

Vaja 8: Houghova preslikava

Pripravila: Gašper Podobnik & Tomaž Vrtovec

Navodila

Postopki za razgradnjo slik, ki temeljijo na iskanju robov v dani sliki na podlagi odvodov, so pogosto nadgrajeni s postopki za povezovanje robov. Eden izmed učinkovitejših postopkov za globalno povezovanje robov, predvsem v obliki premic, pa temelji na *Houghovi preslikavi*.

1. Dana je arhivska datoteka `vaja08.zip` v ZIP formatu. Datoteke, ki se nahajajo v arhivu, shranite v izbrano mapo ter v Pythonu poženite uporabniški vmesnik `vaja08.m`. Uporabniški vmesnik omogoča nalaganje vhodne slike, določanje binarne slike robov, izris slike akumulatorja Houghove preslikave, iskanje lokalnih maksimumov v akumulatorju, izris pripadajočih premic in točk ter shranjevanje slik.
2. Dana je datoteka `houghTransform2D2P.m` z deklaracijo funkcije za dvoparametrično (2P) Houghovo preslikavo v dvonimenzionalnem (2D) prostoru slike:

```
def houghTransform2D2P(iImage, stepR, stepF):  
    # ...  
    # your code goes here  
    # ...  
    return oAcc, rangeR, rangeF
```

kjer vhodni argument $iImage = f(x, y)$ predstavlja binarno sliko robov (za $f(x_i, y_i) \neq 0$ je (x_i, y_i) robna točka), $stepR = \Delta r = r_{n+1} - r_n$ korak spremembe parametra r , $stepF = \Delta \varphi = \varphi_{m+1} - \varphi_m$ pa korak spremembe parametra φ . Izhodni argument $oAcc = A(r_n, \varphi_m)$ predstavlja akumulator Houghove preslikave, $rangeR = [r_1, r_2, \dots, r_N]$ vektor vseh diskretnih vrednosti parametra r , $rangeF = [\varphi_1, \varphi_2, \dots, \varphi_M]$ pa vektor vseh diskretnih vrednosti parametra φ .

Privzeto implementacijo dane funkcije nadomestite z delujočo implementacijo tako, da bo funkcija dejansko izvedla Houghovo preslikavo.

3. Preizkusite delovanje Houghove preslikave na sliki `box1.png`.

Vprašanja

Odgovore na sledeča vprašanja zapišite v poročilo, v katerega vstavite zahtevane izrise in programske kode.

1. Priložite slike uspešnega iskanja najbolj izrazite premice ($n = 1$) na sliki `box1.jpg` in na sliki `box2.jpg`. Zapišite izbrane vrednosti parametrov iskanja robov (prag, standardni odklon), vrednost korakov Δr in $\Delta \varphi$ ter največjo končno vrednost akumulatorja. Kaj lahko na podlagi rezultatov sklepate o Houghovi preslikavi? Obrazložite odgovor.
2. Kaj se zgodi, če zmanjšamo koraka Δr in $\Delta \varphi$ na najnižjo vrednost ter iščemo večje število premic? Obrazložite odgovor.

3. Od česa je odvisna natančnost določanja premic s pomočjo Houghove preslikave? Obrazložite odgovor.
4. Navedite in obrazložite nekaj lastnosti Houghove preslikave.
5. Dana je slika `circles-160x160-08bit.raw` velikosti $X \times Y = 160 \times 160$ slikovnih elementov, ki je zapisana v obliki surovih podatkov (RAW) z 8 biti na slikovni element, velikost slikovnega elementa pa je enaka $\Delta x \times \Delta y = 1,0 \times 1,0$ mm. Naslednja naloga je neodvisna od danega grafičnega uporabniškega vmesnika:
- Določite sliko gradienta s pomočjo Sobelovih operatorjev. Priložite pripadajočo programsko kodo ter izris amplitudne slike gradienta.
 - Upragovite amplitudno sliko gradienta. Priložite pripadajočo programsko kodo ter izris uprakovljene slike. Zapišite tudi uporabljeno vrednost praga.
 - Originalna slika predstavlja tri kroge različnih velikosti, pri čemer ima največji krog polmer $r = 39$ slikovnih elementov. Napišite funkcijo, ki bo na podlagi Houghove preslikave določila koordinate središča največjega kroga:

```
def getCenterPoint(iImage, iRadius):  
    # ...  
    # your code goes here  
    # ...  
    return oCenter, oAcc
```

kjer vhodni argument `iImage` predstavlja binarno sliko robov, `iRadius` = r pa polmer iskanega kroga. Izhodni argument `oCenter` = $[x_0, y_0]$ predstavlja koordinate središča kroga (x_0, y_0) , `oAcc` pa akumulator Houghove preslikave. V pomoč pa sta vam lahko enačbi krožnice: $x = x_0 + r \cos \varphi$ in $y = y_0 + r \sin \varphi$.

Priložite programsko kodo funkcije `getCenterPoint()` ter zapišite koordinate središča iskanega kroga ter vrednost akumulatorja Houghove preslikave na teh koordinatah. Priložite tudi izris slike akumulatorja Houghove preslikave.

Nalogo rešujte brez uporabe naprednih funkcij ter z uporabo funkcij in postopkov, ki smo jih do sedaj spoznali na laboratorijskih vajah.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati k poročilu, prispevajo pa naj k boljšemu razumevanju vsebine.

V skladu z vprašanjem št. 5 napišite funkcijo, ki bo iskala kroge v sliki na podlagi Houghove preslikave:

```
def getCenterPointandRadius(iImage):  
    # ...  
    # your code goes here  
    # ...  
    return oCenter, oRadius, oAcc
```

kjer vhodni argument `iImage` predstavlja binarno sliko robov, medtem ko izhodni argument `oCenter = [x0, y0]` predstavlja koordinate središča kroga (x_0, y_0), `oRadius = r` polmer kroga, `oAcc` pa akumulator Houghove preslikave.

