 離れたセルを修正し，つないだ例

図 3　離れた線をつなぐ例

ドで表したものを Algorithm 2 に示す．ここで，is_exist_point_in_cell(*point*,*cell*) は，*cell* 上に *point* が存在するかどうかを返す関数，get_side_between_cells(*cellA*,*cellB*) は，隣接する *cellA* と *cellB* の間にある辺を返す関数，get_points(*side*) は，*side* 上に存在するすべての点を返す関数，center(*points*) は，*points* の中心点を返す関数である．

Algorithm 2 隣接するセルの線の接続

```
1: foreach cellα ∈ 線が引かれたすべてのセル
2:     foreach pointα ∈ cellα 上の線の両端の点
3:         cells ⇐ φ
4:         foreach cellA ∈ around(cellα)
5:             if is_exist_point_in_cell(pointα,cellA) then
6:                 cells ⇐ cells ∪ cellA
7:             end if
8:         end foreach
9:         cellβ ⇐ max(cells)
10:        side ⇐ get_side_between_cells(cellα,cellβ)
11:        points ⇐ get_points(side)
12:        pointα ⇐ center(points)
13:        redrawline(cellα) /* pointα の変更を反映し，再描画 */
14:    end foreach
15: end foreach
16:
17: foreach lineα ∈ 両端がつながっていないすべての線
18:     delete(lineα)
19: end foreach
```

最初に，すべてのセルから，線が引かれたセルを取得する．取得したセルにおいて，あるセルとそのセルに隣接するセルの線が繋がっていない場合は，2 つのセルが共有するセルの辺上に存在するそれぞれの線端の位置をそれらの中間点に変更することで線をつなぐ (図 2(b), 2(c))．ただし，線端がセルの頂点上にある場合は，その点が存在する 2 つの辺のうち，辺を共有するセルの写真の枚数を比較し，写真の枚数が多い方のセルと共有する辺上に線端があるとする．その後，両方の線端が隣のセルとつながっていない線を削除する．

3.3　離れたセル間の線の接続

本節では，離れたセル間の線をつなげる手法について述べる．離れたセルをつなげる例を図 3 に示す．基本的には，図 3(a) のようにつながっていない線端を近い順につなげる．図 3 の赤色の線が前節までの手法で線を引き，つないだ状態であり，青色の線は，赤色の線の終端を近い順に結んだ線である．しかし，図 3(b) に示す例のように，単純に近い線をつなげるだけでは正しい線を引けない場合がある．そこで，図 3(c) のように修正を行う．図の黄色の線が図 3(b) の赤色の線から削除した線であり，緑色の線が最終的につないだ線である．

Algorithm 3 離れたセルの線の接続 (1)

```
1: arguments: maxdist
2: arguments: maxdel
3: arguments: disttable
4: sort_order_by_distance_ascending(disttable)
5: foreach row ∈ disttable
6:     p1 ⇐ get_first_point(row)
7:     p2 ⇐ get_second_point(row)
8:     if not_connect(p1) and not_connect(p2) and get_distance(row)
       < maxdist then
9:         for 1 to maxdel do
10:            if try_connect(p1,p2) then
11:                break
12:            end if
13:            tmp1 ⇐ p1
14:            p1 ⇐ get_connect_point(get_another_point(p1))
15:            if try_connect(p1,p2) then
16:                break
17:            end if
18:            tmp2 ⇐ p1
19:            p1 ⇐ tmp1
20:            p2 ⇐ get_connect_point(get_another_point(p2))
21:            if try_connect(p1,p2) then
22:                break
23:            end if
24:            p1 ⇐ tmp2
25:        end for
26:    end if
27: end foreach
```

セル間の線をつなげるアルゴリズムを擬似コードで表したものを Algorithm 3, Algorithm 4 に示す.

Algorithm 4 離れたセルの線の接続 (2)

```
1: declare function delete_connect_line (point)
2:   while is_exist(point) do
3:     tmp ⇐ get_connect_point(get_another_point(point)
4:     delete (get_line(point))
5:     point ⇐ tmp
6:   end while
7: end function
8:
9: declare function try_connect (p1,p2)
10:  if aim_to(p1,p2) and aim_to(p2,p1) then
11:    delete_connect_line(get_connect_point(p1))
12:    delete_connect_line(get_connect_point(p2))
13:    redrawline(p1,p2)
14:    return true
15:  end if
16:  return false
17: end function
18:
19: declare function aim_to (p1,p2)
20:  result ⇐ true
21:  if left_side(p1) and lon(p1) < lon(p2) then
22:    result ⇐ false
23:  end if
24:  if right_side(p1) and lon(p1) > lon(p2) then
25:    result ⇐ false
26:  end if
27:  if top_side(p1) and lat(p1) > lat(p2) then
28:    result ⇐ false
29:  end if
30:  if bottom_side(p1) and lat(p1) < lat(p2) then
31:    result ⇐ false
32:  end if
33:  return result
34: end function
```

ここで，引数 *maxdist* は，つなぐことを許可する最大距離，引数 *maxdel* は，削除を許可する最大の線の数，引数 *disttable* は，つながっていないすべての線端とそれらの距離を保持するテーブル，get_first_point(*row*) は，任意の一方の線端を *row* から取得する関数，get_second_point(*row*) は，get_first_point 関数とは異なるもう一方の線端を *row* から取得する関数，get_distance(*row*) は，*row* から線端間の距離を取得する関数，not_connect(*point*) は，*point* が異なるセルの線とつながっていないかを確認する関数，get_connect_point(*point*) は，*point* とつながっている線端の点を返す関数，get_another_point(*point*) は，*point* と同じセルに存在するもう一方の線端の点を返す関数，left_side(*point*) は，*point* がセルの左辺上に存在するかを確認する関数，right_side(*point*) は，*point* がセルの右辺上に存在するかを確認する関数，top_side(*point*) は，*point* がセルの上辺上に存在するかを確認する関数，bottom_side(*point*) は，*point* がセルの下辺上に存在するかを確認する関数，get_line(*point*) は，*point* が存在するセル上の線を返す関数，lon(*point*) は，*point* の経度を返す関数，lat(*point*) は，*point* の緯度を返す関数である．

最初に，つながっていないすべての線端から他のつながっていないすべての線端までの距離を求める．そして，距離がパラメータ *maxdist* 未満のペアに対して，近い順に修正し，つなぐ処理を行う．ただし，ペアの一方が，すでに別の点とつないでいた場合はつなぐ処理を行わない．修正の流れは，まず，セル上の線端の位置に基づいて，線端が向いている方向を求める．ここで，双方の線端の向きが向い合っていれば，2 つの点をつなぐ．向い合っていない場合は，それぞれの線を 1 セル分ずつ削除し向かい合った時点で 2 点をつなぐ．ただし，一方の線を削除した数がパラメータ *maxdel* を超えた時点で向かい合っていない場合はつながない．

4. 実 行 結 果

本章では，提案手法を実装し，Flickr 上の写真を用いて海岸線を描いた結果を示す．海岸線の描画結果は，OpenLayers [3] を用いて地図上に表示した．

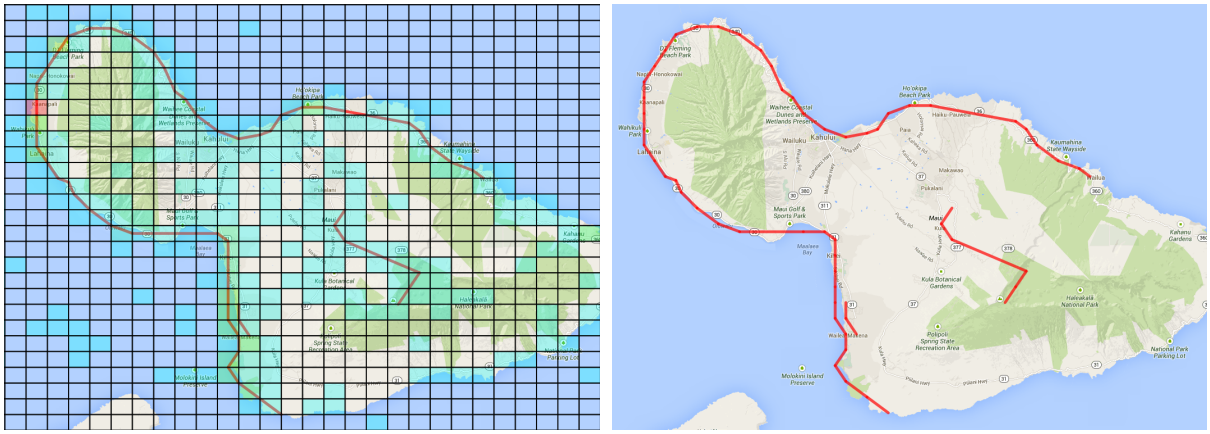
4.1 データセット

本論文では，2 つのデータセットに対して，提案手法を適用した．1 つは，ハワイのマウイ島周辺で撮影された写真である．これは，緯度が 20.4〜21.3，経度が–157.3〜–156 の範囲である．もう 1 つは，イギリス南部で撮影された写真である．これは，緯度が 50〜53，経度が–6〜2 の範囲である．それぞれのデータセットは，指定の範囲内で撮影された，タグ「beach」を持つ写真であり，ハワイの写真数は 12,747 枚，イギリスの写真数は 218,566 枚である．ただし，これらのデータセットにおいて，撮影位置の緯度経度のいずれかが整数値のものは除外した．それらは，GPS によって付与されたものでなく，ユーザにより手入力されたものである可能性が高い．人手による入力は，実際に撮影された位置との誤差が大きいことが多いためである．

4.2 結果・考察

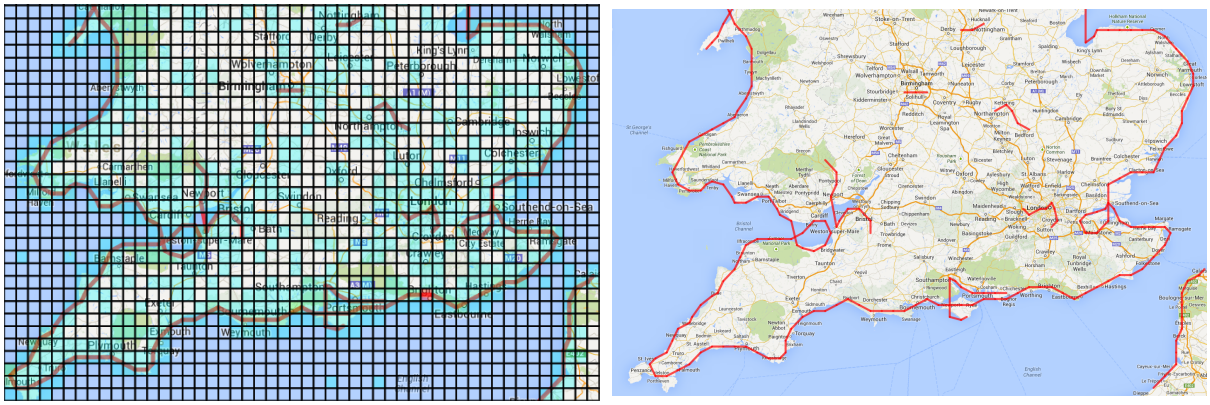
図 4 は，ハワイのマウイ島に本手法を用いて海岸線を引いた結果である．パラメータは，θ を 15，*maxdist* を 5，*maxdel* を 5 とした．図 4(a) は，分割したグリッドの位置，図 4(b) は，本手法を用いて描いた海岸線を表しており，赤色の線が描いた海岸線である．島の西側は，概ね海岸線に沿って線が引かれているが，東側では，ほとんど線が線が引かれていない．これは，写真の枚数が少なく，線を引く候補となる条件の閾値を超えていないことが原因である．また，海岸線から離れた内陸部にも線が引かれている．現在の手法では，線の両端が隣接するセルの線とつながっていない場合にのみ線の削除を行う．内陸部に残っている線は，2 つ以上のセルの線がつながっているため，削除されていない．そのため，線を削除する条件を，線がつながっているセル数が N 未満の場合に削除する変更することや，隣り合うセルに並行な線が引かれていた場合に線を 1 つにまとめて位置を調整するといったことを検討する必要があると考えられる．

図 5 は，イギリスの南部に本手法を用いて海岸線を引いた結果である．パラメータは，前述したハワイの結果と同じものを

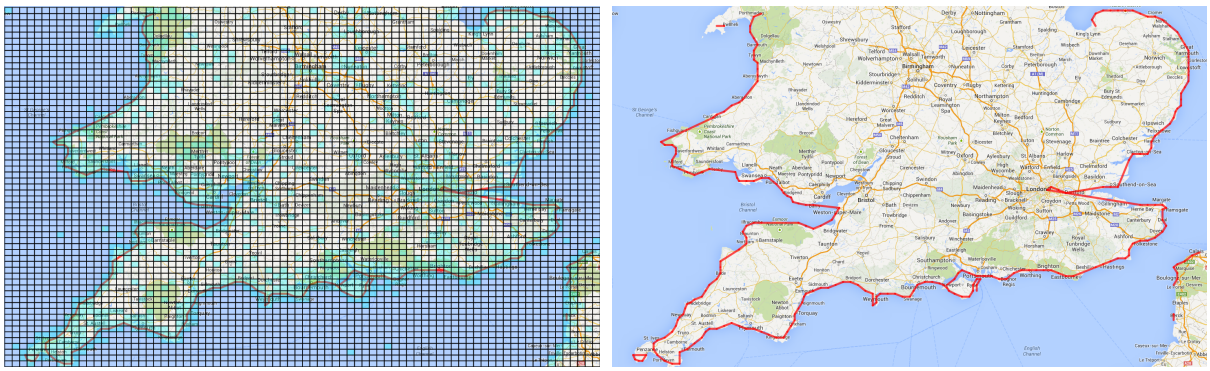


(a) 分割したグリッド　　(b) 本手法で描いた海岸線

図 4　ハワイのマウイ島の写真を用いた海岸線の描画結果



(a) 分割したグリッド (グリッドサイズ大)　　(b) 本手法で描いた海岸線 (グリッドサイズ大)



(c) 分割したグリッド (グリッドサイズ小)　　(d) 本手法で描いた海岸線 (グリッドサイズ小)

図 5　イギリス写真を用いた海岸線の描画結果

用いた．図 5(a) は，分割したグリッドの位置，図 5(b) は，描いた海岸線を表している．ハワイ同様に，概ねの海岸線が描けているが，内陸部にも線が存在する．また，線が交差している場所が存在する．これは，離れたセルのつなげる際に線端の向きだけを利用しており，つなぐ候補となる 2 点が向い合っていれば，その 2 点間に線があるかどうかを考慮していないためである．これについては，今後の課題とする．

図 5(c)，図 5(d) は，図 5(a)，図 5(b) のグリッドサイズを小さくしたものである．図 5(d) と図 5(b) を比較すると，図 5(d) では，図 5(b) に存在した内陸部の線が存在しない．また，グリッドのサイズが小さいため，ウェーマス付近のポートランドやブリストル海峡のような細かい凹凸部分も再現出来ている．しかし，南西の端のペンザンス付近は，線が途切れて島のように表示されている．

内陸部の線が存在しない理由は次のように考えられる．グリッドサイズが大きい場合は，多少写真が散らばっていても 1 つのセルにまとめられるが，グリッドサイズが小さい場合は，写真が複数のセルに散らばり，写真枚数が少ない地域は線を引く候補となる可能性が低い．また，ノイズとなる写真 (e.g., 内陸部で撮影された beach と関係のない写真に「beach」とタグ付けされた写真) の撮影位置は，海岸付近で撮影された写真に比べると線状にはなっていないと考えられる．そのため，内陸

表 4　図 4(b) の線と実際の海岸線との比較結果

Distance	Number of lines
$0m$〜$250m$	40 (69 %)
$250m$〜$500m$	4 (7 %)
$500m$〜$750m$	7 (12 %)
$750m$〜$1km$	2 (3 %)
$1km$〜	5 (9 %)

表 5　図 5(b) の線と実際の海岸線との比較結果

Distance	Number of lines
$0m$〜$250m$	112 (59 %)
$250m$〜$500m$	10 (5 %)
$500m$〜$750m$	9 (5 %)
$750m$〜$1km$	4 (2 %)
$1km$〜	55 (29 %)

表 6　図 5(d) の線と実際の海岸線との比較結果

Distance	Number of lines
$0m$〜$250m$	174 (73 %)
$250m$〜$500m$	22 (9 %)
$500m$〜$750m$	14 (6 %)
$750m$〜$1km$	10 (4 %)
$1km$〜	20 (8 %)

部で線を引く候補となるセルになった場合でも，隣接するセルに線を引く候補となるセルが存在しないため線は引かれない．

海岸線上に写真が少ない地域があった場合は，本手法の線を引く段階では線は途切れるが，離れた線をつなぐ際につなぐことができる．しかし，これは海岸線が直線である場合であり，半島になっている場所では，途切れた場所を最短距離でつなぐと半島部分が島のようになってしまう場合がある．そのため，図 5(b) の南西部のような現象が起きると考えられる．

5. 評　　価

本手法を用いて引いた線と実際の海岸線の位置を比較することで，本手法の評価を行う．本手法で描いたそれぞれのセル内の線と実際の海岸線との最短距離を求める．この距離が近いほど海岸線付近に線が引かれており，提案手法の性能が高いと定義する．実際の海岸線のデータには，OpenStreetMap の海岸線データを用いた．

表 4，表 5，表 6 に図 4(b)，図 5(b)，図 5(d) で引いた線と実際の海岸線との距離の分布を示す．表 4，表 5，表 6 のすべてにおいて，0m〜250m に引かれた線が最も多く，良い結果が得られたことがわかる．しかし，250m〜1km に引かれた線も存在する．これは，第 2 章で述べたように，タグ「beach」が付与された写真は海岸線から 500m 以内に約 80 %であり，線を引くために用いた写真の撮影位置が海岸線から多少ずれていることが原因として考えられる．また，1km 以上にも線が存在し，表 5 で最も多い．これは，図 5(b) において，内陸部に引かれた線が原因と考えられる．これらのことから，実際の海岸線との距離が近い線が多く，提案手法によって，おおよその海岸線が描けたと考えられる．

6. 関 連 研 究

ジオタグとタグから領域を求める研究として Thomee [5] らの研究がある．Thomee らは，Scale-space theory を用いて，地図空間上でタグが表す領域を求めるアルゴリズムを提案し，地図上に可視化した．これに対し，我々は，1 つのタグに着目し，線を引きタグが表す領域を求め，正解データと比較し評価することで，タグ付けの信憑性の評価についても行う．

また，ジオタグとタグを利用した研究として，Zhang [4] らの研究や Shirai [6] らの研究がある．Zhang らは，写真のジオタグやタグから雪，または緑に覆われている地域と時間を求め，地図上に可視化するシステムを提案した．Shirai らは，写真のジオタグやタグから人々が興味を持つ領域を可視化した．これらの研究でも，領域の正確な境界を求めていない．本手法をこれらの領域を求める研究に利用することで，領域のより正確な境界を求めることができると考えられる．また，本論文では，正解データが存在する海岸線に着目しているが，イベントや人々が興味を持つ領域など正解の定義が困難であると考えられる地理的な特徴を持つタグ以外のタグへ適用することで，そのタグが用いられる領域を求めることが可能になると考えられる．

7. おわりに

本論文では，ある任意のタグが付与された写真の撮影位置からタグが表しているものの地理的な形状や広がりを求める手法として，タグ「beach」を持つ写真の撮影位置から海岸線を描く手法を提案した．また，本手法を用いて引いた線と実際の海岸データを比較し，評価を行い，引いた線の多くが実際の海岸線付近であることを示した．

今後の課題として，セル内の線の位置を調整することがあげられる．現在では，写真の撮影位置は，分割した際のセルごとの枚数を調べる時にのみ利用しているが，セルの内部において，どの地点で撮影されたものかといった情報は利用していない．そこで，線を引く時に写真の撮影位置を利用することで，より海岸線に近い位置に線を引くことができると考えられる．また，今回は，候補となるセルを決める時や線をつなぐ時に用いるパラメータを調節していない．そのため，場所や写真数に応じて最適となるパラメータを見つけることが必要であると考えられる．さらに，「beach」以外のタグを用いることや，海岸線以外への適用を目指す．

文　　献

[1] “Flickr”, http://www.flickr.com/
[2] “Panoramio”, http://www.panoramio.com/
[3] “OpenLayers”, http://openlayers.org/
[4] Haipeng Zhang, Mohammed Korayem and David J. Crandall, “Mining Photo-sharing Websites to Study Ecological Phenomena”, Proceedings of the 21th international conference on World wide web, Pages 749-758, 2012
[5] Bart Thomee and Adam Rae, “Uncovering Locally Characterizing Regions within Geotagged Data”, Proceedings of the 22th international conference on World wide web, 2013
[6] M. Shirai, M. Hirota, H. Ishikawa and S. Yokoyama, “A method of area of interest and shooting spot detection using geo-tagged photographs”, ACM SIGSPATIAL Workshop on Computational Models of Place 2013 at ACM SIGSPATIAL GIS 2013, Orlando USA, 2013.11.5
[7] “OpenStreetMap Data”, http://openstreetmapdata.com/data/coastlines
[8] “NOAA National Geophysical Data Center”, http://www.ngdc.noaa.gov/mgg/shorelines/shorelines.html