Univerza v Ljubljani, Fakulteta za elektrotehniko
Podiplomski magistrski študijski program 2. stopnje Elektrotehnika
Informacijsko komunikacijske tehnologije - Obdelava slik in videa (64238)

ZAGOVOR

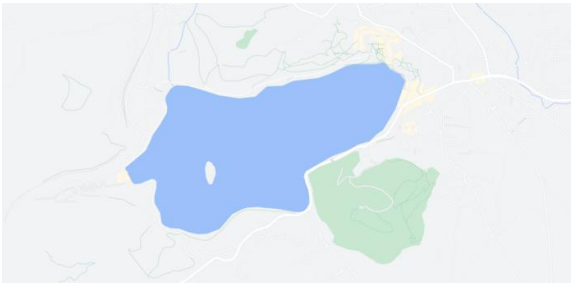
Zagovor laboratorijskih vaj - Naloga 2022-1

Pripravila: Gašper Podobnik & Tomaž Vrtovec

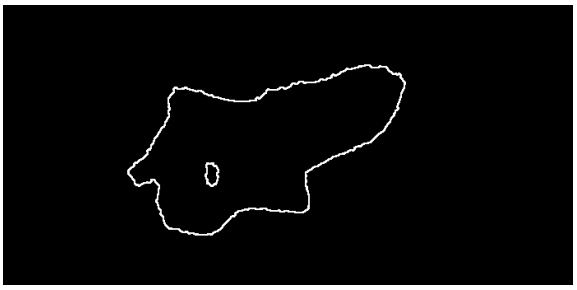
Naloga

Računanje relativnih razdalj med različnimi objekti v sliki je uporabno orodje, ki ga lahko izkoristimo v različne namene. V nadaljevanju boste iz originalne slike, ki prikazuje zemljevid Blejskega jezera in okolice, najprej izločili masko jezera, nato pa izračunali sliko razdalj med točkami, ki ležijo znotraj jezera, ter obalo jezera.

Vhodna slika



Slika meje med kopnim in jezerom



Dana je dvodimenzionalna (2D) barvna (RGB) slika bled-lake-decimated-uint8.jpeg velikosti $X \times Y = 693 \times 340$ slikovnih elementov, ki je zapisana v formatu `jpeg` z 8 biti na slikovni element. Slikovni elementi so izotropni, za merske enote pa povsod upoštevajte *slikovni element*.

1. Napišite funkcijo, ki na izhodu vrne masko območij, kjer na sliki prevladuje modra barva:

```
def get_blue_region(iImage, iThreshold):
    """
    Funkcija, ki vrne masko modrih obmocij na sliki
    """

    return oImage
```

   kjer vhodni argument `iImage` predstavlja vhodno RGB sliko, `iThreshold` pa vrednost praga, ki se uporabi za izločanje maske modrih območij. Izhodni argument `oImage` je matrika velikosti $Y \times X$ (X in Y sta dimenziji vhodne slike), v kateri imajo slikovni elementi, kjer na vhodni sliki prevladuje modra barva, vrednost 255, v nasprotnem primeru pa vrednost 0.

   "Modra komponenta" (RG**B**) slikovnega elementa ima visoko vrednost v primeru, ko le-ta prikazuje modro barvo ter tudi v primeru, ko prikazuje belo oz. (zelo) svetlo barvo. Zato je potrebno upragovanje narediti v dveh korakih in sicer tako da:

   - določite masko *modrih in svetlih območij*, ki ima vrednost `True` v tistih slikovnih elementih, kjer velja $b >$ `iThreshold` pri čemer je b modra komponenta RGB slike, ter vrednost `False` povsod, kjer ta pogoj ne drži,
   - določite masko *svetlih območij*, ki ima vrednost `True` le v tistih slikovnih elementih, kjer so vse tri barvne komponente večje od `iThreshold`, drugje pa ima vrednost `False`,

- dobljeni maski iz prve in druge točke združite na način, da dobite le masko modrih območij (brez belih oz. svetlih območij). Masko skalirajte, tako da bo vsebovala le vrednosti 0 in 255 (glej navodila v prejšnjem odstavku).

Preizkusite funkcijo na vhodni sliki. Nastavite `iThreshold = 235`. Dobljeno sliko prikažite.

2. Izhodna maska iz prejšnje naloge še ni primerna za računanje razdalj od obale jezera, saj poleg maske jezera vsebuje še veliko število majhnih pik oz. svetlih območij. Da se jih znebite, opravite morfološko erozijo ter nato še morfološko dilacijo. Obe operaciji izvedite z enakim filtrom - njegovo velikost in obliko sami določite tako, da na izhodu dobite samo masko jezera (brez dotokov v jezero). Pri tem pazite, da otoček sredi jezera ostane viden. Masko shranite v spremenljivko `lake_mask` in jo prikažite.

   Iz dobljene maske nato s poljubno metodo izločite robove tako, da bodo imeli slikovni elementi, ki ustrezajo robovom na sliki, vrednost 255, vsi ostali pa vrednost 0. Slika robov naj bo enaka oz. podobna sliki meje med kopnim in jezerom, ki je prikazana na začetku navodil. Sliko robov shranite v spremenljivko `lake_edge_mask` ter jo tudi prikažite.

3. Napišite funkcijo, ki vrne matriko vseh koordinat, ki sestavljajo rob, ki ga prikazuje vhodna slika robov:

```
def find_edge_coordinates(iImage):
    """
    Funkcija za iskanje koordinat robnih slikovnih elementov
    """

    return oEdges
```

   kjer vhodni argument `iImage` predstavlja vhodno sliko robov, na kateri vsak slikovni element, ki ima vrednost večjo od nič, predstavlja robno točko. Izhodni argument `oEdges` naj bo matrika x in y koordinat robnih točk sledeče oblike:

$$\texttt{oEdges} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ & \vdots \\ x_N & y_N \end{bmatrix},$$

   kjer je N število vseh robnih točk na vhodni sliki.

   Preizkusite funkcijo na sliki robov `lake_edge_mask` in izpišite velikost matrike.

4. Napišite funkcijo, ki vrne sliko razdalj:

```
def compute_distances(iImage, iMask = None):
    """
    Funkcija za racunanje razdalj od robov
    """

    if iMask is None:
        # TODO: nastavite zeleno vrednost

    # TODO: funkcija za izracun slike razdalj
```

```
    return oImage
```

kjer vhodni argument `iImage` predstavlja vhodno sliko robov, na kateri vsak slikovni element, ki ima vrednost večjo od nič, predstavlja robno točko, `iMask` pa masko točk, ki je enakih dimenzij kot `iImage`. Funkcija naj za vsak slikovni element (x, y), kjer je `iMask[y,x] > 0` izračuna razdaljo do roba, ki je definiran z vhodnim argumentom `iImage`[1]. Privzeta vrednost `iMask` naj bo `None`.
Če uporabnik vhodnega argumenta `iMask` ne določi oz. ga postavi na vrednost `None`, naj se izračunajo vse razdalje na sliki (torej razdalje do vseh slikovnih elementov na sliki).
Namig: na začetku funkcije `compute_distances` iz `iImage` izločite vse koordinate robov na sliki. Pri tem si lahko pomagate s funkcijo `find_edge_coordinates`.

Preizkusite funkcijo na sliki robov `lake_edge_mask`. Vhodni argument `iMask` nastavite na `lake_mask`.
V istem prikaznem oknu prikažite:

- sliko razdalj, ki jo predhodno normalizirate na interval $[0, 255]$,
- sliko robov ter
- točko, ki je najbolj oddaljena od obale. Točko prikažite z rdečim križcem.

Izpišite tudi x in y koordinato te točke ter njeno oddaljenost od obale. Spodnja slika prikazuje pričakovani izris rezultatov.

[1] Opomba: uporabite razdaljo, ki je najkrajša izmed razdalj do vseh robnih točk