VEŽBA 3 – Detekcija ivica u slici

Potrebno predznanje

- Poznavanje programskog jezika C
- 2D signali
- RGB i YUV prostori boja i konverzija između ova dva prostora
- 2D diskretna konvolucija
- Filtriranje 2D signala

Šta će biti naučeno tokom izrade vežbe

- Šta je predstavlja problemom detekcija ivica u slici
- Detekcija ivica u slici upotrebom Sobel operatora.
- Detekcija ivica u slici upotrebom Cany algoritma

Motivacija

Svrha detekcije naglih promena u osvetljaju kao ivica unutar slike jeste prepoznavanje osobina onoga što je na slici prikazano. U idealnom slučaju primena algoritama za detekciju ivica dovodi do prepoznavanja objekata i njihovih svojstava u smislu oblika, dimenzija, orijentacije itd. Detekcija ivica predstavlja jedan od fundamentalnih koraka u obradi slike koja se odnosi na analizu slike, prepoznavanje šablona unutar slike, prepoznavanje pokreta I tehnikama računarskog vida.

Teorijske osnove

Detekcija ivica u slici

Jedna od primena filtriranja slike jeste detekcija ivica u slici. Detekcija ivica je jedna od važnih metoda obrade slike, jer je to neophodni korak u segmentaciji objekata u slici. Pod detekcijom ivica se podrazumeva generisanje slike na osnovu ulazne, takveda su vidljive samo ivice objekata (vrednost 1 označava tačku na ivici a vrednost 0 tačku koja nije na ivici).

Osnovna ideja je korišćenje gradijenta (prvog izvoda) za meru da li neka tačka pripada ivici ili ne. U tome konceptu prvi izvod po x-osi predstavlja meru za vertikalnu ivicu a izvod po y-osi predstavlja meru za horizontalnu ivicu.

Najpoznatiji operatori za detekciju ivica su poznati kao Sobel operatori:

$$h_v(v,h)=\frac{1}{4}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} \qquad h_h(v,h)=\frac{1}{4}\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

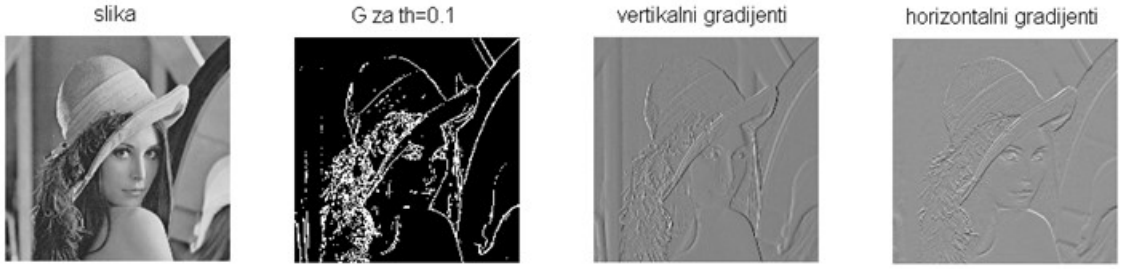
Detekcija ivica upotrebom Sobel operatora vrši se tako što se ulazna slika filtrira horizontalnim i vertikalnim operatorom odvojeno. Rezultat filtriranjem horizontalnim Sobel-operatorom jeste horizontalni gradijent, a vertikalnim vertikalni gradijent. Zato se za detekciju ivica u bilo kom pravcu koristi Euklidska mera gradijenta kao globalni gradijent, koji se izračunava na osnovu horizontalnog i vertikalnog gradijenta upotrebom sledeće jednačine:

$$G(v,h)=\sqrt{G_v^2(v,h)+G_h^2(v,h)}\,.$$

Ako vrednosti normalizovane ulazne slike leže u opsegu vrednosti od 0 do 1, vrednosti vertikalnih i horizontalnih gradijenata leže u opsegu od -1 do 1. Vrednosti globalnog gradijenta su u opsegu od 0 do$\sqrt{2}$. Na kraju, za svaku tačku ispituje se da li binarni globalni gradijent prelazi određenu graničnu vrednost. Ukoliko *G(v,h)* prelazi granicu smatra se da tačka sa koordinatama (v,h) pripada ivici:

$$G(v,h)=\begin{cases}0 & G(v,h)<th\\ 1 & G(v,h)\geq th\end{cases},$$

Menjanjem praga odlučivanja povećava se ili smanjuje osetljivost na ivice. Na slici ispod je prikazan primer detekcije ivica u jednoj slici.



Slika 1 - Detekcija ivica upotrebom Sobel operatora

Prilikom implementacije algoritma za detekciju ivica koristeći Sobel operator neophodno je voditi računa da se rezultat filtriranja slike nalazi u opsegu [-1, 1), tako da je rezultat filtriranja neophodno sačuvati kao označenu vrednost. Algoritam za pronalaženja ivica može se implementirati modifikacijom priložene funkcije za računanje konvolucije, tako da unutrašnja petlja (po *m* i *n*) računa istovremeno rezultat konvolucije i jednim i drugim operatorom.

```
for (int m = 0; m < N; m++)
    {
for (int n = 0; n < N; n++)
        {
            hAccum += extendedImage[(j + n) * newXSize + i + m] * hCoeff[(N - n) * N - m
- 1];
            vAccum += extendedImage[(j + n) * newXSize + i + m] * vCoeff[(N - n) * N - m
- 1];
        }
    }
```

Izračunate vrednosti nije potrebno ograničavati na opseg [0, 255). Umesto toga potrebno je izračunati vrednost globalnog gradijenta i uporediti ga sa zadatom granicom. Ukoliko je izračunati gradijent veći od granice, vrednost izlaznog piksela je 255, u suprotnom, vrednost je 0.

Kako bi se povećale performanse algoritma, najčešće se pre same detekcije ivica, vrši filtriranje slike niskopropusnim filterom, kako bi se uklonio visokofrekventni šum, koji bi mogao biti lažno detektovan kao ivica.

Kani algoritam za detekciju ivica

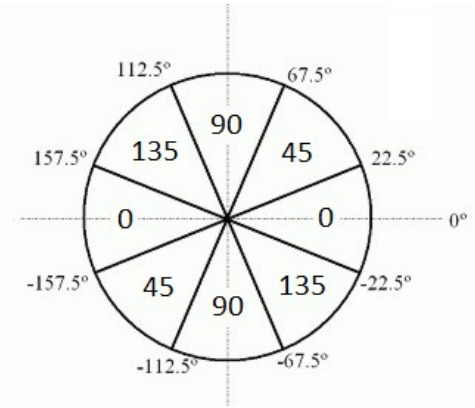
Kani algoritam za detekciju ivica predstavlja napredniji algoritam od prethodno opisanog, i jedan je od najčešće korišćenih algoritama. Algoritam se sastoji iz više faza, a osnovu algoritma kao i u prethodnom slučaju čini računanje gradijenta.

Računanje globalnog gradijenta i orijentacije ivice.

Prva faza algoritma jeste izračunavanje horizontalnog i vertikalnog gradijenta. Najčešće se za računanje gradijenata koriste Sobel operatori kao u prethodnom primeru. Nakon računanja horizontalnog (G_h) i vertikalnog (G_v) gradijenta, računa se globalni gradijent G koristeći jednačinu iz prethodnog poglavlja. Pored globalnog gradijenta, potrebno je za svaku tačku odrediti i orijentaciju ivice. Orijentacija ivice računa se na sledeći način:

$$\theta = \tan^{-1}\left(\frac{G_v}{G_h}\right)$$

Za računanje tangensa, može se iskoristiti funkcija *atan2* iz zaglavlja *math.h*. Nakon računanja ugla ivice potrebno je grupisati ivice u zavnisnosti od pravca: 0^o, 45^o, 90^o ili 135^o. Da bi se ovo uradilo potrebno je dobijeni ugao preslikati u opseg [0, π] dodavanjem π na vrednost ugla ukoliko je ugao negativan. Nakon toga svakoj tački se pridružuje jedan od 4 pomenuta ugla, onaj koji je najbliži izračunatoj vrednosti ugla.



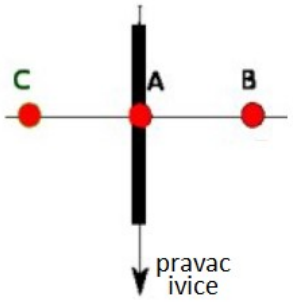
Računanje ugla ivice, i pronalaženje ogovarajućeg pravca dato je u narednom primeru. vAccum i hAccum predstavljaju vrednosti vertikalnogi horizontalnog gradijenta. Vrednost PI_8 je konstanta koja odgovara uglu $\frac{\pi}{8}$.

```
    double edgeAngle = atan2(vAccum, hAccum);
    if (edgeAngle < 0)
        edgeAngle += PI;

    if ((edgeAngle < PI_8) || (edgeAngle > 7 * PI_8))
        edgeDirection[j * xSize + i] = 0;
    else if (edgeAngle < 3 * PI_8)
        edgeDirection[j * xSize + i] = 45;
    else if (edgeAngle < 5 * PI_8)
        edgeDirection[j * xSize + i] = 90;
    else
        edgeDirection[j * xSize + i] = 135;
```

Suzbijanje ne-maksimalnih elemenata

Nakon računanja gradijenata i orijentacije ivica, potrebno je proći kroz čitavu sliku i izvršiti nuliranje svih piksela koji ne mogu činiti ivicu uodnosu na svoje lokalno okruženje. Tehnika kojom se ovo radi, naziva se suzbijanje ne-maksimalnih elemenata. Potrebno je za svaku tačku u slici uporediti vrednost gradijenta, u odnosu na dve susedne tačke koje su normalne na pravac ivice. Ukoliko vrednost nije veća ili jednaka od obe susedne vrednosti, za vrednost gradijenta se postavlja 0.



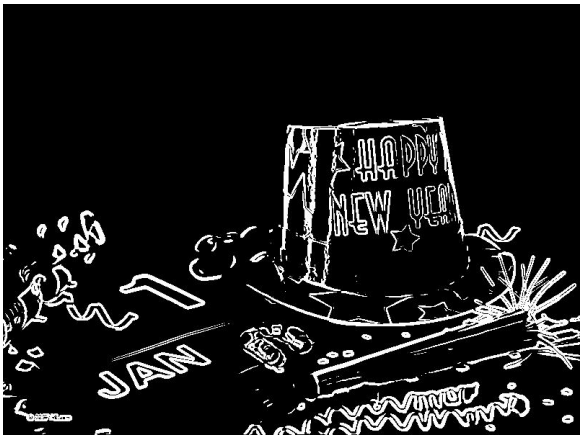
Na prethodnoj slici, tačka A predstavlja tačku koju analiziramo, tačke B i C su susedne tačke tački A, u pravcu normalnom na pravac ivice koja je u ovom primeru vertikalna.Proverava se da li je vrednost gradijenta u tački A manja od vrednosti gradijenata B ili C, i ukoliko jeste, postavlja se na 0. U suprotnom zadržava vrednost. U datom primeru, tačka A pripada ivici, samim tim vrednost gradijenta je veća nego u tačkama B i C. Cilj ovog koraka jeste „stanjivanje“ ivica i eliminacija pogrešno detektovanih ivica.

Izdvanjanje ivica primenom dva praga

Poslednji korak u algoritmu, kao i kod prethodno opisanog algoritma, čini primena praga. Za razliku od prethodnog, kod Kanijevog algoritma postoje 2 praga, gornji i donji. Prag se primenjuje nad vrednostima gradijenata nad kojima je obavljeno suzbijanje nemaksimalnih elemenata.

Određivanje da li je određena tačka ivica, vrši se na sledeći način:

- Svaka tačka čiji gradijent prelazi gornji prag **jeste tvrda ivica**
- Svaka tačka čiji gradijent ne prelazi donji prag **nije ivica**
- Svaka tačka čiji gradijent ima vrednost između donjeg i gornjeg praga **jeste ivica**, **ako** makar jedna od dve susedne tačke, koje se nalaze u **pravcu** ivice jeste čvrsta ivica



Slika nakon primene Sobel algoritma

Ulazna slika

Slika nakon primene Cany algoritma

Zadaci

U okviru ove vežbe vršiće se obrada nad Y komponentom slike. Na samom početku funkcije obrade izvršena je konverzija iz RGB prostora u YUV, a nakon obrade iz YUV u RGB upotrebom funkcija realizovanih u prošloj vežbi.

Zadatak 1

Implementirati funkciju:

- `void performSobelEdgeDetection(uchar input[], int xSize, int ySize, double threshold)`

koja vrši detekciju ivica upotrebom Sobel operatora. Granična vrednost data je parametrom *treshold*. Prilikom poziva funkcije kao parametar *threshold* proslediti vrednost grafičke kontrole *params[0].*

Zadatak 2

Implementirati funkciju:

- `void performNFplusSobelEdgeDetection(uchar input[], int xSize, int ySize, int stages, doublethreshold)`

koja prvo vrši sukcesivnofiltriranje upotrebom NF filtra kao u drugom zadatku, a potom detekciju ivica kao u zadatku 1. Komentarisati uticaj filtriranja na NF filtrom na rezultat detekcije ivica.

Zadatak 3

Implementirati funkciju:

- `void performCannyEdgeDetection(uchar input[], int xSize, int ySize, double threshold1, double threshold2)`

koja vrši detekciju ivica upotrebom Sobel operatora. Donja granična vrednost data je parametrom *treshold1,* a gornja granična vrednost parametrom *threshold2*. Prilikom poziva funkcije kao parametar *threshold1* i *threshold2* proslediti vrednost grafičke kontrole *params[0]* i *params[1].*

Za potrebe potiskivanja ne-maksimalnih elemenata data je funkcija:

- `static void nonMaxSupression(double edgeMagnitude[], uchar edgeDirection[], int xSize, int ySize, double out[])`

Zadatak 4

Implementirati funkciju:

- `void performNFplusCannyEdgeDetection(uchar input[], int xSize, int ySize, int stages, double threshold1, double threshold2)`

koja prvo vrši sukcesivnofiltriranje upotrebom NF filtra kao u drugom zadatku, a potom detekciju ivica kao u zadatku 3. Komentarisati uticaj filtriranja na NF filtrom na rezultat detekcije ivica.