Peter Rotar, OBSS: Zaznavanje kontur človeških organov z Cannyjevim algoritmom

Peter Rotar

Januar 2023

Povzetek

Za drugi projekt sem izbral detekcijo kontur človeških organov s pomočjo Cannyjevega algoritma[1]. Poročilo na vsakem koraku opiše problem, idejo rešitve in povzame implementacijo v Matlabu.

1 Uvod

Algoritem deluje v štirih korakih. Sprva dano sliko filtrira, nato s pomočjo odvoda (vektor, gradient) poračuna spremembe na sliki in tako za posamezno točko poračuna magnitudo ter kot. Z Non-maximum supression se robovi kontur stanšajo, vendar se lahko zgodi da jih prerazredčimo (razpadejo na manjše, nepovezane robove), zato na koncu algoritem uporablja t. i. Hysteresis tresholding, da jih povežemo nazaj.

2 Implementacija algoritma v Matlabu

V nadaljnih podrazdelkih je na vsakem koraku povzeto delovanje Cannyjevega algoritma. Zadnja koraka sta podrobnejše opisana, saj je za implementacijo bilo dano več svobode.

2.1 Gaussov filter

Slika je sestavljana iz točk (pikslov), katere vsebujejo določene vrednosti. Gaussov filter sestavi novo sliko iz točk originalne tako, da za vsak piskel vzame okolico, vsaki točki v okolici dodeli težo (prispevek) in jih sešteje. Tako dobimo novo sliko katere točke so sestavljene iz linearnih kombinacij točk v okolici na originalni sliki. S tem 'zameglimo' sliko in se hkrati znebimo točk nizkih vrednosti.

2.2 Gradient, magnituda in kot

V naslednjem koraku nas zanima sprememba vrednosti med sosednjimi točkami. Z uporabo numeričnega odvoda dobimo gradient, vektor, ki kaže smer (kot) in magnitudo spremembe.

2.3 Non-maximum suppresion

Sedaj želimo stanjšati debelino kontur. To naredimo tako, da točki pogledamo magnitudo spremembe in jo primerjamo s sosednjima paroma (ki sta si med seboj antipodna). Če je magnituda točke manjša od sosede, jo odstranimo (nastavimo vrednost magnitude na nič).

Seveda sledi vprašanje, kateri izmed osmih sosedov (okolica velikosti 3×3) izbrati? Tukaj nastopi kot gradienta. Ker pri implementaciji uporabimo drugačna bazna vektorja, moramo kot gradienta pravilno nastavit (tako, kot kaže na prosojnicah predavanj). Seveda je pa to le del naše perspektive in kje si postavimo koordinatni sistem. Zato pri moji implementaciji nisem rotiral kote, temveč sosede. Tako na primer za kot $(-22.5, 22.5)$ ne vzamemo navpičnih sosedov, temveč vodoravnih.

2.4 Hysteresis tresholding

Posledica Non-maximum supressiona je, da nam lahko vrne sliko odsekanih, nepovezanih robov. Da se tega znebimo, moramo določene vmesne točke med odsekanimi konturji nastaviti kot pravi robovi.

Ideja je sledeča: Nastavimo zgornji in spodnji trak, ki deli vrednosti točk na tri segmente. Točke, ki so nad zgornjim pragom avtomatsko vzamemo kot kontur in točke z vrednostmi pod spodnjim pragom odstranimo (magnitudo nastavimo na nič). Tako nam preostanejo še vmesne točke. Seveda ne moremo vzeti vseh, temveč tiste, ki imajo v bližnji okolici soseda (točko), ki je kontur. Lahko pogledamo pa točke, za katere vemo da so pravi robovi in pogledamo, če se v njihovi okolici nahaja vmesna (potencialna) točka. Matlab implementacija uporablja slednjo idejo.

Vzamemo indeksno matriko I, ki je velikosti slike in vsebuje 0 in 1. Vrednost $I(i,j) = 1$ nam pove, da je na (i,j) mestu matrike kontur. Če je katerakoli sosednja točka (k,l), $k \in \{i-1, i+1\}$, $l \in \{j-1, j+1\}$ potencialna, jo nastavimo na 1 in nastavimo vrednost magnitude na (k,l) mestu nad zgornjim pragom. Po končanem pregledu sosedov nastavimo $I(i,j) = 0$. Po končani iteraciji dobimo novo indeksno matriko, v kateri je vrednost 1 na tistih mestih, kjer so se v predhodnji iteraciji nahajale potencialne točke v okolici kontur. Iteracijo izvajamo rekurzivno, dokler ni vsota indeksne matrike enaka nič.

3 Zaključek in diskusija

S Cannyjevim algoritmom lahko (dokaj) natančno zaznamo robove. Implementacija je dopuščala ogromno svobode, še posebno Non-maximum suppresion in Hysteresis tresholding. Nekaj težav sem naletel pri izbiri zgornjega pragu. Seveda bi lahko za vsako sliko izbral empiričen prag, vendar se mi je tak pristop zdel neproduktiven. Poskusil sem Otsujev algoritem, ampak mi ni vračal pravilnih vrednosti. Nato sem poskusil z že implementiranimi funkcijami, ki vsebujejo Otsujev algoritem, vendar so mi vračale previsoke vrednosti. Na koncu sem vzel določen percentil urejenih vrednosti magnitud, kar je zadoščalo potrebam.

Literatura

[1] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 8(6):679–698, 1986.