Seam Carving w przetwarzaniu obrazu

Adam Chojecki

Seam carving to metoda skalowania obrazu polegająca na usuwaniu ze zdjęcia mniej więcej pionowych ścieżek pikseli, które mają najmniejsze znaczenie dla zawartości wizualnej (ang. *seam* = pol. *szew*). W zadaniu studenci mają za zadanie wyznaczyć ścieżkę (tzw. *seam*) przebiegającą od górnego do dolnego wiersza obrazu, która minimalizuje sumaryczny *score* pikseli. Dozwolone ruchy przy przechodzeniu z jednego wiersza do kolejnego to: lewo-dół, dół oraz prawo-dół.

Etap 1. Znalezienie ścieżki o minimalnym sumarycznym score.

Etap 2. Rozszerzenie zadania o dodatkowy warunek — kara za zmianę kierunku ruchu. Przy każdym przejściu, gdy kierunek ruchu różni się od poprzedniego, do łącznego kosztu dodawana jest stała kara $K \geq 1$. Celem jest wyznaczenie ścieżki, która minimalizuje sumę wartości pikseli oraz naliczonych kar. Patrz przykład.

Dane

- `int[,] S` — macierz, gdzie element `S[i][j]` to liczba całkowita nieujemna reprezentująca *score* (ważność) piksela w wierszu i i kolumnie j; oznaczmy: `H` - liczba wierszy, `W` - liczba kolumn
- (Tylko dla etapu 2) `int K` — stała kara, dodawana przy każdej zmianie kierunku ruchu. $K \geq 1$.

Szukane

W obu etapach należy zwrócić krotkę:

`(int cost, (int i, int j)[] seam)`

gdzie:

- `int cost` — minimalny łączny koszt znalezionej ścieżki (suma wartości pikseli, a dla etapu 2 także kara za zmianę kierunku),
- `(int i, int j)[] seam` — ciąg pozycji pikseli stanowiących ścieżkę (włącznie z pikselem z pierwszego i ostatniego wiersza).

Przykład

Rozważmy obraz o wymiarach 5×5 z macierzą score:

$$S = \begin{bmatrix} 3 & 2 & 1 & 3 & 7 \\ 6 & 1 & 8 & 2 & 7 \\ 5 & 9 & 3 & 9 & 9 \\ 4 & 4 & 1 & 5 & 6 \\ 7 & 2 & 3 & 8 & 1 \end{bmatrix}$$

Etap 1: Ścieżka o minimalnym sumarycznym score przebiega przez piksele:

$$(0,2) \to (1,1) \to (2,2) \to (3,2) \to (4,1)$$

i ma łączny koszt: $1 + 1 + 3 + 1 + 2 = 8$.

Zatem w tym przykładzie należy zwrócić:

```
int cost = 8;
(int, int)[] seam = new (int, int)[] { (0,2), (1,1), (2,2), (3,2), (4,1) };
return (cost, seam);
```

Etap 2 z $K = 2$: Rozważmy ścieżkę z rozwiązania Etapu 1:

- $(0,2) \to (1,1)$: pierwszy krok nie podlega karze, gdyż nie mamy wcześniejszego ruchu do porównania
- $(1,1) \to (2,2)$: ruch zmienia kierunek – z ruchu *lewo-dół* na *prawo-dół* – kara 2.
- $(2,2) \to (3,2)$: zmiana kierunku – z *prawo-dół* na *dół* – kara 2.
- $(3,2) \to (4,1)$: zmiana kierunku – z *dół* na *lewo-dół* – kara 2.

Łączna kara wynosi $3 \times 2 = 6$, co daje całkowity koszt $8 + 6 = 14$.

Łatwo zauważyć, że w tym przypadku (biorąc pod uwagę karę za zmianę kierunku) opłaca się rozważyć alternatywny ostatni krok $(3,2) \to (4,2)$. Wartość piksela jest większa, ale mamy mniejsza karę.

Punktacja

Złożoność algorytmu powinna być rzędu $O(H \cdot W)$.

- Etap pierwszy (**1.5p**) — poprawne wyznaczenie minimalnego kosztu (**1p**), poprawna ścieżka (**0.5p**).
- Etap drugi (**1p**).

Uwagi i wskazówki

- Jeśli istnieje więcej niż jedna ścieżka o minimalnym koszcie, można zwrócić dowolną z nich.
- Zakładamy $H \geq W \geq 3$, $K \geq 1$ oraz, że wartości macierzy S są liczbami całkowitymi nieujemnymi.
- Indeksujemy od 0, czyli pierwszy wiersz od góry ma właśnie indeks 0 (patrz przykład).
- Gdy rozważany ruch wychodzi poza granice obrazu, nie jest on dozwolony (nie można wyjść poza obraz).
- W Etapie 2 pierwszy krok nigdy nie podlega karze (nie ma poprzedniego kroku do porównania).
- Warto zauważyć, że gdyby $K = 0$, to Etap 2 sprowadzałby się do Etapu 1. Dlatego Zakładamy $K \geq 1$.
- Najlepsza ścieżka zawsze istnieje, ale nie zawsze jest jednoznaczna.
- Można przyjąć, że wszystkie potencjalne ścieżki mają wartość mniejszą niż `int.MaxValue`.