Primjena histograma orijentiranih gradijenata za detekciju objekata

Jahaj, Bljeona

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Primjena histograma orijentiranih gradijenata za detekciju objekata

Završni rad

Bljeona Jahaj

Osijek, 2020.

Sadržaj

1. Uvod

Nove tehnologije sve više uključuju i korištenje računalnog vida. Ljudski vizualni sustav provodi izuzetno složenu obradu slike i baš taj proces u današnje vrijeme želi se što bolje razumjeti i pokušati implementirati u algoritmima kako bi se iskoristila se za tehnološki napredak. Računalni vid je dio znanosti koji pokušava izvući, analizirati i automatski razumjeti korisne informacije iz jedne slike ili niza slika. Kako bi se razumio sadržaj digitalnih slika i videa moraju se razviti metode koje pokušavaju reproducirati sposobnost ljudskog vida. Računalni vid zahtjeva i znanje o obradi slike. Obrada slike je proces stvaranja nove slike iz postojeće slike, koja obično pojednostavljuje ili poboljšava sadržaj na neki način.. U ovom radu obrađuje se posebna tehnika obrade slike, detekcija objekata pomoću histograma orijentiranih gradijenata - HOGs. Ova tehnika detekcije objekata koristi histograme orijentiranih gradijenata za klasifikaciju i segmentaciju objekata na slici. HOGs predstavlja objekt na slikama ili u nizu slika pomoću jednog vektora značajki. Algoritam detekcije objekata pomoću histograma orijentiranih gradijenata bit će detaljno prikazan u ovom radu. Algoritam sadrži sljedeće glavne korake:

- Podjela slike u manje dijelove te računanje gradijenata;
- Proračun histograma gradijenata prema njihovoj orijentaciji - HOGs;
- Normaliziranje histograma u svrhu smanjenja osjetljivosti na promjenu boje i svjetline slike;
- Primjena SVM (engl. *Support Vector Machine*) i HOGs kao deskriptora značajki za detekciju objekta.

Algoritam detekcije objekata pomoću HOGs-a zahtjeva poznavanje filtriranja slika (visoko propusno filtriranje, nisko propusno filtriranje, dvodimenzionalno filtriranje ili korištenje Gaussovog nisko propusnog filtra), karakteristike histograma i metodu za donošenje odluke o detekciji traženog objekta. Pri obradi slike korištene su biblioteka OpenCV, Numpy, Matplotlib te programsko sučelje Python, isto tako treba se koristiti i SVM.

U drugom poglavlju opisani su proces normalizacije te računanje histograma orijentiranih gradijenata. Objašnjena je konvolucija te računanje smjerova i kutova gradijenata. U trećem poglavlju opisan je stroj s potpornim vektorima te je objašnjen proračun za optimalnu hiperravninu te su navedene četiri jezgre koje koristi stroj s potpornim vektorima. U četvrtom poglavlju je

objašnjena funkcija HOG koja se poziva u Pythonu. Navedene su biblioteke, objašnjeni su izračuni te su dati primjeri za različite vrijednosti koje funkcija HOG koristi u izračunu. U petom poglavlju je testirana funkcija HOG na primjeru videa koji je uzet iz materijala EPFL-a (fr. *The École polytechnique fédérale de Lausanne*).

1.1. Zadatak završnog rada

Detekcija objekata je jedan od često korištenih postupka obrade slike u različitim aplikacijama kao što je detekcija automobila, prometnih znakova, ljudskog lica, biciklista i slično. Pri tome detekciji prethodi postupak izdvajanja značajki traženog objekta. Jedan od često korištenih deskriptora značajki objekta je histogram orijentiranih gradijenata (engl. *Histogram of Oriented Gradients* - HOGs).

U radu treba opisati postupak proračuna HOGs deskriptora značajki objekta i njegovu primjena za detekciju objekata. U programu Python potrebno je napraviti algoritam za detekciju objekata primjenom HOGs deskriptora i ispitati njegove karakteristike na nekoliko odabranih primjera.

2. Procesiranje slike

U ovom dijelu rada opisat će se procesi normalizacije te računanje histograma orijentiranih gradijenata (konvolucija i računanje smjerova i kutova gradijenata).

2.1. Normalizacija

Normalizacija boje i svjetline slike je postupak predobrade slike koja smanjuje osjetljivost daljnjih postupaka obrade na promjene osvjetljenja.

Osvjetljenje je veliki izazov kod detekcije, naročito u prometu, gdje je moguće da je detektor pokretan te ide kroz različite zone osvijetljenosti. Pri tome vrijedi voditi računa ne samo o mijenjanju osvjetljenja na cijeloj slici, već i na to da se osvjetljenje može promijeniti i na dijelu slike. Neophodna je normalizacija slike koja će osigurati što manju osjetljivost i na lokalne i na globalne (preko cijele slike) varijacije svjetla. Nekoliko je načina normalizacije. Dalal i Triggs u [1] analizirali su četiri metode normalizacije. Ako je v nenormalizirani vektor deskriptora $\|v\|$ i ako je e proizvoljna mala konstanta čija vrijednost neće djelovati na rezultat, onda normalizacijski faktor može biti jedan od sljedećih:

L2-norma: $$\frac{v}{\sqrt{\|v\|_2^2 + e^2}} \quad (8)$$

L1-norma: $$\frac{v}{\|v\|_1 + e} \quad (9)$$

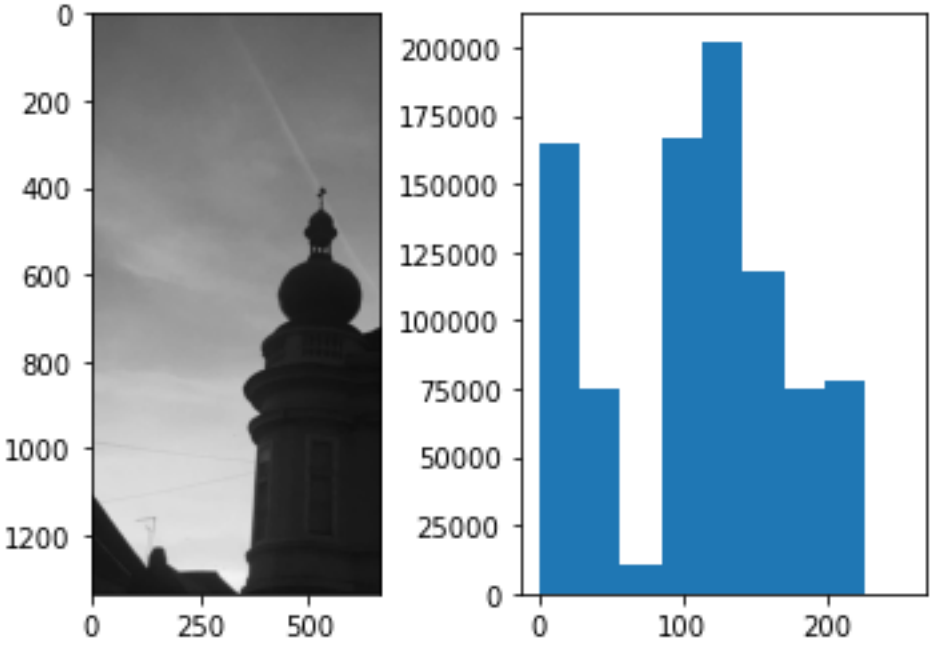
L1-sqrt norma: $$\sqrt{\frac{v}{\|v\|_1 + e}} \quad (10)$$

Postoji još jedna normalizacija koja se spominje u radu Dalala i Triggsa, a to je L2-Hys normalizacija. Ova normalizacija ima istu jednadžbu izvršavanja kao i L2-normalizacija, ali pod uvjetom da se v ograniči do maksimalne vrijednosti 0.2. U [1] je zaključeno da L1-norma od svih normalizacija djeluje najmanje učinkovito, dok L2-normalizacija i L1-sqrt normalizacija daju dobre rezultate. L1-normalizacija djeluje za 5% manje efikasnije od ostalih normalizacija.

Normalizacija dokazano poboljšava rezultate bez obzira na vrstu normalizacije, dok se uspješnost rezultata smanjuje za 27% kada se ne koristi normalizacija.

2.2. Histogram

Pri detekciji slike pomoću orijentiranih gradijenata, moraju se izračunati histogrami. Oni su grafički prikaz distribucije frekvencija pojavljivanja vrijednosti neke varijable. Histogram prikazuje učestalost (frekvenciju) pojave vrijednosti varijable unutar određenog intervala, pri čemu se ukupno dinamičko područje varijable dijeli na željeni broj jednakih intervala U slučaju detekcije slike, grupiraju se gradijenti prema orijentaciji te se računa histogram gradijenata. Na slici 2.1. prikazan je primjer histograma razina sive boje jedne monokromatske slike.

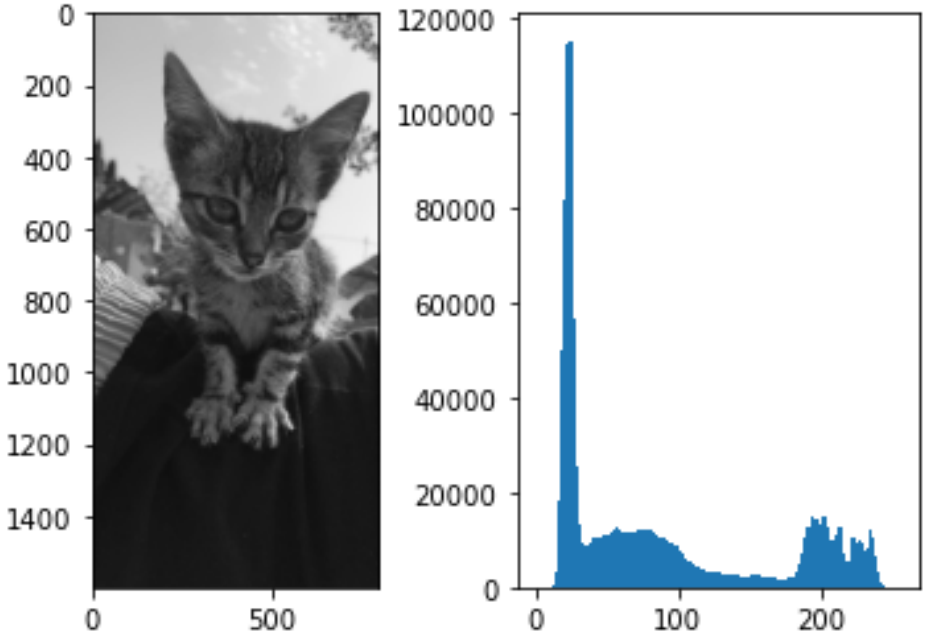


Slika 2.1. Histogram razina sive boje monokromatske slike

Slika ima dimenzije 750x1334 elemenata slike, monokromatska je, s 8-bitnom dubinom boje (slika 2.1.). Raspon intenziteta je [0, 255], gdje 0 predstavlja crnu boju, a 255 predstavlja bijelu boju. Sve između 0 i 255 predstavlja nijanse sive. Zasnovano na ovom intervalu, računa se histogram pojavljivanja određenih vrijednosti u elementima slike. Histogram na slici 2.1. ima 9 intervala (engl. *bins*), i za svaki interval (apscisa) se prikazuje broj elemenata slike (ordinata) koji

imaju intenzitet unutar vrijednosti zadanih tim intervalom. Vidi se s primjera da se bijela boja pojavljuje najmanje u slici, što pokazuje i histogram. Primjer koda korišten u Pythonu za ovu primjenu histograma je u Prilogu 1.

Na slici 2.2. dan je drugi primjer monokromatske slike i njezina histograma.



Slika 2.2. Histogram razina sive boje monokromatske slike

Slika ima dimenzije 799x1598 elemenata slike. Raspon intenziteta je isti kao kod slike 2.1., [0,255], gdje 0 predstavlja crnu boju, a 255 predstavlja bijelu boju. Na slici se najviše pojavljuju crna i tamnosive boje, što je vidljivo na histogramu (slika 2.2.). Histogram na slici 2.1. ima 9 intervala (engl. *bins*), dok na slici 2.2., histogram ima 128 intervala i za svaki interval (apscisa) se prikazuje broj elemenata slike (ordinata) koji imaju intenzitet unutar vrijednosti zadanih tim intervalom.

2.3. Histogram orijentiranih gradijenata

Temeljna karakteristika histograma orijentiranih gradijenata je dijeljenje ulazne slike u više dijelova te izrada histograma koji pokazuju učestalost pojedinog smjera (orijentacije) gradijenata

unutar određenog dijela slike. Slika dimenzija $m \times n$ dijeli se na manje dijelove te se računa histogram orijentiranih gradijenata za svaki od tih dijelova. Slika se obično dijeli u područja 8×8, te se histogram orijentiranih gradijenata računa za svaki dio .

2.4. Konvolucija i računanje gradijenata

Nakon podjele slike u dijelove računaju se vrijednosti gradijenata za svaku dio. Računanje tih vrijednosti provodi se konvolucijom ulazne slike s posebnim filtrima. Prema [9] gradijenti se računaju primjenom filtara (konvolucijskih kernela) koji računaju prvu derivaciju, kao što su filtri D_x i D_y, dani izrazima (2-1) i (2-2).

$$D_x = [-1 \quad 0 \quad 1] \tag{2-1}$$

$$Dy = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{2-2}$$

Konvolucija slike L primjenom filtara D_x i D_y, odvija se prema izrazima (2-3) i (2-4).

$$L_x = L * Dx \tag{2-3}$$

$$L_y = L * Dy \tag{2-4}$$

Vrlo često se za proračun gradijenata koriste Sobelovi filtri. Prema [11] je riječ o diskretnom operateru diferencijacije koji izračunava aproksimaciju gradijenta funkcije intenziteta slike. U svakoj točki slike rezultat operatora Sobel-Feldman je ili odgovarajući vektor gradijenta ili norma ovog vektora. Operater Sobel-Feldman zasnovan je na odvojivim filtrima za x i y smjer, koji sadrže cjelobrojne vrijednosti te je prema tome izračun relativno brz. S druge strane, aproksimacija gradijenta koja nastaje je relativno gruba, posebno za varijacije u području visokih frekvencija na slici. Prema [11], u horizontalno se smjeru kod Sobelovih filtra odvija konvolucija dana izrazom (2-5).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \tag{2-5}$$

U vertikalnom smjeru se odvija konvolucija dana izrazom (2-6).

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I \tag{2-6}$$

Primjer dijela slike veličine 6×6 elemenata slike prikazan je slikom 2.3., a primjer dvodimenzionalnog konvolucijskog filtra slikom 2.4.

L 11	L 12	L 13	L 14	L 15	L 16
L 21	L 22	L 23	L 24	L 25	L 26
L 31	L 32	L 33	L 34	L 35	L 36
L 41	L 42	L 43	L 44	L 45	L 46
L 51	L 52	L 53	L 54	L 55	L 56
L 61	L 62	L 63	L 64	L 65	L 66

Slika. 2.3. Ćelije veličine 6x6

K 11	K 12	K 13
K 21	K 22	K23

Slika 2.4. Primjer konvolucijskog filtra (kernela)

Proračun konvolucije primjenom konvolucijskog filtra (kernela) K je dan izrazom (2-6).

$$O_{i,j} = \sum_{k=1}^{m} \sum_{l=1}^{n} L(i+k-1, j+l-1) \cdot K(k,l) \qquad (2\text{-}6)$$

gdje $m \times n$ predstavlja dimenzije filtra (kernela). Proračun prema (2-6) se provodi za svaki pojedini element dijela slike *i, j*.

Nakon što se izvede filtriranje slike (tj. izračunaju derivacije u x i y smjeru), izračunavaju se gradijenti, odnosno njihova orijentacija i iznos. Iznos gradijenata se računa pomoću izraza (2-6),

$$\|G\| = \sqrt{{L_x}^2 + {L_y}^2} \qquad (2\text{-}7)$$

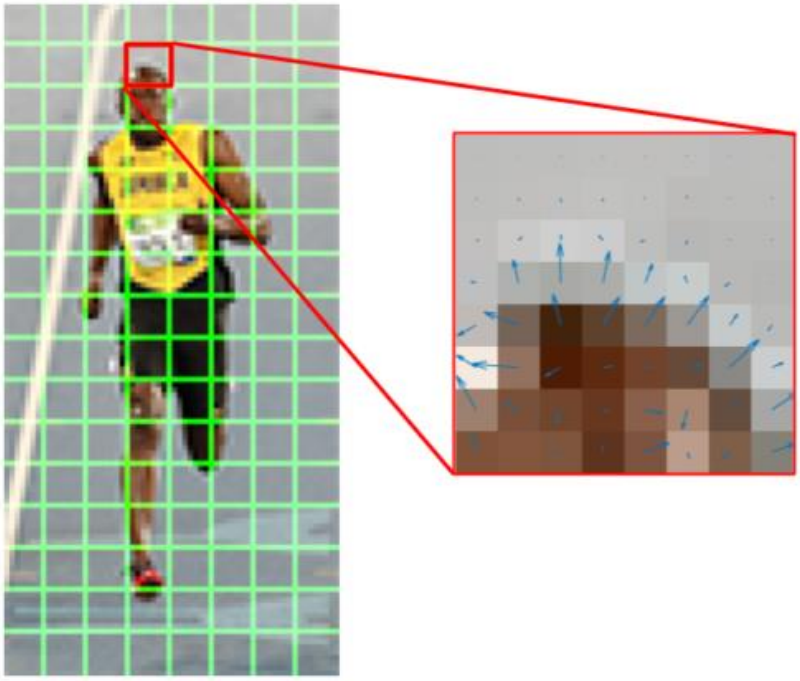
dok se orijentacija gradijenata računa pomoću izraza (2-8).

$$\theta = \tan^{-1} \frac{Lx}{Ly} \qquad (2\text{-}8)$$

Gradijent predstavlja intenzitet promjene signala i to u smjeru njegove najveće promjene. Prema tome na mjestima gdje su gradijenti velikog iznosa ukazivat će na neku skokovitu promjenu u slici, što odgovara mjestima rubova objekata na slici. Informacija o orijentaciji gradijenta ujedno je i informacija o smjeru ruba (smjer ruba će biti okomit na orijentaciju gradijenta). Zbog ovih svojstava gradijent je dobar deskriptor značajki slike za detekciju objekata, jer su svi objekti omeđeni rubovima.

Kada se dobiju vrijednosti orijentacije i iznosa gradijenata, računa se histogram orijentacije gradijenata za svaki pojedini dio slike.

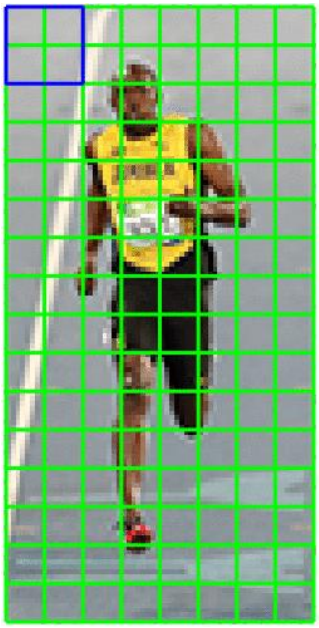
Orijentacije mogu biti (slika 2.5) u rasponu od 0^0 do 180^0 (engl. *unsigned gradient*) ili od 0^0 do 360^0, a prema [1] najbolji rezultati su dobiveni korištenjem raspona od 0^0 do 180^0 podijeljenim na 9 intervala za proračun histograma.



Slika 2.5. Intenziteti i orijentacije gradijenata jednog dijela slike, [11]

Za svaki interval prema [10] računa se kumulativna vrijednost (suma) iznosa gradijenata (elemenata slike unutar područja za koji se računa histogram) koji imaju orijentaciju u danom intervalu.

Tako dobiveni histogrami područja veličine 8×8 se normaliziraju i to na način da se normalizacija radi za združena 4 takva područja (tj. blok 16×16) primjenom pomičnog prozora (slika 2.6.). Na taj se način [10] za svaki prozor dobije normalizirani vektor veličine 36 elemenata (4 dijela slike x 9 vrijednosti za svaki od 9 intervala po dijelu slike). Slika 2.7. prikazuje vektore histograma. Slika se od sastoji od 16 ćelija vertikalno i 8 ćelija horizontalno. U jednoj ćeliji vektor ima 36 položaja, zbroj svih položaja je 3071.



Slika 2.6. Prikaz podjele slike na područja (zelena rešetka) i 4 združena područja za proračun normizacije (plavi kvadrat), [11]

Slika 2.7. HOG, [11]

3. Linearni SVM

Prema [8] stroj s potpornim vektorima (engl. *Support vector machines*) je skup metoda nadziranog modela učenja s povezanim algoritmima učenja koji se koriste za klasifikaciju, regresiju i detekciju. Strojevi s potpornim vektorima su vrlo efikasni alati za korištenje, no njihovi zahtjevi za računanjem, procesiranjem i pohranom povećavaju se s brojem vektora za treniranje.

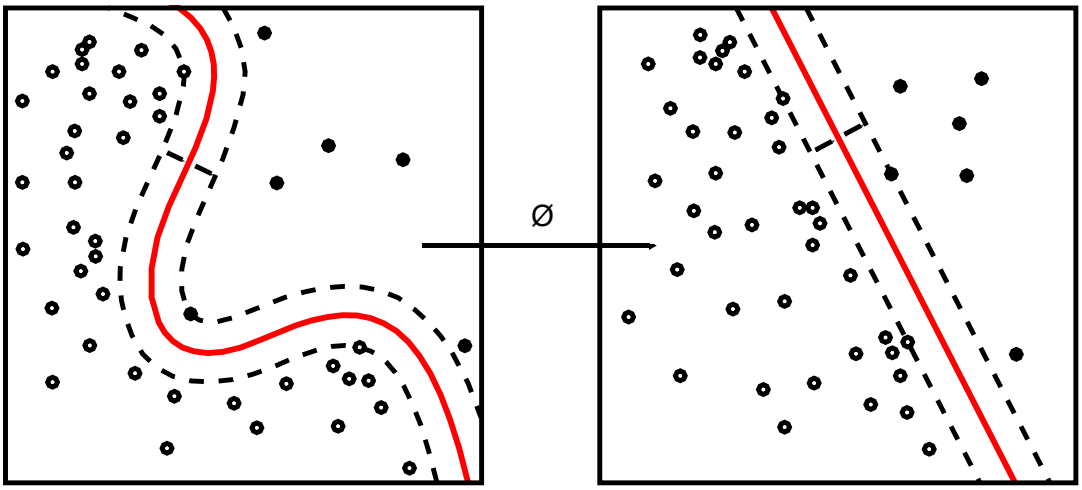
SVM konstruira hiperravninu ili skup hiperravnina u prostorima velikih ili čak i beskonačnih dimenzija, koji se onda koriste za klasifikaciju, regresiju ili za druge zadatke koji zahtijevaju sposobnosti SVM-a. Točno i efikasno razdvajanje postiže hiperravnina koja ima najveću udaljenost do najbliže točke podataka. Prema [2] ova se udaljenost naziva funkcionalna margina. U slučaju SVM-a, što je veća vrijednost margine to je manja pogreška generalizacije klasifikatora.

SVM metoda koristi trening podatke kao točke u prostoru, i u postupku učenja određuje različite kategorije podataka i to na način da su podaci koji pripadaju različitim kategorijama podijeljeni jasnim što je moguće većim prostorom između njih. Tako se podaci, [4], formiraju u zasebne grupe. Ovako trenirani (naučeni) SVM se koristi za klasifikaciju novih podataka, pri čemu se ti novi podaci pridružuju grupama definiranim u postupku učenja.

Osim što provode linearnu klasifikaciju, prema [5] SVM-ovi mogu učinkovito obavljati nelinearnu klasifikaciju koristeći se kernelima, implicitno preslikavajući svoje ulaze u više dimenzijske prostore značajki, kako je prikazano slikom 3.1

3.1. Kako se računa optimalna hiperravnina?

Prema [3] za prikaz i izračun hiperravnine koristi se izraz (3-1):



Slika 3.1. Podjela (klasifikacija) podataka u dvije grupe, [8]

$$f(x) = \beta_0 + \beta^T x, \tag{3-1}$$

gdje je β_0 poznat kao dijagonalni vektor, a β je težinski vektor.

Optimalna hiperravnina može se predstaviti na beskonačan broj različitih načina skaliranjem β i β_0. Sporazumno, među svim mogućim prikazima hiperravnina, onaj koji je izabran je:

$$|\beta_0 + \beta^T x| = 1 \tag{3-2}$$

gdje x simbolizira podatke za treniranje koji su najbliži hiperravnini. Općenito, podaci za treniranje koji su najbliži hiperravnini nazivaju se vektori podrške. Taj je prikaz poznat kao kanonska hiperravnina. Onda se koristi rezultat, dobiven geometrijski, koji daje udaljenost između točke x i hiperravnine:

$$udaljenost = \frac{|\beta_0 + \beta^T x|}{||\beta||} \tag{3-3}$$

Konkretno, za kanonsku hiperravninu, brojnik je jednak jedinici, a udaljenost do vektora podrške je:

$$udaljenost_{vektor\ potpore} = \frac{1}{||\beta||} \tag{3-4}$$

Margina koja je definirana u prethodnom ulomku ovdje je definirana kao M:

$$M = \frac{2}{||\beta||} \tag{3-5}$$

Na kraju, problem maksimiziranja M ekvivalentan je problemu minimiziranja funkcije $L(\beta)$ uz određena ograničenja. Ograničenja modeliraju zahtjev da hiperravnina pravilno razvrstava sve primjere x_i koji su se istrenirali.

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \tag{3-6}$$

$$yi(\beta Txi + \beta\mathbf{0}) \geq \mathbf{1}\ \forall i \tag{3-7}$$

gdje yi predstavlja svaku od značajki istreniranih vektora.

Kad su podaci linearni, moguće ih je razdvojiti odgovarajućom hiperravninom. Postoji puno slučajeva kada podaci nisu linearni i onda ih je nemoguće razdvojiti. U takvim slučajevima koriste se različite funkcije jezgri koje nelinearno preslikavaju takve podatke u višedimenzionalan prostor. Iz [2] četiri su vrste jezgre koje koristi SVM:

1. Linearna $K(x_i, x_j) = x_i^T x_j$
2. Polinomna $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0,$
3. RBF (eng. *Radial Basis Function*) $K(x_i, x_j) = e^{-\gamma(\|x_i - x_j\|^2)}, \gamma > 0$
4. Sigmoid $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

gdje su γ, r i d parametri jezgre.

4. Funkcija HOG u programskom sučelju Python

U programskom sučelju Python postoji već gotova funkcija histograma orijentiranih gradijenata. Funkcija će biti detaljno objašnjena u nastavku, a prikazana je na slici 4.1.

```
hog(image, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(3, 3), block_norm='L2-Hys', visualize=False, transform_sqrt=False, feature_vector=True, multichannel=None)
```

Slika 4.1. HOG funkcija u Pythonu

Funkcija "**HOG**" sastoji se od 9 parametara od kojih su 2 parametra obavezna, a ostali se parametri koriste po izboru.

Parametar *image* je obavezan parametar pomoću kojeg se dohvaća slika iz određene destinacije na kojoj se ulazna slika nalazi.

Parametar *orientations*, nije obavezan parametar, on pokazuje u koliko će se intervala podijeliti gradijenti. U ovom slučaju broj intervala je devet.

Parametar *pixels_per_cell* nije obavezan parametar. On pokazuje dimenzije pojedinog dijela slike nad kojim se računaju gradijenti. U ovom slučaju dijelovi su veličine 8x8 elemenata slike.

Parametar *cells_per_block* nije obavezan parametar. Ovaj parametar pokazuje broj dijelova slike koji se udružuju za proračun normalizacije histograma. U ovom slučaju je broj dijelova devet, odnosno 3×3.

Parametar *block_norm='L2-Hys'* je obavezan parametar. Ovaj parametar određuje vrstu normalizacije histograma.

Parametar *visualize=False* nije obavezan parametar. Parametar *visualize* daje prikaz slike HOG-a, centriranu za svaku ćeliju.

Parametar *transform_sqrt=False* također nije obavezan parametar, a vrši Gamma filtriranje, koje popravlja svjetlinu i sjenčanje na slici.

Parametar *feature_vector=True* vraća informacije u obliku vektora, te se poziva neposredno prije vraćanja programa. Ovaj parametar također nije obavezan.

Parametar *multichannel=None* nije obavezan parametar. Kod ovog parametra vrijednost „True“ označava da je zadnja dimenzija slike interval za određenu boju.

Funkcija „HOG“ poziva se za različite svrhe u programima. U ovom radu koristit će se u svrhu pravljenja histograma orijentiranih gradijenata te analiziranje učinka na detekciju objekata.

4.1. Biblioteke za analizu slike

Za izradu programa za detekcija objekata pomoću HOG-a, u okviru ovog rada, korištene su već postojeće biblioteke koje su namijenjene računalnom vidu, detektiranju, raspoznavanju i analiziranju slike.

Biblioteka koja je najprimjenjivanija za analiziranje slike je OpenCV. Ona sadrži mnoge funkcije potrebne za obradu slike i detekciju objekata i vrlo je detaljno objašnjena i razumljiva. Nadalje, korištene su i biblioteke Matplotlib, NumPy, libSVM, Dlib, skimage. Ove biblioteke su korištene za pomoć u detektiranju i imaju sve potrebne već implementirane i definirane funkcije.

4.2. Dijelovi algoritma funkcije HOG u Pythonu

U ovom poglavlju je prikazano filtriranje slike, odnosno proračun gradijenata pomoću funkcije „Sobel()“ te izračun iznosa i orijentacija za svaki gradijent.

4.2.1. Provođenje filtra preko ulazne slike

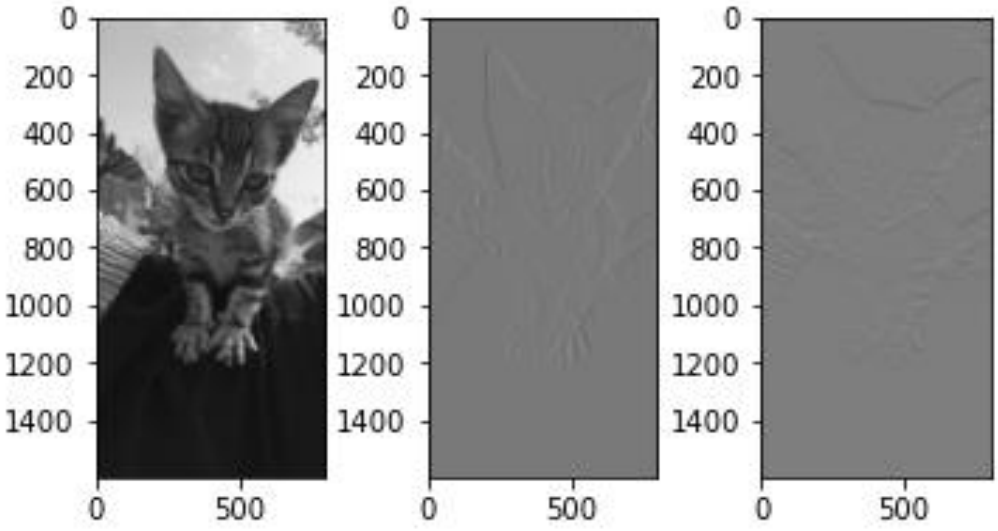
U biblioteci OpenCV već postoje definirane funkcije za provođenje konvolucije. U ovom radu će se raditi na monokromatskoj slici. Slika ima raspon boja od 0 do 255. Ulazna slika je dimenzija 799x1598 elemenata slike. Na slici 4.2. prikazana je originalna slika, rezultat konvolucije sa Sobelovim filtrom u smjeru x osi (slika 4.3.) te rezultat konvolucije sa Sobelovim filtrom u smjeru y osi (slika 4.4.). Može se vidjeti da izražene vrijednosti nakon konvolucije imaju rubovi objekta na slici.

Slika 4.2. Originalna slika za obradu

Slika 4.3. Komponenta gradijenata u smjeru x osi (“ksize=1”)

Slika 4.4. Komponenta gradijenata u smjeru y osi (“ksize=1”)

Slike 4.6., 4.9. i 4.12. prikazuju komponente gradijenata u smjeru x i y osi, s korištenjem parametra “cmap=’gray’”, kako bi se lakše uočile razlike pri mijenjanju parametra “ksize”.



a) b) c)

Slika 4.6. a) Originalna slika; b) Komponenta gradijenata u smjeru x osi; c) Komponenta gradijenata u smjeru y osi

Programski kod za provođenje konvolucije je u Prilogu 2. Filtri se izvršavaju pomoću funkcije Sobel().

sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=1)

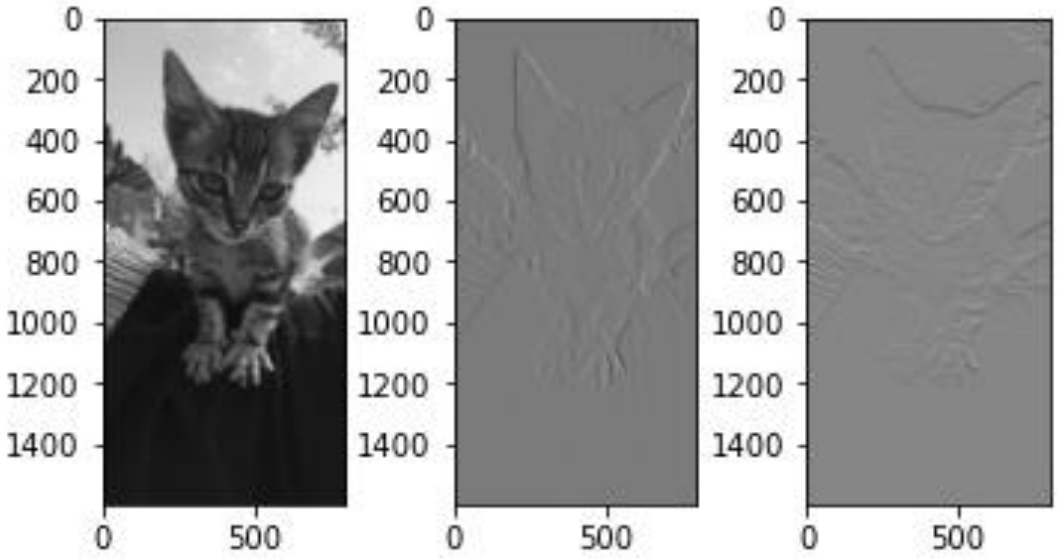
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=1)

Funkcija Sobel() je, kao što se vidi iz navedenog isječka koda, iz biblioteke OpenCV koja je pozvana kao „cv2“ pomoću naredbe „import“. Izvršava se po x i y osi što označavaju drugi i treći parametri u funkciji „0,1“, odnosno „1,0“. Ako je drugi parametar 0, znači da se izvršava filtriranje po y osi jer je treći parametar uključen, odnosno 1. Parametar „ksize“ označava veličinu proširenja filtra koji će se izvršiti na slici i može biti 1, 3, 5 ili 7.

Na slici 4.7 prikazana je konvolucija Sobelovim filtrom u smjeru x osi, na slici 4.8 prikazana je konvolucija slike Sobelovim filtrom u smjeru y osi.Na slici 4.9. prikazana je originalna slika, rezultat konvolucije sa Sobelovim filtrom u smjeru x osi, te rezultat konvolucije sa Sobelovim filtrom u smjeru y osi. Parametar „ksize“ u ovom slučaju je 5. Rubovi su očitiji, ali imamo više nepotrebnih objekata na slici koji su isto vidljivi, npr. pozadina iza objekta ili dijelovi sa strane.

Slika 4.7. Komponenta gradijenata u smjeru x osi (“ksize=5”)

Slika 4.8. Komponenta gradijenata u smjeru y osi (“ksize=5)



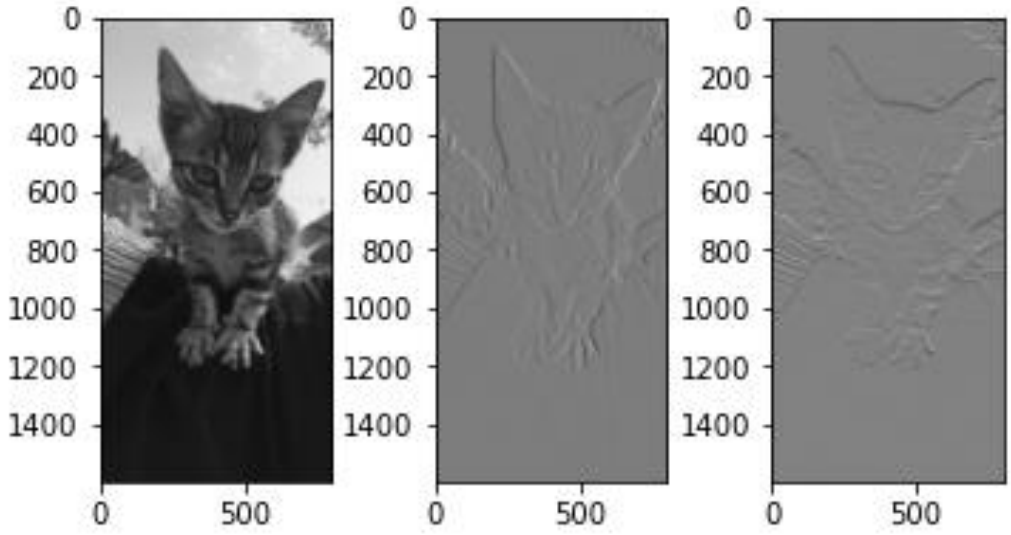
Slika 4.9. a) Originalna slika; b) Komponenta gradijenata u smjeru x osi; c) Komponenta gradijenata u smjeru y osi

Na slici 4.10 prikazana je konvolucija Sobelovim filtrom u smjeru x osi, na slici 4.11 prikazana je konvolucija slike Sobelovim filtrom u smjeru y osi, u oba slučaja parametar „ksize“ je 7. Na slici 4.12. prikazana je originalna slika, rezultat konvolucije sa Sobelovim filtrom u

smjeru x osi, te rezultat konvolucije sa Sobelovim filtrom u smjeru y osi. Parametar „ksize“ u ovom slučaju je 7. Rubovi su puno oštriji nego u prethodna dva slučaja (slika 4.6. i 4.9.).

Slika 4.10. Komponenta gradijenata u smjeru x osi (“ksize=7”)

Slika 4.11. Komponenta gradijenata u smjeru y osi (“ksize=7”)



Slike 4.12. a) Originalna slika; b) Komponenta gradijenata u smjeru x osi; c) Komponenta gradijenata u smjeru y osi

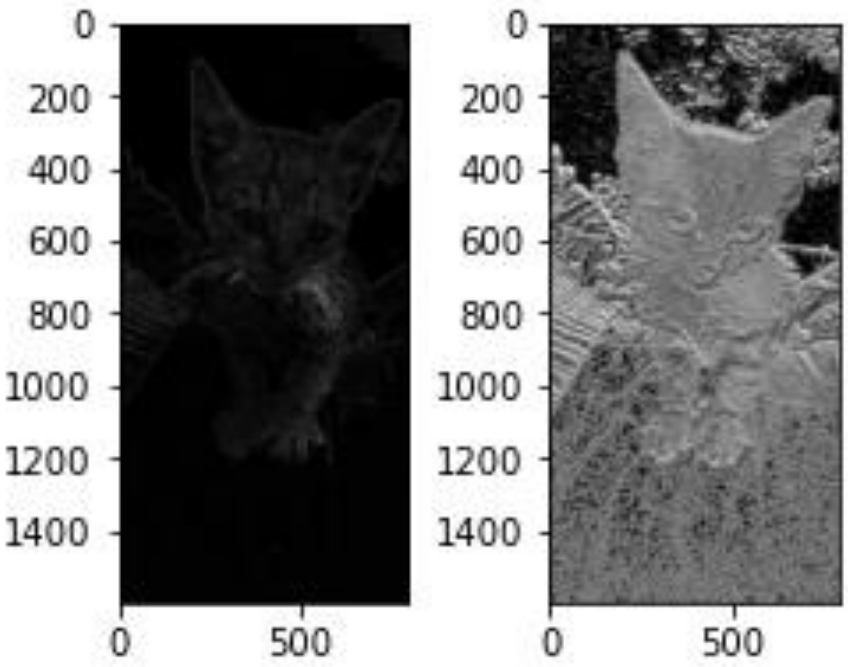
4.2.2. Računanje iznosa i orijentacije gradijenata

Prije nego se pozove funkciju HOG koja je već implementirana u biblioteci OpenCV, najprije će se pokazati kako izračunati iznos i orijentaciju gradijenta u Pythonu.

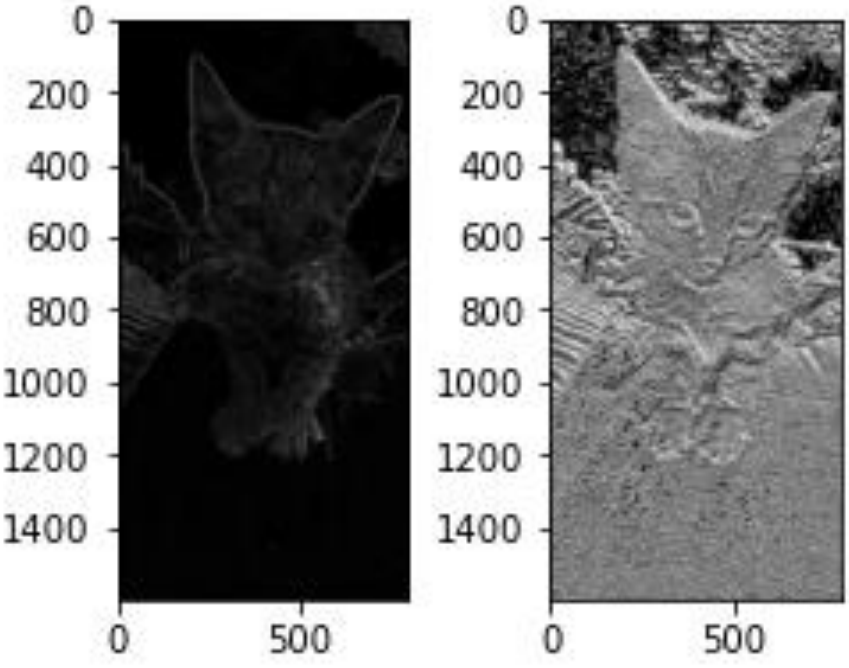
Iznos i orijentacija gradijenata računa se po već navedenim formulama (2-6) i (2-7). U OpenCV-u postoji funkcija koja se poziva i računa iznos i orijentaciju gradijenata bilo koje slike, „*cv2.cartToPolar()*“. Ova funkcija vraća iznos i orijentaciju čim se izvrši, no da bi se ona izvršila, morali su se prethodno izračunati gradijenti po x i po y osi. Ti gradijenti će se prema [7] uvrstiti u funkciju cartToPolar().

mag, angle = cv2.cartToPolar(sobelx, sobely, angleInDegrees=True).

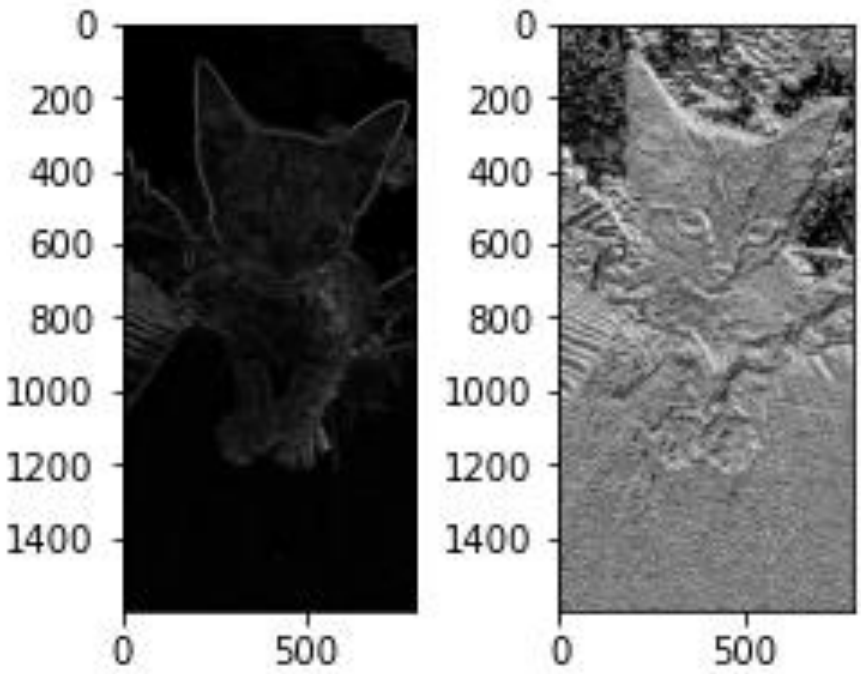
Funkcijama *plt.imshow()* i *plt.show()*, pokazat će se izgled slika kada je izražena samo preko iznosa i orijentacije gradijenata kao na slici 4.13.



Slika 4.13. Prikaz iznosa i orijentacija gradijenata (“ksize=1”)



Slika 4.14. Prikaz iznosa i orijentacija gradijenata (“ksize=5”)



Slika 4.15. Prikaz iznosa i orijentacije gradijenata (“ksize=7”)

4.2.3. Pozivanje funkcije HOG i parametri funkcije

U ovom poglavlju opisan je način kako se poziva već definirana funkcija HOG i kako se uvrštavaju parametri potrebni za izvršavanje naredbe. Funkcija HOG unaprijed je već definirana u različitim bibliotekama koje se koriste u Pythonu. Isto tako u ovom poglavlju koristit će se biblioteke skimage, Matplotlib te OpenCV.

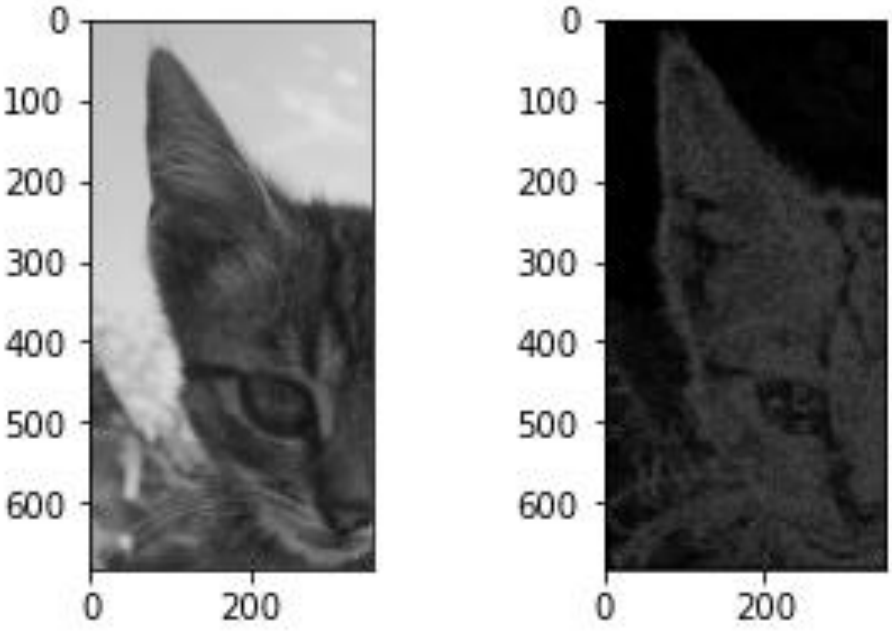
Za primjer pozivanja i korištenja funkcije HOG koristit će se programski kod koji je u prilogu pod brojem tri. Tako će se najlakše prikazati djelovanje i pozivanje funkcije HOG u Pythonu. Kada se poziva funkciju, ona ovisi o različitim parametrima kao što je broj orijentacija, vrsta normalizacije ili broj dijelova slike koji se udružuju za normizaciju histograma ili elemenata slike unutar pojedinog područja za koji se računaju histogrami. O tome ovisi izvedba funkcije.

Primjer mijenjanja performansi kada se mijenja parametar „pixels_per_cell“ dan je slikama 4.16., 4.17., 4.18. i 4.19. Na slikama su prikazane slike koje su procesirane te histogrami orijentiranih gradijenata. Na slici 4.16. uz originalnu sliku (4.16.a)) je prikazan histogram orijentiranih gradijenata (4.16.b)) kada “pixels_per_cell” ima vrijednost (4,4), na slici 4.17.

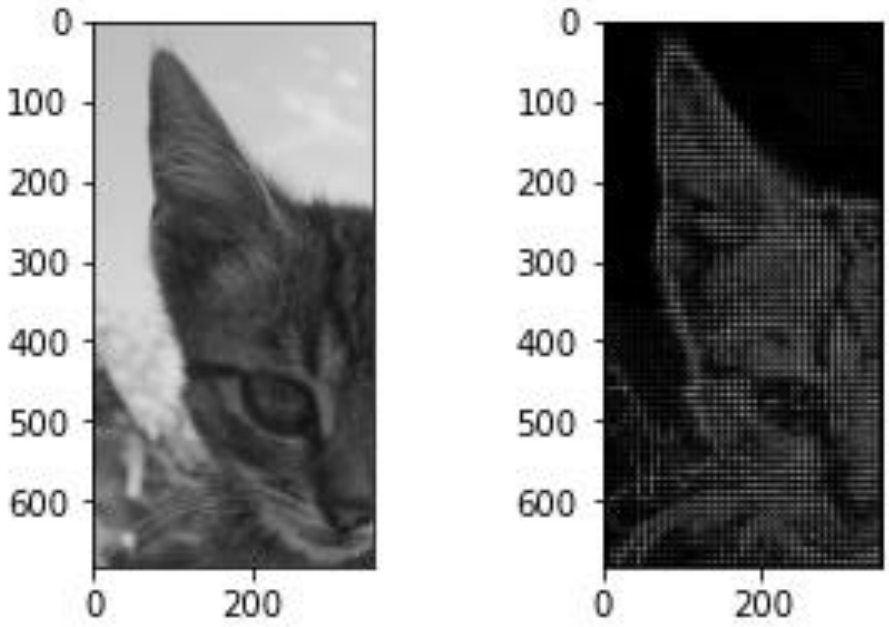
parametar ima vrijednost (8,8), dok je na slici 4.17. parametar “pixels_per_cell” postavljen na (16,16) te je vrijednost parametra od (64,64) na slici 4.19.

Na ovim primjerima se vidi kako različite vrijednosti parametara utječu na dobivene rezultate. Ako se uzme manji broj elemenata slike za obradu, onda će nastati bolji i jasniji rezultati, ali i dolazi do sporijeg procesiranja.

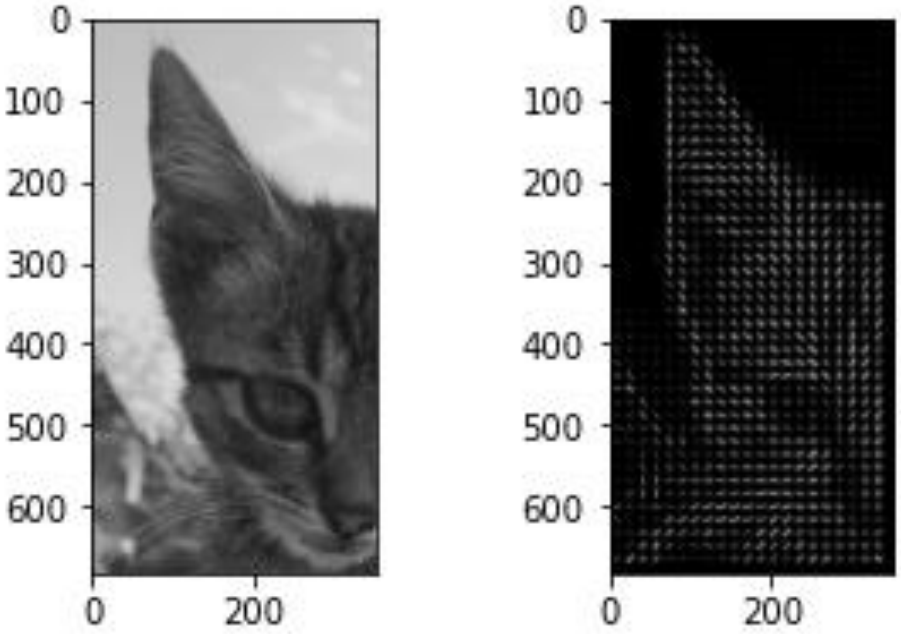
Na prvom primjeru (slika. 4.16.) prikazan je histogram orijentiranih gradijenata uz parametar “pixels_per_cell” jednak (4,4) te se dobila izlazna slika u kojoj je jasno ocrtan obrub objekta i na kojoj se može raspoznati da se mačka nalazi na procesiranoj slici. Na drugom primjeru (slika. 4.17.) parametar je (8,8) te se mačka raspoznaje na slici iako nejasnije nego na slici 4.16., dok je na trećem primjeru (slika 4.18) objekt neprepoznatljiviji jer nedostaju detalji, a i obrub objekta je nejasan. Na zadnjoj slici 4.19., objekt je u potpunosti neprepoznatljiv, nema detalja i rubovi nisu oštri jer je uzet parametar “pixels_per_cell” s vrijednosti od (64,64) što znači da je više ćelija obuhvaćeno u obradi.



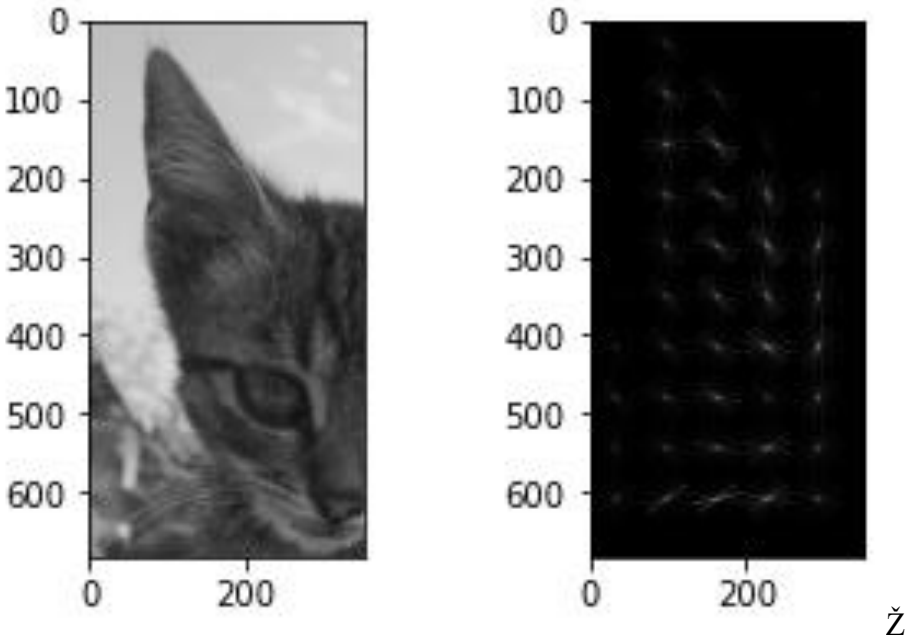
Slika 4.16. a) Originalna slika; b) Histogram orijentiranih gradijenata (4,4)



Slika 4.17. a) Originalna slika; b) Histogram orijentiranih gradijenata (8,8)



Slika 4.18. a) Originalna slika; b) Histogram orijentiranih gradijenata (16,16)



Slika 4.19. a) Originalna slika; b) Histogram orijentiranih gradijenata (64,64)

5. Detekcija objekata pomoću HOG-a

U ovom dijelu rada prikazat će se koliko se uspješno može koristiti HOGs za detekciju objekata (u ovom slučaju pješaka). Koristit će se funkcija „cv2.HOGdescriptor()“ koja poziva ovaj deskriptor značajki. No, da bi se mogli detektirati i izbrojati željeni objekti na slici, mora se koristiti i stroj s potpornim vektorima - SVM koji se također poziva pri procesiranju slike ili niza slika.

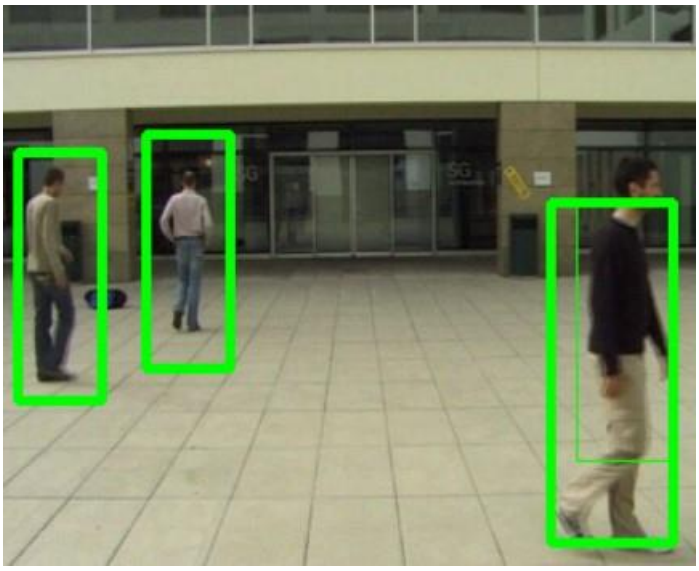
5.1. Detekcija pješaka

U ovom poglavlju, opisan je postupak detekcije pješaka na videu koji je preuzet sa stranice Ženevske institucije EPFL-a [5] na kojoj su postavljeni materijali kao što su fotografije, video materijali i ostalo za procesiranje i korištenje u razvoja algoritama za obradu slike. Rezultati su dobiveni programskim kodom koji se nalazi u Prilogu 4.

U programskom kodu prema [6] koristi se „cv2.HOGDescriptor()“ koji izvršava procesiranje slike pomoću histograma orijentiranih gradijenata. Poziva se ta funkcija te joj se pridružuje stroj s potpornim vektorima tako što se pozove funkciju „cv2.setSVMDetector()“. Primjer detekcije pješaka prikazan je na videu koji se nalazi na DVD-u priloženom uz završni rad.

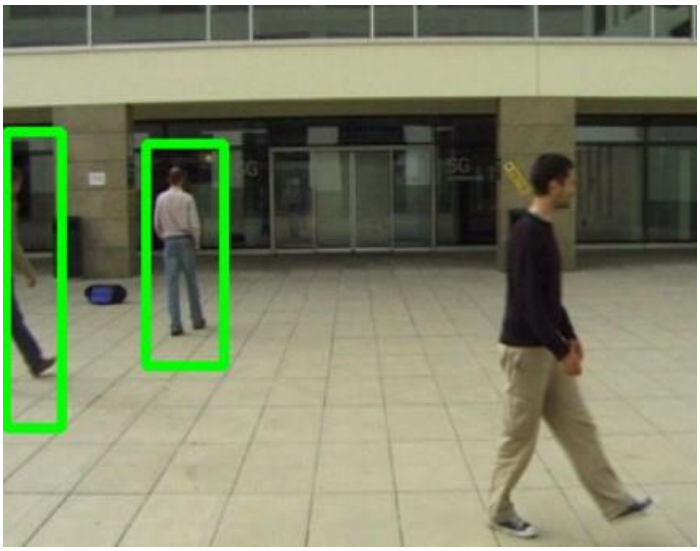
Iako je ovaj algoritam u OpenCV-u već istreniran i dalje postoje slučajevi kada pokazuje netočne rezultate. Ima trenutaka kada lažno detektira osobu ili ju detektira dva puta.

Na slikama 5.1., 5.2. i 5.3. prikazani su različiti rezultati detekcije pješaka te se vidi kako ovaj algoritam ne detektira uvijek točno. Nekad se dogodi da algoritam izbroji dvije osobe na slici umjesto jedne. Na slici 5.1. je algoritam izbrojao 4 osobe, a na slici se jasno vide 3 osobe. Na slici 5.2. se točno vidi jedna osoba i izbrojana je jedna osoba, dok na slici 5.3. osobu na desnoj strani detektor uopće nije detektirao.



Slika 5.*1. Dvostruka detekcija pješaka na jednoj osobi*
Izvor: [5]

Slika 5.2. *Točna detekcija pješaka*
Izvor: [5]



Slika 5.2. *Netočna detekcija pješaka*
Izvor: [5]

6. Zaključak

Cilj ovo završnog rada je objasniti postupak dobivanja histograma orijentiranih gradijenata (HOGs) te njegovu primjenu kao deskriptora značajki za detekciju objekata na slici . Uz algoritam HOGs-a, objašnjen je i način rada stroja s potpornim vektorima - SVM-a koji omogućava detekciju objekta.

U radu Dalala i Triggsa jasno je prikazan algoritam procesiranja slike. Iako se algoritam sastoji od gama korekcije, normalizacije, konvolucije, proračuna histograma gradijenata i korištenje SVM-a, nisu svi dijelovi algoritma jednako značajni. U detektiranju objekata najveći učinak imaju konvolucija, proračun histograma i korištenje SVM-a. Gama korekcija i normalizacija slike samo pridonose boljim rezultatima, ali nisu nužne za detekciju objekata i proces koji dovodi do te detekcije. Korištenje normalizacije poboljšava procesiranje i rezultate za 27%, dok gama korekcija nema značajan učinak na rezultate. Konvolucija se provodi zbog računanja gradijenata. Kako bi se izračunao smjer i amplituda gradijenta, primjenjuje se konvolucija s filtrima tj. kernelima koji odgovaraju prvoj derivaciji. Sljedeći korak je proračun histograma gradijenata prema njihovoj orijentaciji, pri čemu se histogram ne radi za cijelu sliku već za manje dijelove kako bi se dobile značajke na svakom pojedinom dijelu. Dobri rezultati se postižu podjelom na područja 8×8 elemenata slike, ali ta veličina može biti i drugačija ovisno o rezoluciji slike i uvjetima snimanja. Izračunati histogrami predstavljaju dobre deskriptore značajki objekata koji se primjenom SVM mogu koristiti za detekciju i prepoznavanje objekata.

Ovaj način detektiranja objekata je dobar, ali ne osigurava uvijek točnu detekciju. Za poboljšanje rezultata potrebno je istrenirati SVM na najbolji mogući način. Iako je ova detekcija dobra i dalje postoji slučaj kada se pomoću funkcije HOG ne prepoznaje osobe na slikama ili sustav lažno raspoznaje osobu. Ovaj algoritam je među starijim algoritmima za prepoznavanje objekata te su razvijeni točniji i bolji algoritmi.

Literatura

[1] Navneet Dalal i BillTriggs, „Histograms of Oriented Gradients for Human Detection“, Computer Vision and Pattern Recognition (CVPR), 2005., (https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf), [pristup: 5.7.2019.]

[2] Satya Mallick „Support Vector Machines (SVM), Learn OpenCV, srpanj 2018., (https://www.learnopencv.com/support-vector-machines-svm/), [pristup: 5.7.2019.]

[3] OpenCV, „Introduction to Support Vector Machines“, srpanj 2019., (https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html), [pristup: 12.8.2019.]

[4] Sven Lončarić i Marko Subašić, „Neuronske mreže: Stroj s potpornim vektorima (SVM)“, IEEE predavanje, Zagreb, (https://www.ieee.hr/_download/repository/08a-SVM.pdf), [pristup: 16.8.2019.]

[5] EPFL, „Multi-camera pedestrian video“, 2008., (https://cvlab.epfl.ch/data/data-pom-index-php/), [pristup: 16.8.2019.]

[6] OpenCV, „HOGDescriptor Struct Reference“, prosinac 2015., (https://docs.opencv.org/3.1.0/d5/d33/structcv_1_1HOGDescriptor.html), [pristup: 20.8. 2019.]

[7] OpenCV, „Operations on Arrays“, 2014., (https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html), [pristup: 25.8.2019.]

[8] Wikipedia, „SupportVectorMachine“, (https://en.wikipedia.org/wiki/Support-vector_machine), [pristup: 11.7.2020.]

[9] Eclass, „Image normalisation“, https://eclass.teicrete.gr/modules/document/file.php/TP283/Lab/03.%20Lab/lesson3Notes.pdf , [pristup: 15.7.2020.]

[10] Satya Mallick, „Histogram of Oriented Gradients“, Learn OpenCV, prosinac 2016., (https://www.learnopencv.com/histogram-of-oriented-gradients/), [pristup: 2.7.2020.]

[11] OpenCV, „Sobel Derivatives“, prosinac 2019. (https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html), [pristup: 17.7.2020.]

Sažetak

Ovim radom je detaljno prikazan proračun histograma orijentiranih gradijenata (HOG) te primjena ovih histograma kao deskriptora značajki objekta. Objašnjena je normalizacija, konvolucija slike i proračun histograma. Nadalje objašnjena je primjena stroja s potpornim vektorima za detekciju i prepoznavanje objekata primjenom HOG-a kao deskriptora značajki. Objašnjena je i funkcija HOG u programskom sučelju Python. Na nekoliko primjera pokazana je učinkovitost ovog algoritma za detekciju i prepoznavanje objekata.

Ključne riječi: histogram, orijentirani gradijenti, openCV, SVM, detekcija objekata

Summary

This paper presents in detail the calculation of histograms of oriented gradients (HOG) and the application of these histograms as object characteristics descriptors. Normalization, image convolution, and histogram calculation are explained. The paper describes the application of a machine with support vectors for object detection and recognition using HOG as a feature descriptor. The HOG function in the Python programming interface is also explained. Several examples show the efficiency of this algorithm for object detection and recognition.

Key words: histogram, oriented gradients, openCV, SVM, object detection

Životopis

Bljeona Jahaj rođena je 7.11.1997. u Slavonskom Brodu, Hrvatska. Završila je osnovnu školu "Vijenac" u Osijeku, te se upisala u srednju školu "I. Gimnazija Osijek" u Osijeku. Nakon srednje škole, upisala se na Fakultet elektrotehnike, računarstva i informacijske tehnologije u Osijeku. Na drugoj godini studija opredjeljuje se za smjer Komunikacije i informacijske tehnologije.

Prilozi

Prilog 1

```
“@author: Bljeona

import cv2

import matplotlib.pyplot as plt

import numpy as np

#dohvaćanje slike za obradu

impath = 'C:/Users/Bljeona/Pictures/kittent.jpeg'

img = cv2.imread(impath,0)

#plot figure

f = plt.figure()

f.add_subplot(1,2, 1)

plt.imshow(img, cmap='gray')

plt.xticks([])

plt.yticks([])

f.add_subplot(1,2, 2)

#drugi parametar u funkciji hist() je bins; u ovom slučaju je 9,

#ali može ga se mijenjati ovisno u željenim rezultatima

plt.hist(img.ravel(),9,[0,256])

plt.yticks([])

plt.show() “
```

Prilog 2

```
“@author: Bljeona

import cv2

from matplotlib import pyplot as plt


#pomoću cv2.imread() dohvati se slika za procesiranje

img=cv2.imread('C:/Users/Bljeona/Pictures/0123899.jpeg',0)
```

```
#definiranje sobelx i sobely
sobelx=cv2.Sobel(img,cv2.CV_64F,1,0,ksize=7)
sobely=cv2.Sobel(img,cv2.CV_64F,0,1,ksize=7)
#definiranje iznosa i orijentacije
mag,angle=cv2.cartToPolar(sobelx,sobely,angleInDegrees=True)
 #sharey parametar nam omogućuje dijeljenje y osi između slika
fig,axs = plt.subplots(1,2,figsize=(6,3),sharey=False)
axs[0].imshow(img,cmap='gray')
axs[1].imshow(mag,cmap='gray')
axs[2].imshow(angle,cmap='gray')
plt.show()

cv2.imwrite('C:/Users/Bljeona/Pictures/primjerizavrsni/0123899test.jpeg',img)
cv2.imwrite('C:/Users/Bljeona/Pictures/primjerizavrsni/0123899ax1.jpeg',sobelx)
cv2.imwrite('C:/Users/Bljeona/Pictures/primjerizavrsni/0123899ay1.jpeg',sobely)
cv2.imwrite('C:/Users/Bljeona/Pictures/primjerizavrsni/0123899ang1.jpeg',angle)
cv2.imwrite('C:/Users/Bljeona/Pictures/primjerizavrsni/0123899mag1.jpeg',mag)”
```

Prilog 3

```
“@author: Bljeona

import matplotlib.pyplot as plt
import cv2
from skimage.feature import hog
from skimage import exposure
#dohvacanje dijela slike
img = cv2.imread('C:/Users/Bljeona/Pictures/kittentcrop.jpeg',0)
img_bgr=cv2.cvtColor(img,cv2.COLOR_RGB2BGR)
```

```
#pozivanje funkcije hog iz skimage biblioteke

#multichannel parametar je False jer je slika monokromatska

fig,
hog_img=hog(img,orientations=8,pixels_per_cell=(64,64),cells_per_block=(1,1),visualize=True,
multichannel=False)

fig1, axs = plt.subplots(1,2,figsize=(6,3),sharey=False)

axs[0].imshow(img_bgr,cmap='gray')

hog_img_rescaled=exposure.rescale_intensity(hog_img,in_range=(0,10))

axs[1].imshow(hog_img_rescaled,cmap='gray')

plt.show()

plt.imshow(hog_img_rescaled,cmap='gray')

cv2.imwrite('C:/Users/Bljeona/Pictures/primjerizavrsni/kittentcroptest.jpeg',hog_img_rescaled)

plt.savefig('C:/Users/Bljeona/Pictures/primjerizavrsni/kittenthogtest.png')”
```

Prilog 4

```
„import cv2
def insider(r, q):
    rx, ry, rw, rh = r
    qx, qy, qw, qh = q
    return rx > qx and ry > qy and rx + rw < qx + qw and ry + rh < qy + qh

def rectangles(img, rects, thickness = 1):
    for x, y, w, h in rects:
        pad_w, pad_h = int(0.15*w), int(0.05*h)
        cv2.rectangle(img, (x+pad_w, y+pad_h), (x+w-pad_w, y+h-pad_h), (0, 0, 0), thickness)

hog = cv2.HOGDescriptor()
hog.setSVMDetector( cv2.HOGDescriptor_getDefaultPeopleDetector() )

cap = cv2.VideoCapture('C:/Users/Altina/OneDrive/Radna površina/b/campus4-c0.avi')

while(cap.isOpened()):
    ret, img = cap.read()

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    found, w = hog.detectMultiScale(img, winStride=(8,8), padding=(32,32), scale=1.2)
    found_filtered = []
    for ri, r in enumerate(found):
        for qi, q in enumerate(found):
            if ri != qi and insider(r, q):
                break
            else:
                found_filtered.append(r)
        rectangles(img, found)
        rectangles(img, found_filtered, 3)
        print('%d (%d) found' % (len(found_filtered), len(found)))
    cv2.imshow('The input image or video', gray)
    cv2.imshow('Frame',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()“
```

Prilog 5

CD s videom na kojem je prikazan primjer detekcije ljudi primjenom HOG i SVM..